NEA – Coffee Shop Order App

Table of Contents

Analysis

Description of Problem

In the districts of Ghubrah, in Muscat, Oman, there lies a coffee shop that I frequently go to. The coffee shop cooks meals from south-east asian countries. Frequently in the evening, the coffee shop gets a lot of calls asking for a home delivery order. The worker would note down the order and delivery address on a piece of paper that would later be thrown away after completion.

My project is inspred from this little situation. What if some of the customer calls didn't get through because the line was busy? I seek to build a system to improve the current state of the customer to coffee shop interaction.

Current and Proposed System

At the moment, the coffee shop simply receives a phone call from a customer and notes down the order and address details on a piece of paper. After the delivery of the food is complete, the piece of paper containing the details would simply be thrown away. This makes the process repititive and highly redundant. If a frequent customer were to order from the coffee shop once a day, they would have to repeatedly state their address each day.

I believe creating a phone application can solve this problem. Having a phone application will allow two customers to order at the same time, instead of one connecting to the line and the other getting a "the line is busy at the moment" on the phone call. Customers can create accounts on the app that will have data of their address so that when they make an order the coffee shop will already know where to deliver the food to. As features, customers can also view what they had ordered in the past in case they would like to see what the tasty thing they ordered back then was.

Customers can also take a pamphlet that contains the menu of the coffee shop when they want to order for home delivery via mobile phone. However, if there is a new item, they will not know. Therefore, the coffee shop workers should be able to update their menu on the app so that the latest version will always be there for the customers.

Client, Users and Needs

Client – Coffee Shop owner

Users – Workers of the coffee shop, Customers of the coffee shop

The primary user of this sytem will be the customers. By using this system, they will make the life of the coffee shop workers easier. They make orders for the workers to view on the app, the workers can receive multiple orders at once, compared to the old system where they can only

receive one order per time. They also have accounts, and order history is stored. This helps, as if they have ordered at least 15 times on that account, they will get a discount, and this will automatically calculate the new prices.

Another primary user of this system will be the workers. Their main use is simply to view the customer's orders. However, they can also update their menu items on the app, so that the customers will get the newest version of the menu straight away. They can also send notifications to the customers to give them a status on their order. Other than that, they can also delete user accounts in case the storage space is getting full.

Potential Solutions

1) Develop Current System
    a. Have multiple phones to contact
    b. Install an answering call (like they have in KFC or Papa Johns)
    c. Store phone number and address in an address book. This however will become overflowed if a lot of unique customers make an order, and flipping through the pages to find the address will be annoying. Even if it is sorted in chronological order, it will be a lot of pages to flip through.
2) Website
    a. Everyone can access a website from their home
    b. The coffee shop would require to have a computer in their shop to check orders and editing menu
    c. Not a lot of open space in the shop
3) Phone Application
    a. If this was created, only customers with the same operating system can use it. E.g. if an Android app is made, Apple or Windows users cannot use it.
    b. Practical – Customers can order on their way home from work, for example.
    c. If home WiFi is down, customers can use 3G
    d. Doesn't take up shop space

The best solution I believe would be option 3. I have decided to make an Android app, as the workers of the coffee shop have an Android phone.

System Limitations

- Only customers that have an Android phone may use the app. People with other OS such as Apple or Windows cannot.

- Customers can only order for home delivery if they have access to the internet, meaning customers without wifi or 3/4G cannot.
- If the number of customers exceeds 5GB (which is highly unlikely as it is just storing text), the coffee shop would have to start paying monthly for the online database
- The coffee shop would need internet for the workers' phone during work hours so that they can receive orders, this may get costly.

Data Source and Destination

Source (Input)	Process	User	Output	Destination
New Menu item	User enters details of item	Workers	New menu item	Screen output, new database record
Edit Menu items	User updates (change or delete) details of items	Workers	Updated menu items	Screen output, updated database record
Send Notification	User selects/types message, user presses button	Workers		Screen output on for customer
View Customer Details	User presses button	Workers		Screen output
Delete Account	User selects account to delete, Confirmation, Typing in master password	Workers		Updated database record, Screen output
New Account	User enters account details	Customers	New account	Screen output, new database record, access to
Edit Account	User changes account details	Customers	Updated account	Screen output, updated database record
Make Order	User chooses items from menu	Customers	New order for worker to view	Screen output, updated database record
View History	User presses button	Customers	-	Screen output

Volume of Data

The volume of data in the system isn't really that large. Most of the data in the system will be simple text.

Both the Input and Output volume is the same. Texts including user's details and menu items and price is what will be downloaded and uploaded to and from the database.

Data Dictionary

Menu & Order Data

Data Item	Data Type	Formatting	Comments	Description
Menu Item	String	String less than 15		The names of each Item e.g. "Lemon Chicken"
Price	Float	2 digits	Measured in Rials	The cost of each item e.g. "2.3"
Quantity	Integer	Greater than 0	To disable useless information orders, e.g. The user wants 0 Lemon Chicken. Or to disable messing up calculation of cost, e.g. The user wants -3 Lemon Chicken, hence the cost which is (-3*Y) is added to the total cost which would reduce the total cost.	The amount of the item the user would like to order e.g. "2"
Total Price	Float	4 digits		The total cost of the order e.g. "12.6"
Order Number	String			"Order 2" or "Order 31"
Date	String	10 digits	It is easier to work in string for comparing and using in texts	26/02/2016
Time	String	4 digits	It is easier to work in string for comparing and using in texts	1632

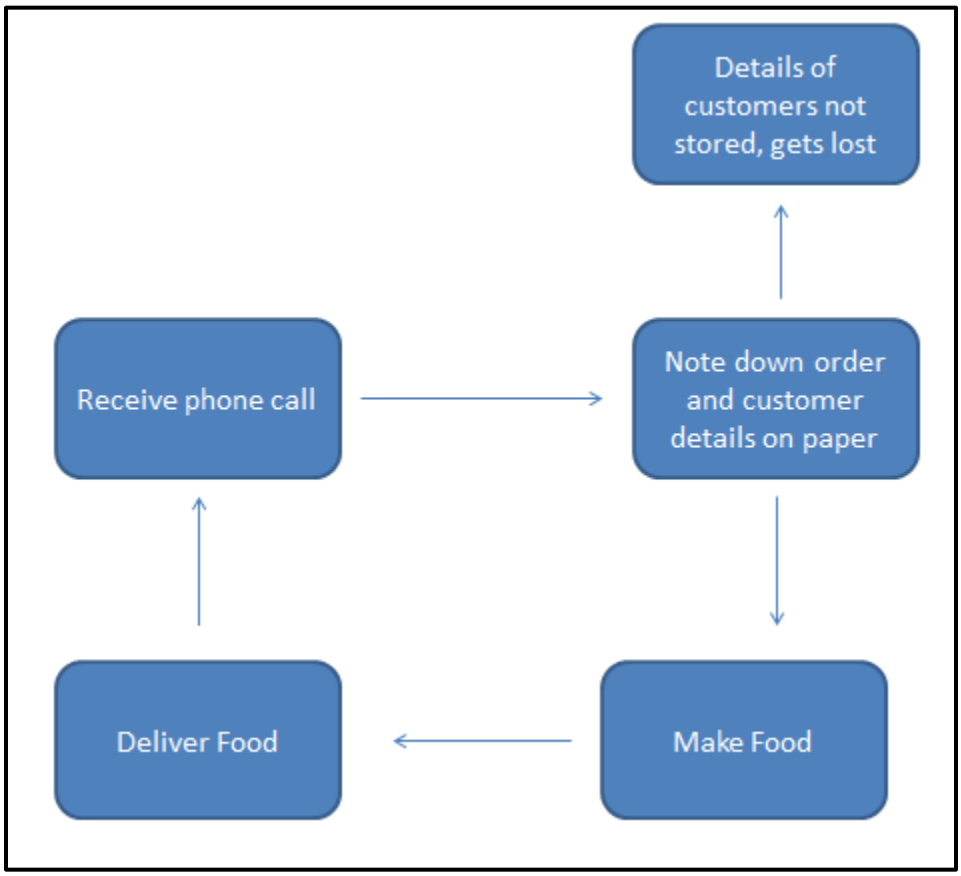
Customer Data

Data Item	Data Type	Formatting	Comments	Description
Name	String	None		The name of the

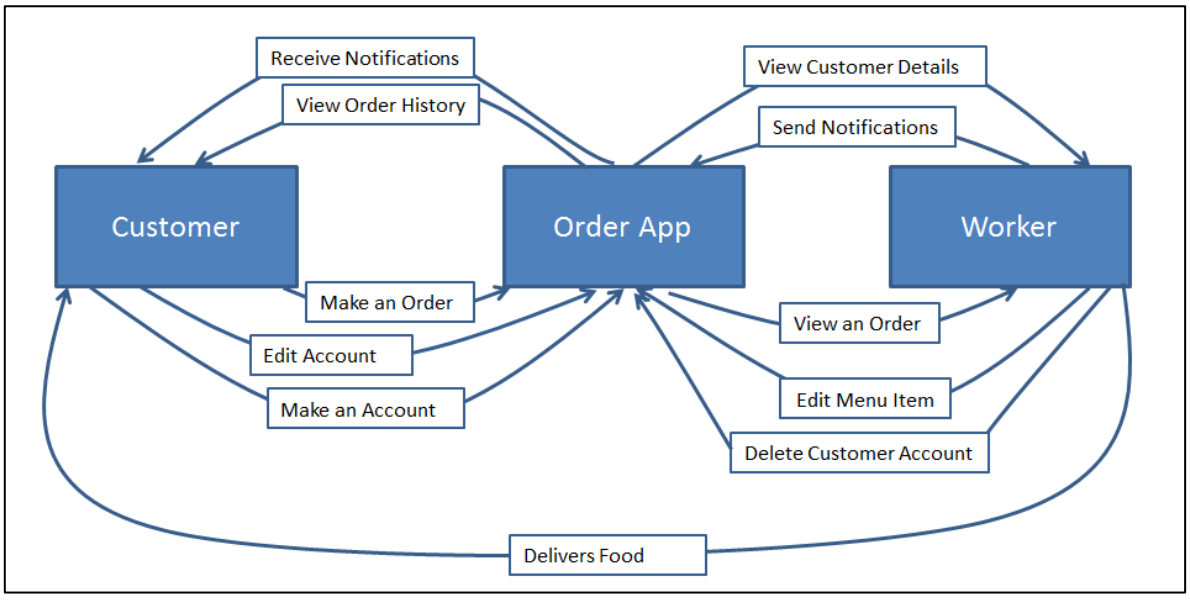
				user e.g. “Stevenson”
Username	String	String less than 12		The username of the user e.g. “TinyMusic”
Password	String	Digits > 6	Has to be at least 6 characters	The password of the user e.g. “coool31”
Region	String	Option Selection	Only 2 options of Azaiba or Ghubrah	The Region of the user e.g. “Azaiba”
Street Address	String	None		The Street Address of the user e.g. “Way 4618”
House Address	String	None		The House Address of the user e.g. “Villa 1100A”

Data Flow Diagrams

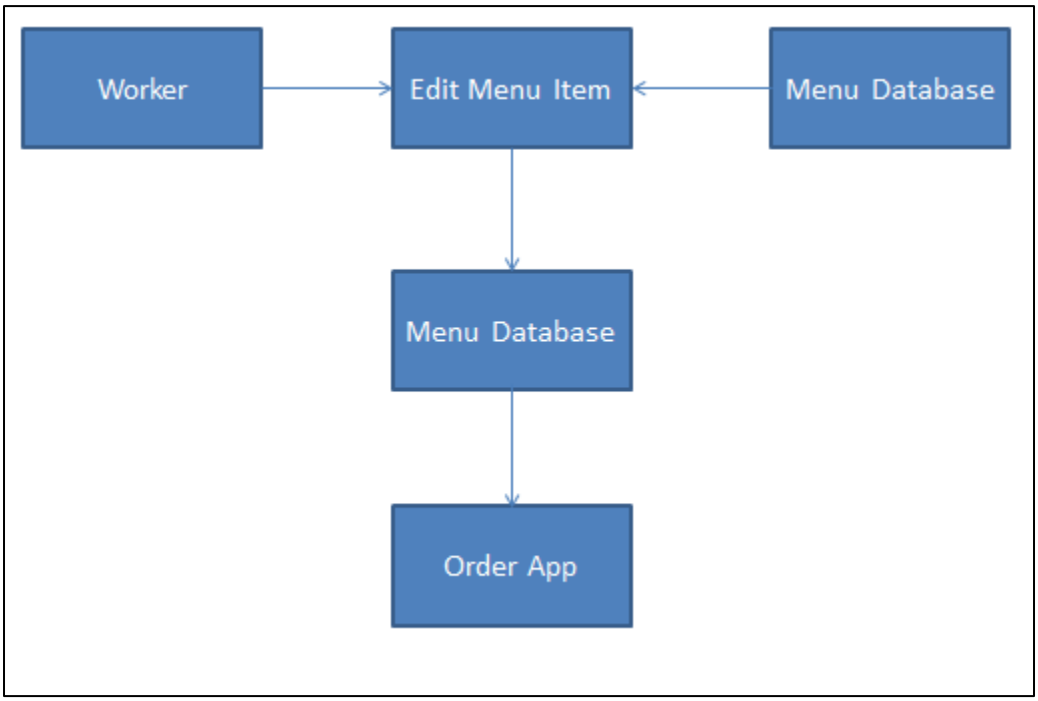
Current System



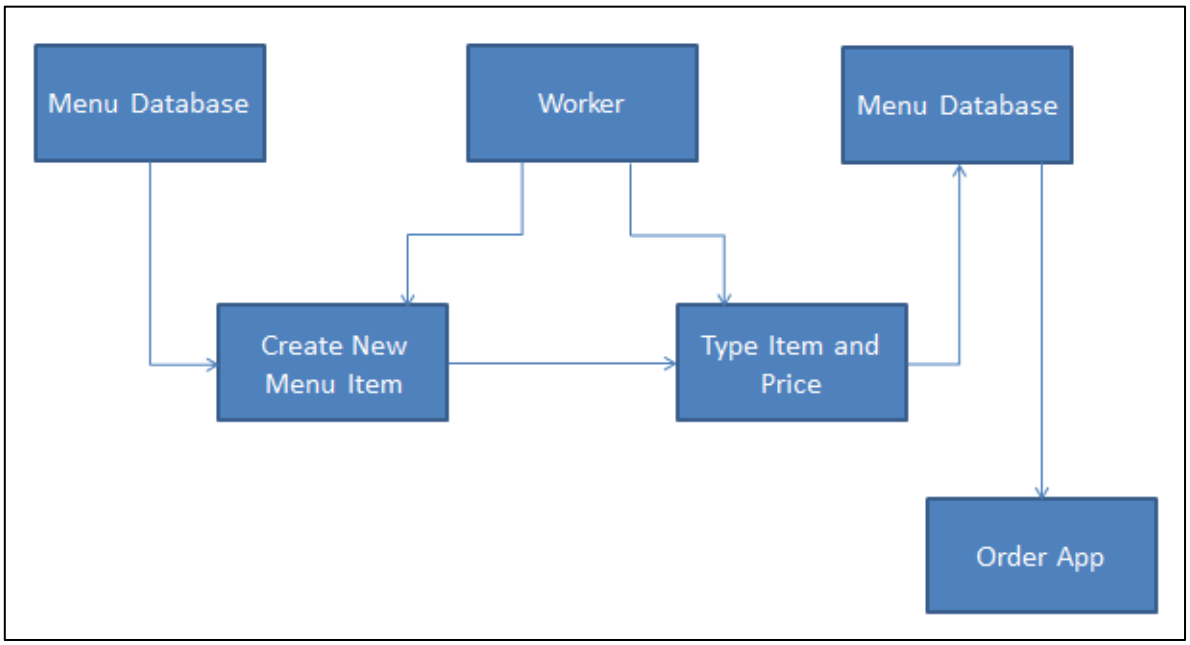
Context View

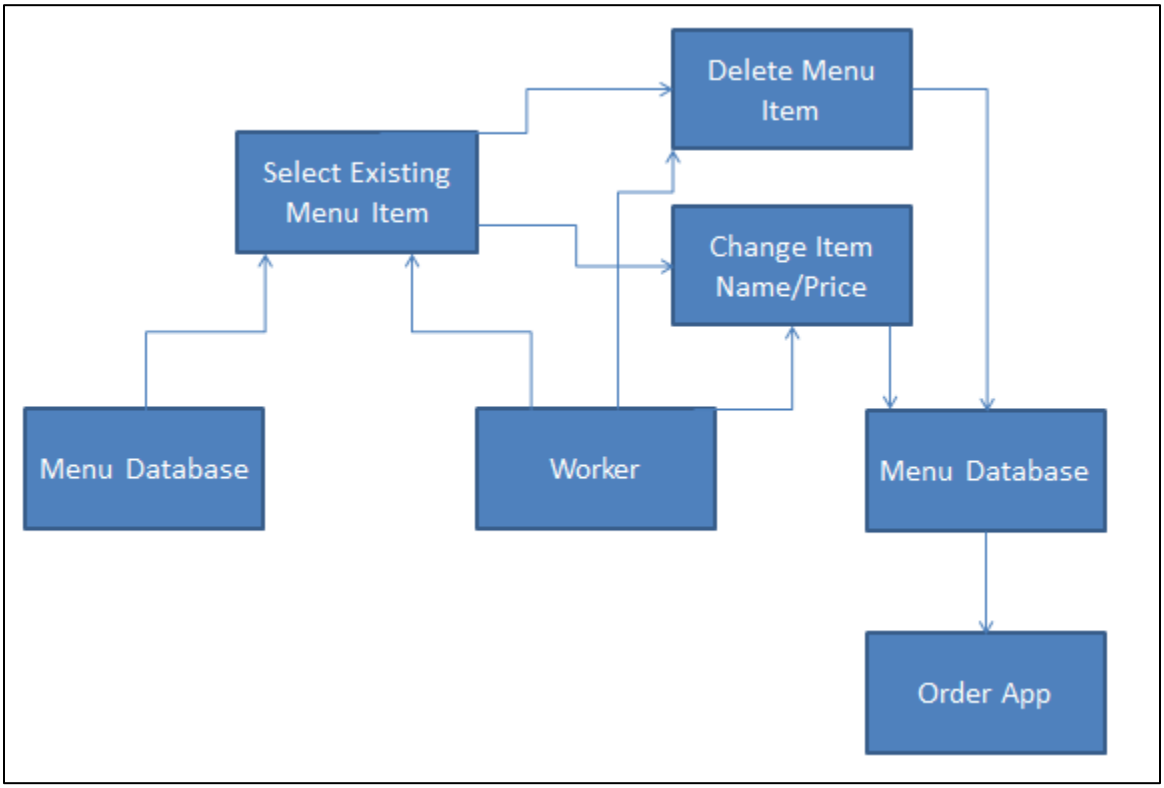


Edit Menu Item – Level 0

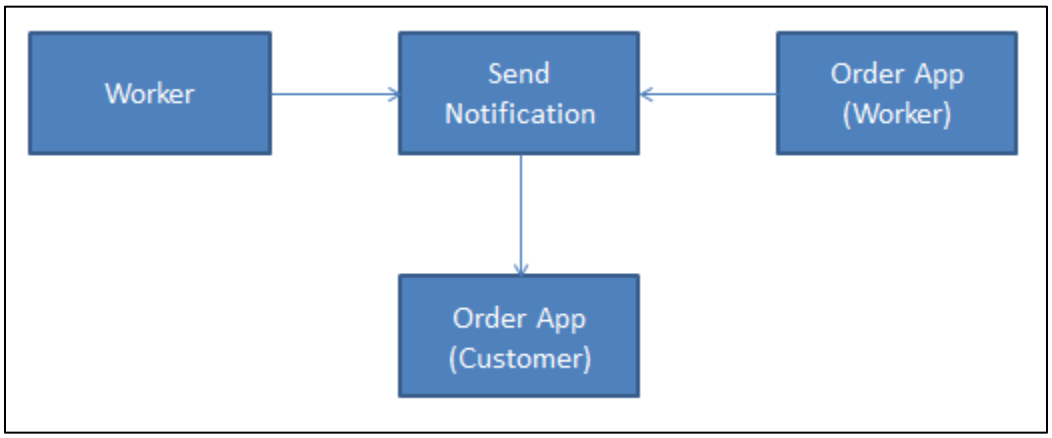


Edit Menu Item – Level 1

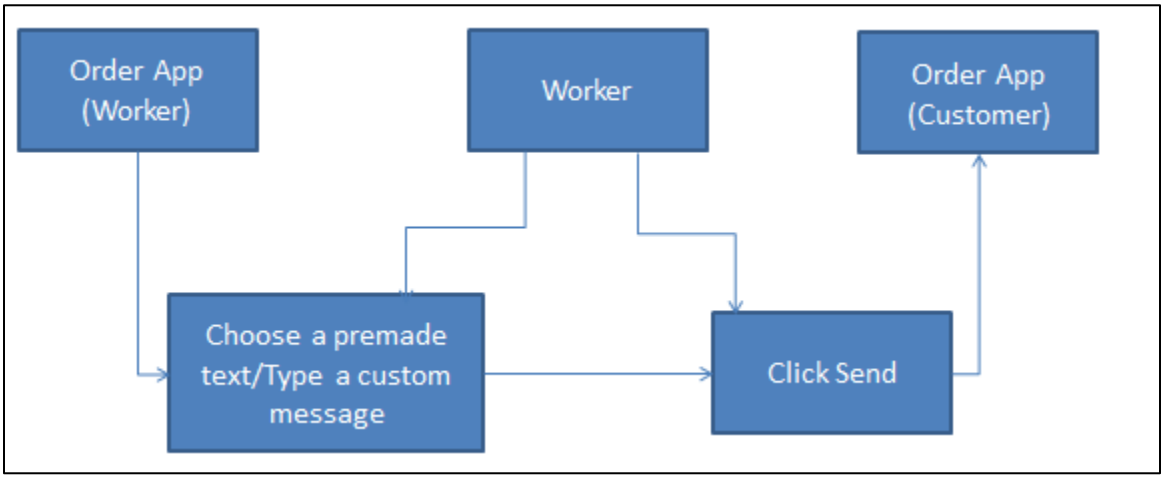




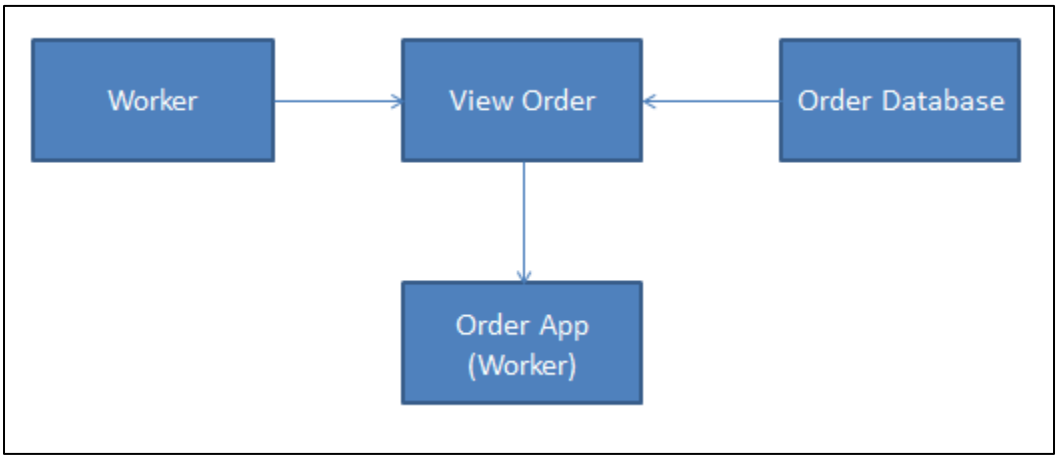
Send Notification – Level 0



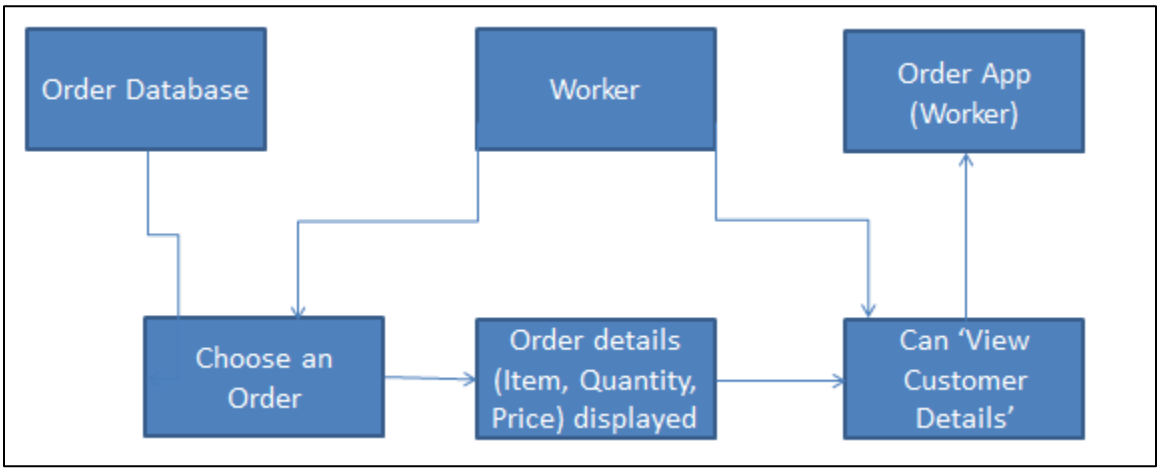
Send Notification – Level 1



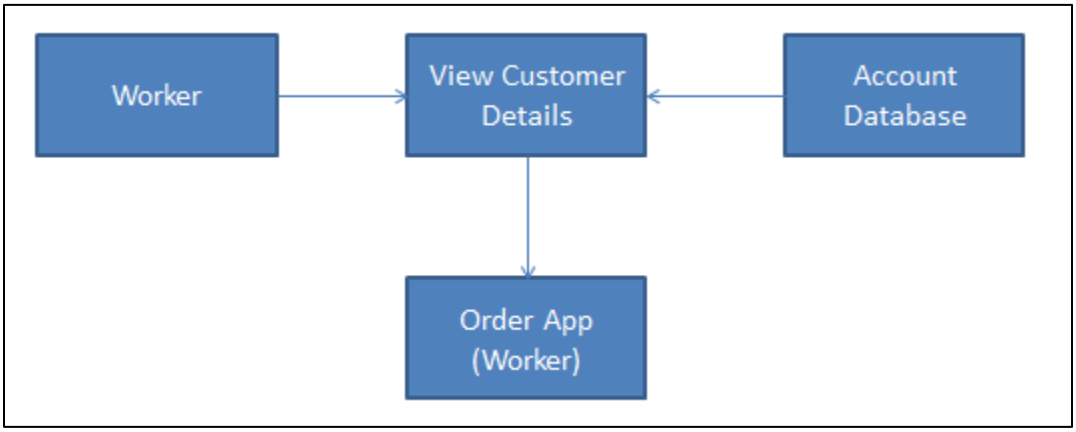
View Order – Level 0



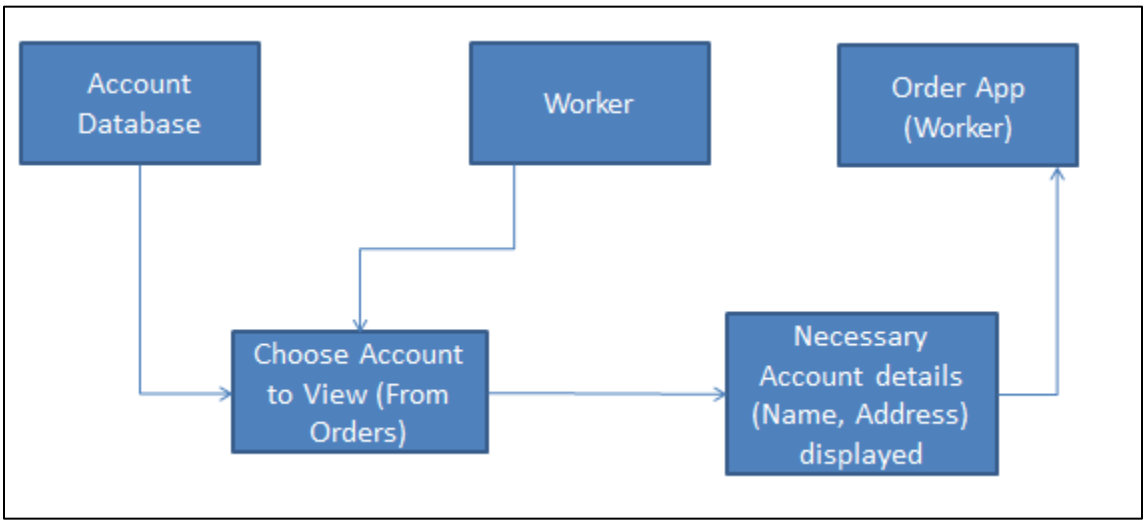
View Order – Level 1



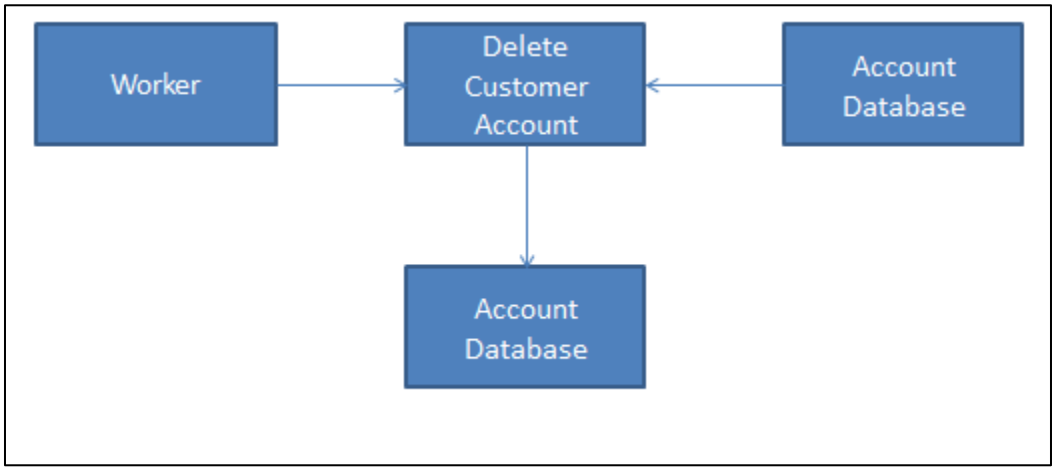
View Customer Details – Level 0



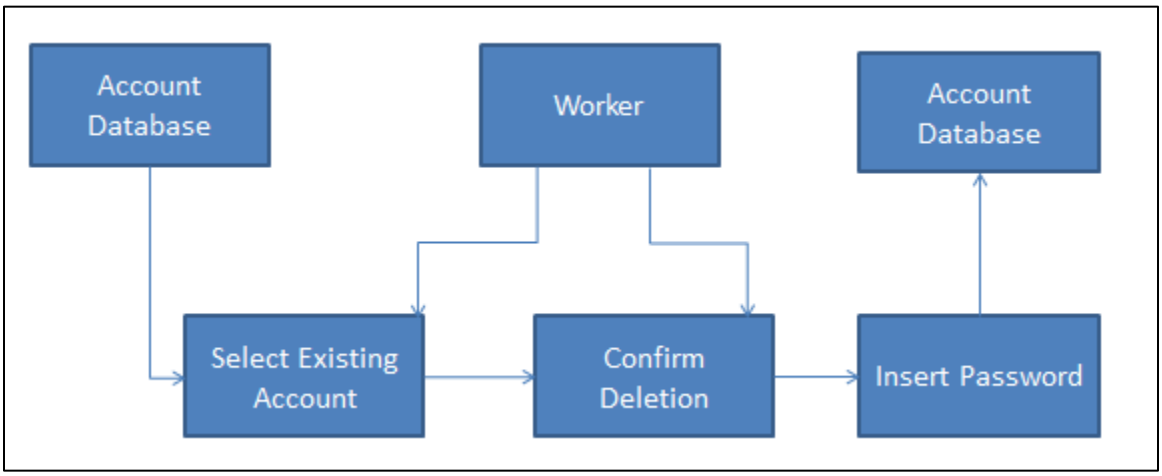
View Customer Details – Level 1



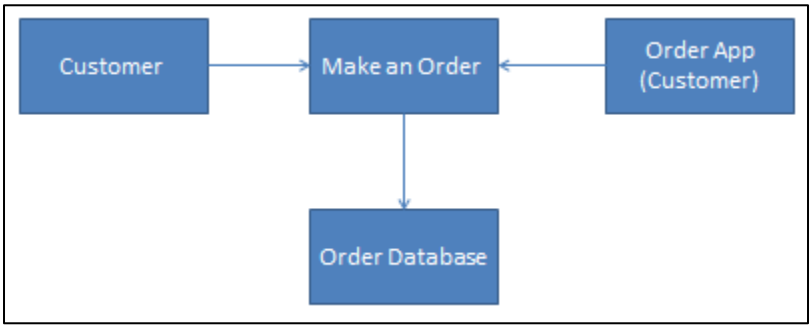
Delete Customer Account – Level 0



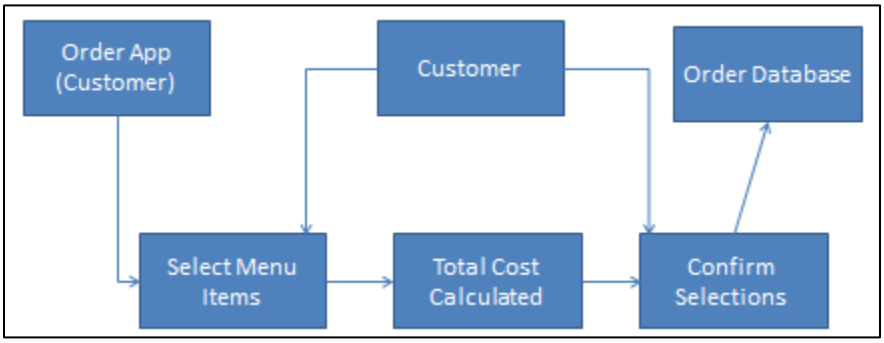
Delete Customer Account – Level 1



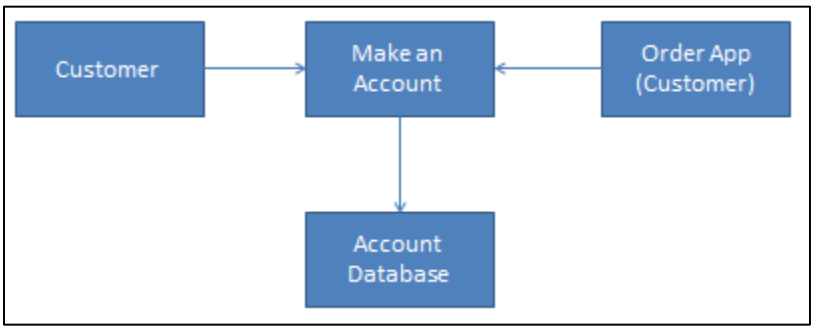
Make an Order – Level 0



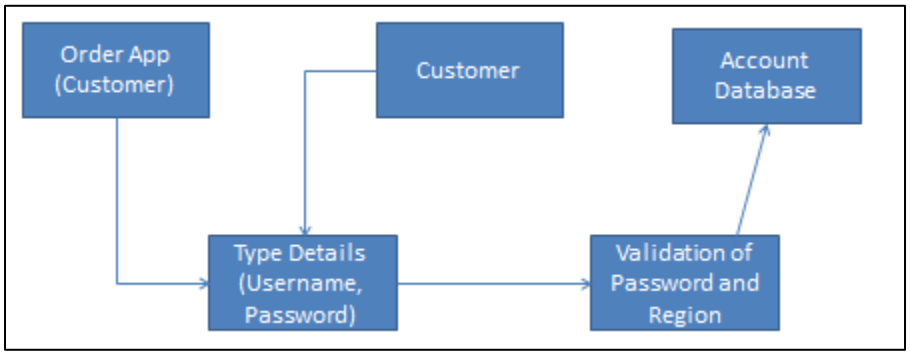
Make an Order – Level 1



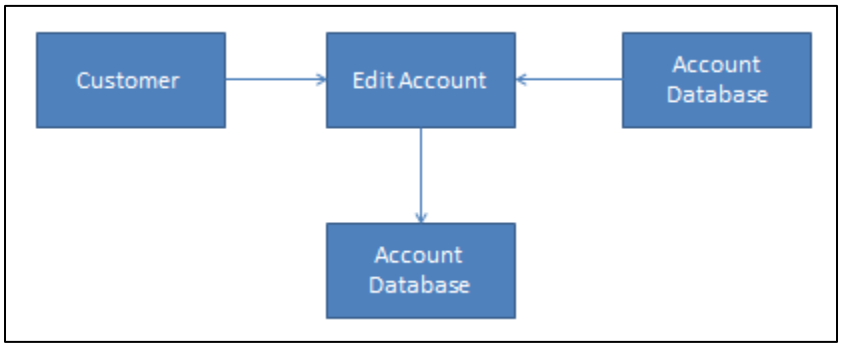
Make an Account – Level 0



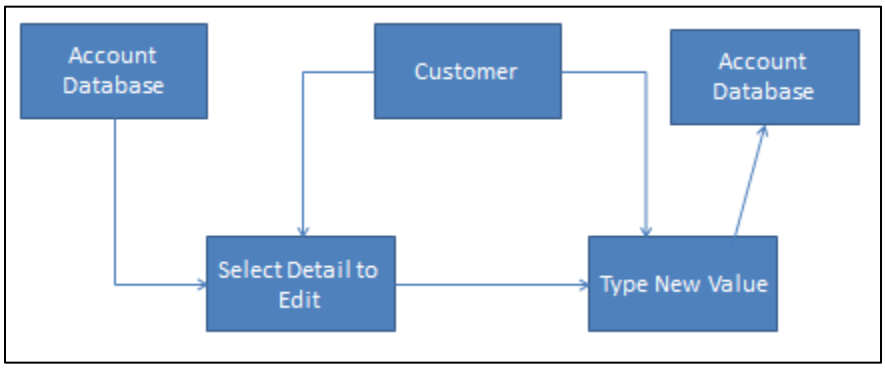
Make an Account – Level 1



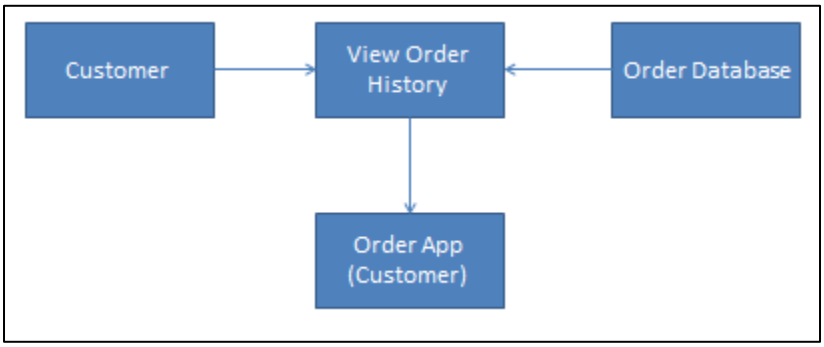
Edit Account – Level 0



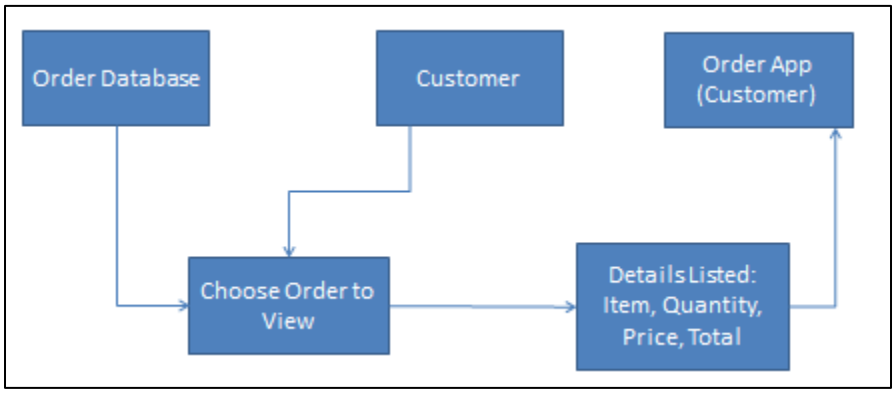
Edit Account – Level 1



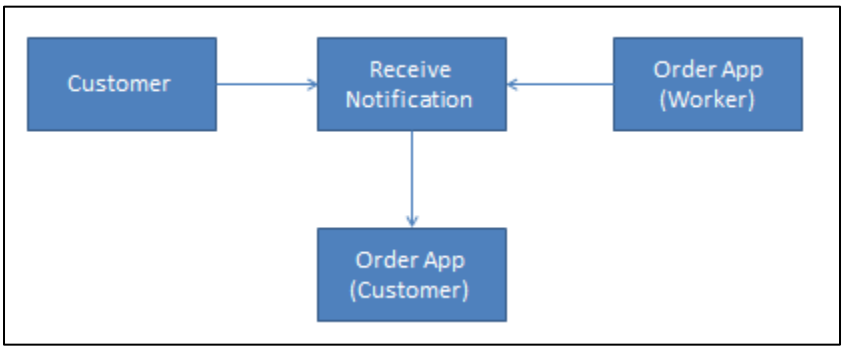
View Order History – Level 0



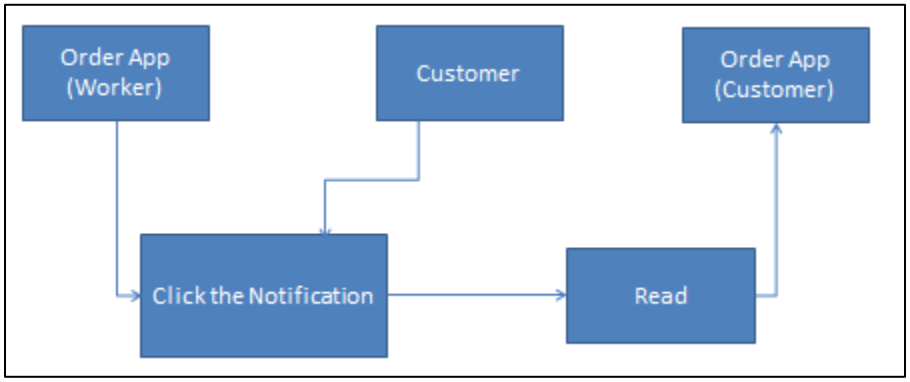
View Order History – Level 1



Receive Notifications – Level 0



Receive Notifications – Level 1



Objectives

1) Functional

Customer App

a) Make an Order

i) Customers selects variety of menu items through searching by category
ii) Inputting the desired quantity of the item chosen
iii) Calculate total price of each item (tax already included with each item) by multiplying the item price with the respective quantity
iv) Calculate the total cost of the order by adding all the total prices
v) Require password to confirm selections
vi) Customers should only be able to order from 9AM until 9PM
vii) Upload order to database, listing item id, quantity and total costs
viii) Get the time of order and make sure it is unique, for an ID

b) Discount Benefits

i) If the customer has ordered at least 15 times, there will be a 15% Discount for every purchase onwards
ii) Calculate this extra benefit in the Send Order part

c) *Make user accounts*
i) Customers must input username and password
ii) Password must be at least 6 characters long (No need variety, e.g. Capital letters, numbers, symbols are not needed)
iii) Customers must input region, region can only be in Ghubrah or Azaiba
iv) Customers must input Street address, House Address
v) Give customers examples of what to type for street and house address
vi) Tell customers why only the two regions is selectable
vii) Store data in database
viii) If user didn't input one or more fields, give a message and don't upload to database
ix) The user's ID will be the user's username in lowercase letters so no other person can have the same username
x) Do not allow if the username is already in use by someone else

d) *Edit user accounts*
i) Customers can change name, password, region, house address and street address
ii) Require customer password to confirm change (If the customer is changing the password then require the old password)
iii) Update data to database
iv) Same rules from Make user accounts is appliedhere

e) *View order history*
i) Download data from database
ii) Customers can select a month and year to view a list of orders
iii) List all orders in the selected month and year as buttons called "Order 1", "Order 2" etc
iv) Click on an order (e.g. click on "Order 1"), and it will display all the menu items ordered, price, quantity, total price, total cost and the date ordered on a seperate page.

f) *Exit app*
i) Closes app completely

g) *Receive Notifications*
i) Get notifications from workers, status of their order

ii) Should appear on phone even if the app is not directly opened and is running in the background

Worker App

h) Edit Menu Item

i) Require the edit menu password to access this feature (So workers can't randomly delete, add or change menu items for no reason)
ii) Can change item name or price
iii) Can delete an item completely
iv) Can add new items, typing the item name, price and category

i) View Menu

i) Workers can select a category to view the items under that category
ii) Download from database
iii) Item name and price will be listed

j) View Order

i) Download order from database
ii) Display under awaiting orders
iii) List the orders in chronological order of time ordered
iv) Display items, quantity and total cost (no need each individual cost)
v) Option to view the customer's details (name and addresses)
vi) Option to send notifications
vii) Option to remove the order when it is completed, however a password input is required from the customer who made the order

k) Send Notifications

i) Select options (e.g. "Food is being made", or "On the way") and send
ii) Can also type send custom notification (e.g. "Arrived at the front of your apartment" or "Arrived at the back of your apartment")
iii) Can also send notifications to everyone (not just customer who ordered) for things like announcements (e.g. "NEW MENU ITEM! Fish Soup for 1 OMR!")
iv) Require the send notifications password to do so (to avoid workers from sending random notifications to users)

l) View Customer Details

i) Can check name, address of those that ordered

m) Delete Customer Details

i) Can delete the account completely
ii) Ask for confirmation prior to deletion

iii) Require Delete customer Password to do so (to avoid workers from deleting accounts for no reason)

n) Exit app
   i) Closes app completely

2) HCI

a) User Friendly
   i) Easy to understand
   ii) Clear titles
   iii) Clear instructions

b) Buttons
   i) Buttons are clearly titled as per their functions

c) Option Limitations
   i) Radio options to limit user input to certain things
   ii) Drop down lists options to limit user input to certain things with a greater range of options

d) Page Navigation
   i) Arrow buttons for going back a page
   ii) Clear titles for advancing pages

3) Data Storage & Security

a) Cloud Storage
   i) Using Firebase for the cloud storage

b) Database
   i) Firebase Database is a JSON Tree Database
   ii) Have 4 main branches for Menu Items, Accounts and Orders Made and Awaiting Orders

c) Require an Account with Username and Password to log in
   i) Account details only known to the account creator
   ii) Creating an account avoids having to type the users' address over and over again

d) Password Protection
   i) Password will be hashed one-way for maximum security and the hashed value will be stored in the database
   ii) Inputted password (logging in, confirming order) will be hashed and compared to the one stored in the database for confirmation

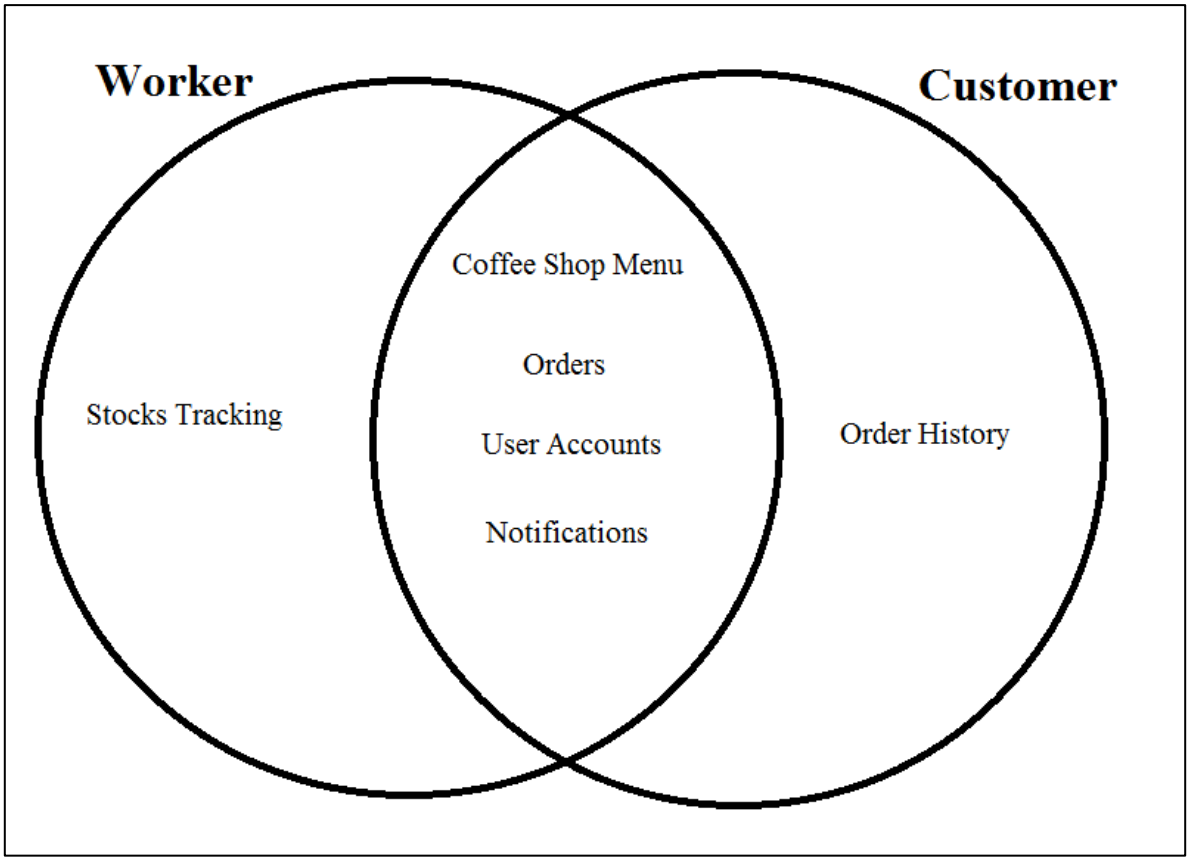
4) Environmental

a) Customer Location

i) Only in Azaiba and Ghubrah (too far for the coffee shop to travel for other regions)
ii) Others cannot order for home delivery
iii) During sign-up there will be a Region detail to verify if they can create an account or not

Scoping

1) *Stocks Tracking*
   a) Stocks will not be tracked as it does not belong in the application as it is not of use for both worker AND customer



Feature	Details
Stocks Tracking	This will help the worker in knowing what ingredients to buy if it is low or what is in full stock. This information is not useful for customers whatsoever
Menu	Workers can check and edit the menu,

	while the customers can view the menu to decide what they are going to order.
Orders	Customers make an order for the worker to check and make.
User Accounts	The address of the user is saved, hence the worker would already know where to deliver, and the user would not need to state their address over and over again.
Notifications	This allows the worker to communicate with the customer.
Order History	Only users can use this feature, to check what they ordered previously, however this can be further developed for the workers to use as well.

2) Workers Editing Customer Details
   a) Workers won't be able to edit customer details as the customer can do that themselves. This also avoids workers from editing customer details for no reason.
3) E-mail verification and recovery not required
   a) This isn't required as there isn't highly sensitive data such as credit card details
   b) Should a user forget their username or password they will simply have to make a new one

Appendix

Date	Who	Draft Feedback
23/10/16	Stefan	1) Check if the address the user input is in the Geographical Limit. Do this by automatically checking the address on Google Maps as a Validation when the user is signing up. - I have decided to only allow the two options of Azaiba and Ghubrah to be selected during the sign up. This will show users that only those who are in these regions can sign up. 2) If a user is not sure about how to do some things on the app or have queries about something how would they have to act? Call? -The premise of the app is fairly simple and most people would easily understand how to function the app. However for the low percentage that somehow don't will simply have to ask help from others or go to the coffee shop.

		3) How to deal with troll orders? - In the event someone else accesses your account from your phone, when you are about to send an order, it will ask you your password to verify the order. This is the only way for me to deal with a "troll" order.
28/10/16	Leandro	1) Is the coffee shop open 24 hours a day? If not then can the user make an order when the shop is closed? - I have made it so the users will only be able to make an order during the time the coffee shop is opened. 2) What happens if the database has exceeded its storage limits? - Apart from the obvious choice of paying monthly to get a bigger storage size, I have made it so that workers can delete user accounts in order to free up space.

https://firebase.google.com

https://developer.android.com

Documented Design

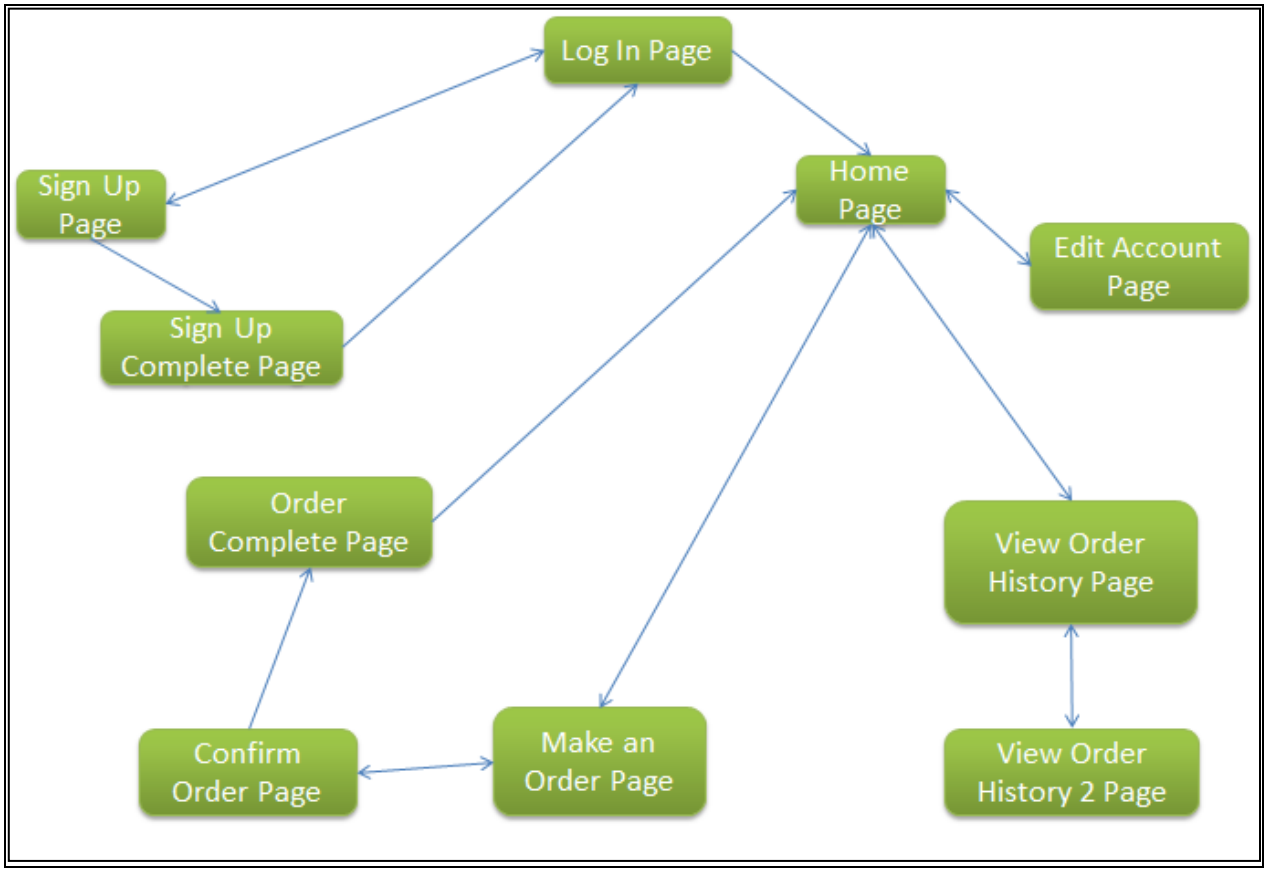
Overview

The aim of this project is to create an app to allow customers to order food from the coffee shop without requiring to call the coffee shop and also to allow the coffee shop to handle orders more efficiently. The customers should be able to; create an account that will have their details stored, including personal and order details, create an order, edit their personal details and view their order history. The coffee shop workers should be able to; view an order, send notifications to a single user or every single user, view and edit their menu, and delete old accounts to save space.

Description of Modular Structure of System

I have created navigation diagrams for the pages in my app. In addition to that there is a table listing the activities the user can perfrom on each page. In order for users to navigate to other pages they must click a button, either one saying the command such as "Log In", or a simple image button of a back arrow. Every page has this activity excluding the Home page, Log In page and the transition pages.

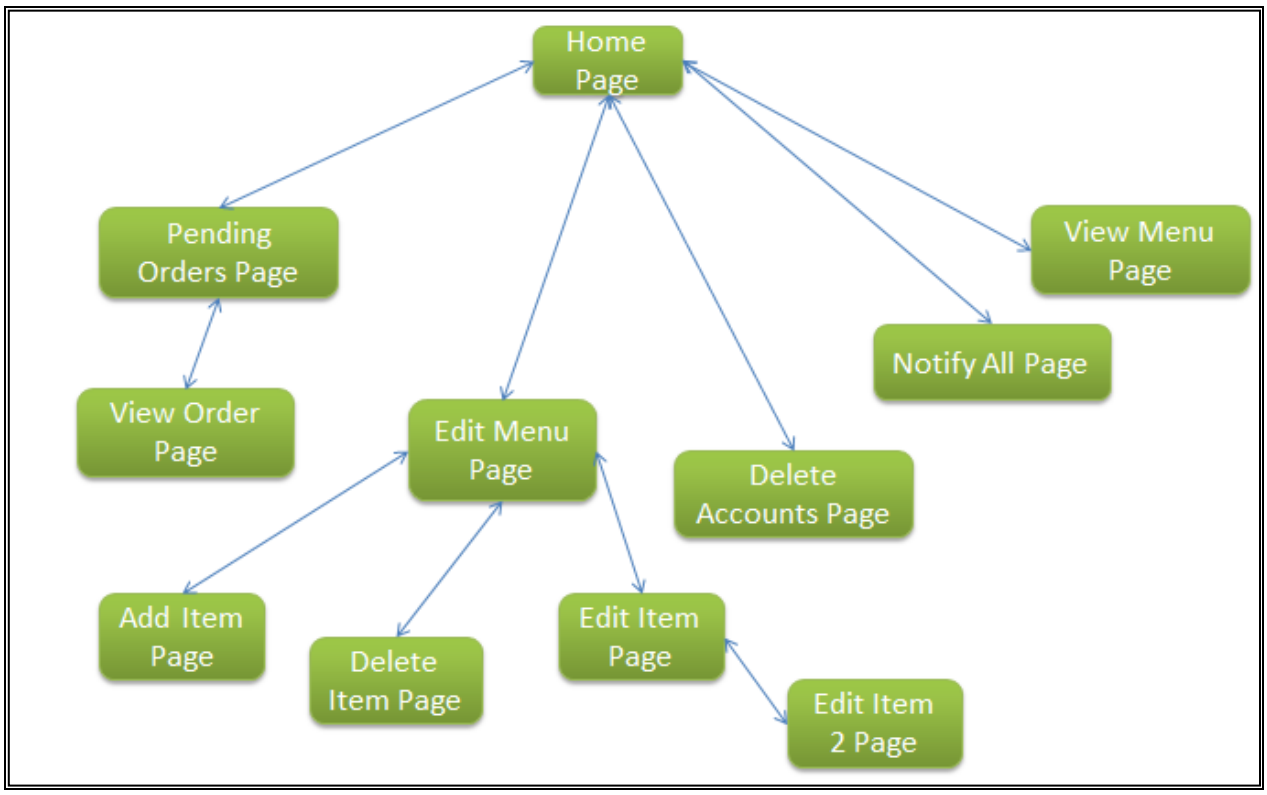
Customer App



Page Name	User Activities	Navigation
Log In	Input username and password to access app functions	• Sign Up • Home
Sign Up	Create an account to use the app by inputting customer details Typing in name, username, password, street and house address Selecting radio button option for region	• Log In • Sign Up Complete
Sign Up Complete	N/A	• Log In
Home	Exit the app completely	• Make an Order

		• Edit Account • View Order History
Make an Order	Select food item category to display for selection Select desired food items with quantity for ordering	• Home • Confirm Order
Confirm Order	Viewing calculations of total price and cost Inputting password to confirm the order	• Make an Order • Order Complete
Order Complete	N/A	• Home
Edit Account	Change details of account Typing in name, password, street and house address Selecting radio button options for region Inputting password to confirm change	• Home
View Order History	Selecting month and year to produce a list of orders made in that time frame Select order for viewing	• Home • View Order History 2
View Order History 2	View order details of the selected order	• View Order History

Worker's App



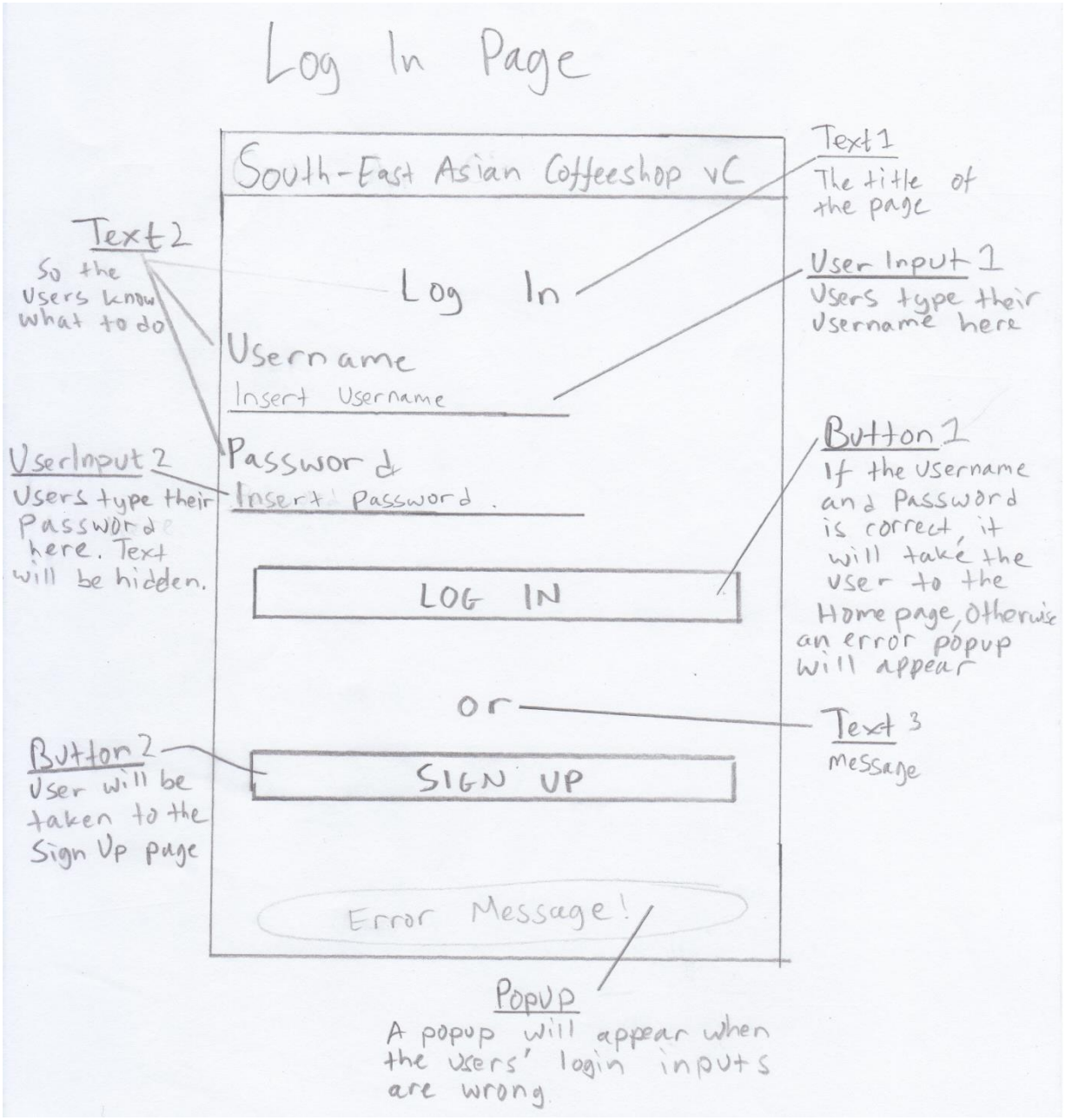
Page Name	User Activities	Navigation
Home	Exit the app completely View number of pending orders Edit Menu, Notify All and Delete Accounts will require the user to input a password to access the pages	• Pending Order • View Menu • Edit Menu • Notify All • Delete Accounts
Pending Order	View who made orders	• View Order • Home
View Order	View the details of the user who made the order Send notification to the user that made the order Input password to delete order	• Pending Order

	for completion	
View Menu	Select item category View items	• Home
Edit Menu	Select what type of editing the menu	• Home • Add Item • Edit Item • Delete Item
Add Item	Type item name Type item price Select item category	• Edit Menu
Edit Item	Select item category Select item for editing	• Edit Menu • Edit Item 2
Edit Item 2	Type item name Type item price	• Edit Item
Delete Item	Select item category Select item for deletion Confirm by pressing button	• Edit Menu
Notify All	Type message	• Home

UI Diagrams

The following diagrams consist of how each page in the app will look like. I will also describe the characteristics of each item. That being said, all text will have the font size 'sans-serif', font colour black (#000000) and the background will be white (#FFFFFF).

These are the UI diagrams for the Customer App:



Text1 – Font size 24sp, centralised

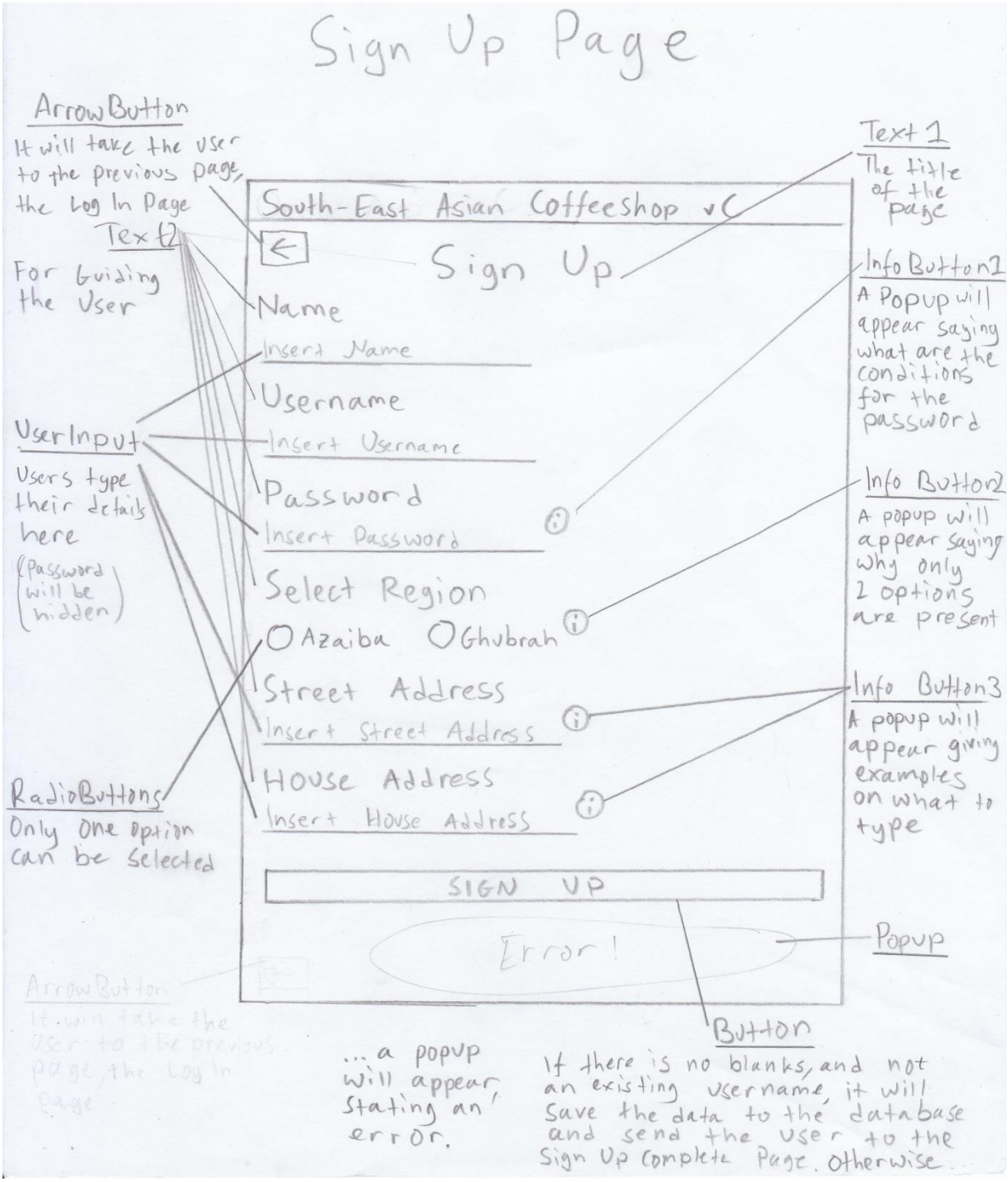
Text2 – Font Size 18sp, left aligned

Text3 – Font Size 18sp, centralised

UserInput1 – Font Size 14sp, left aligned, default text is hinted

UserInput2 – Font Size 14sp, left aligned, default text is hinted, text type 'password' (to make it hidden)

Button1 & 2 – Font Size 14sp, centralised

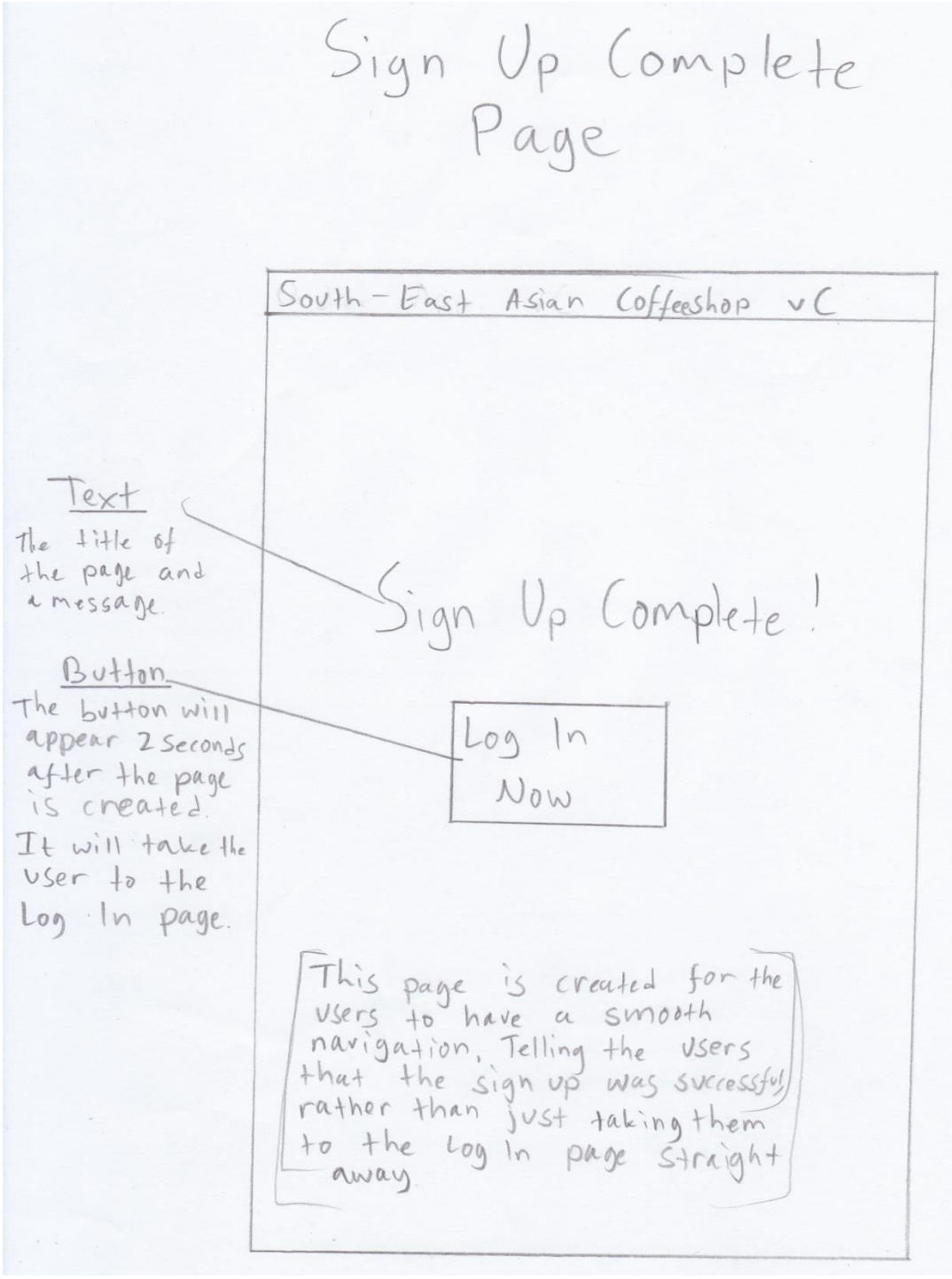


Text1 – Font size 24sp, centralized

Text2 – Font size 18sp, left aligned

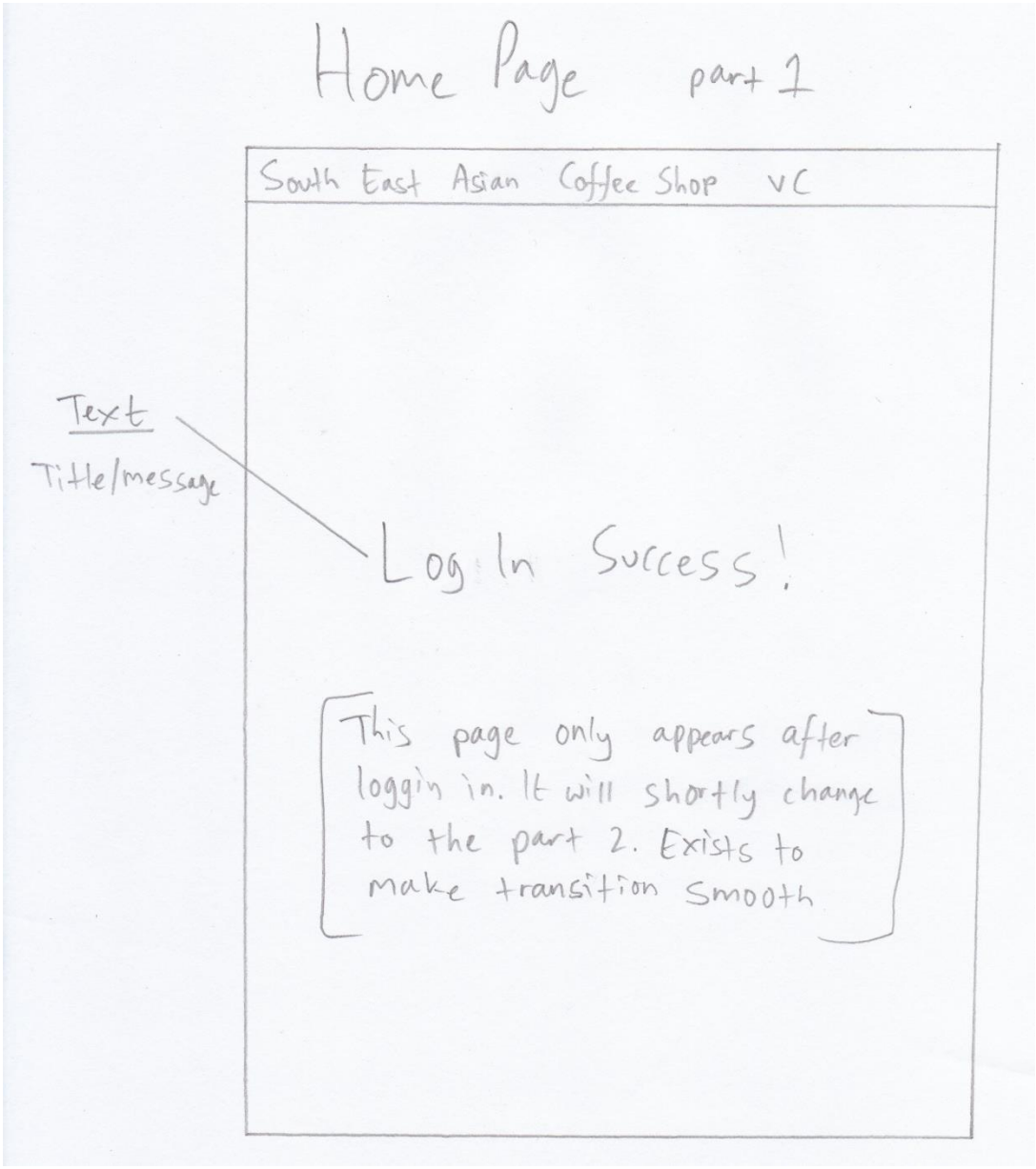
UserInput – Font size 14sp, left aligned, default text hinted, for the password input make text type

'password'
RadioButtons – Font size 14sp, left aligned, Set the radio group's orientation as 'horizontal'
Button – Font size 14sp, centralised



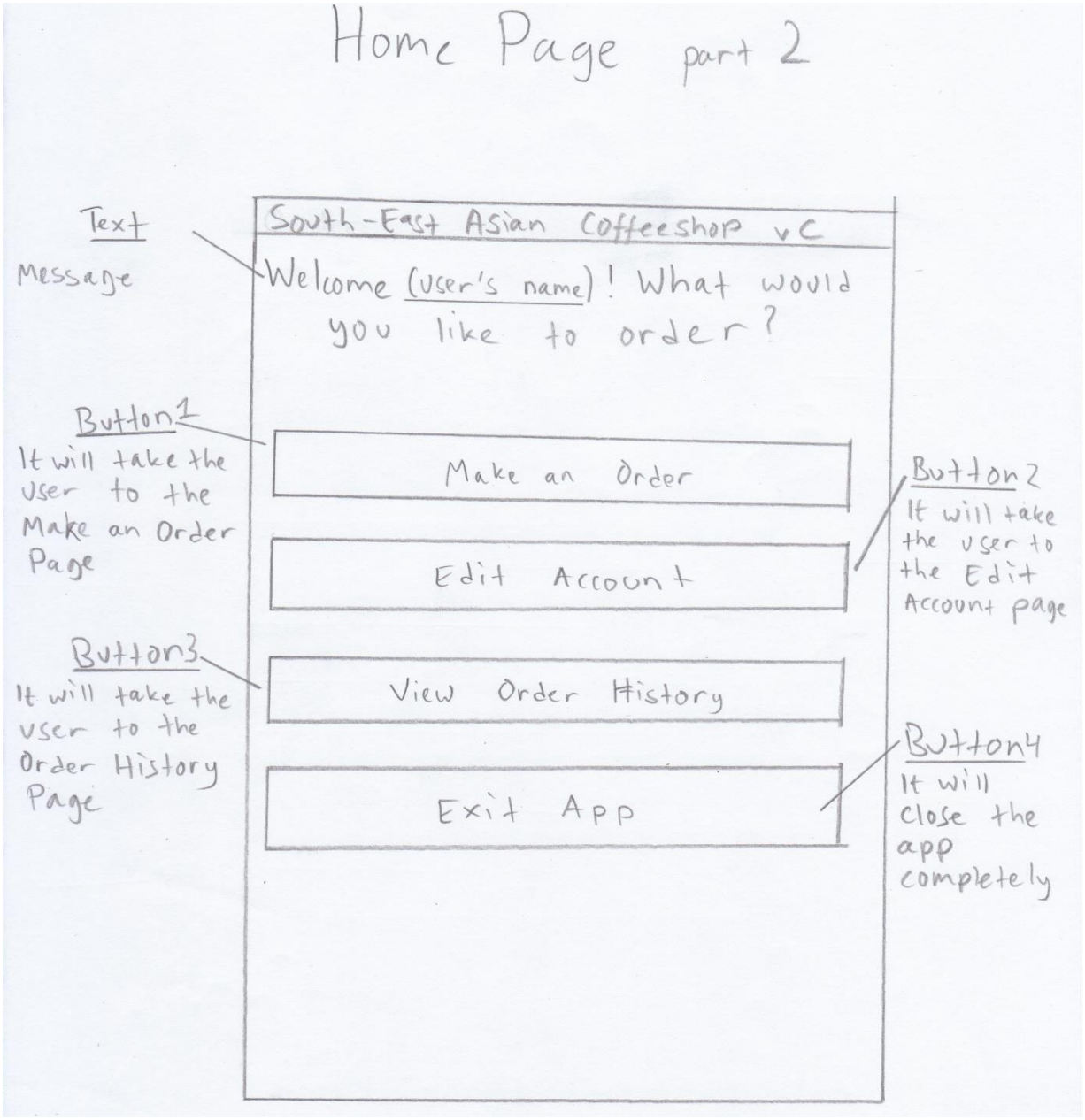
Text – Font size 24sp, centralized

Button – Font size 14sp, centralised

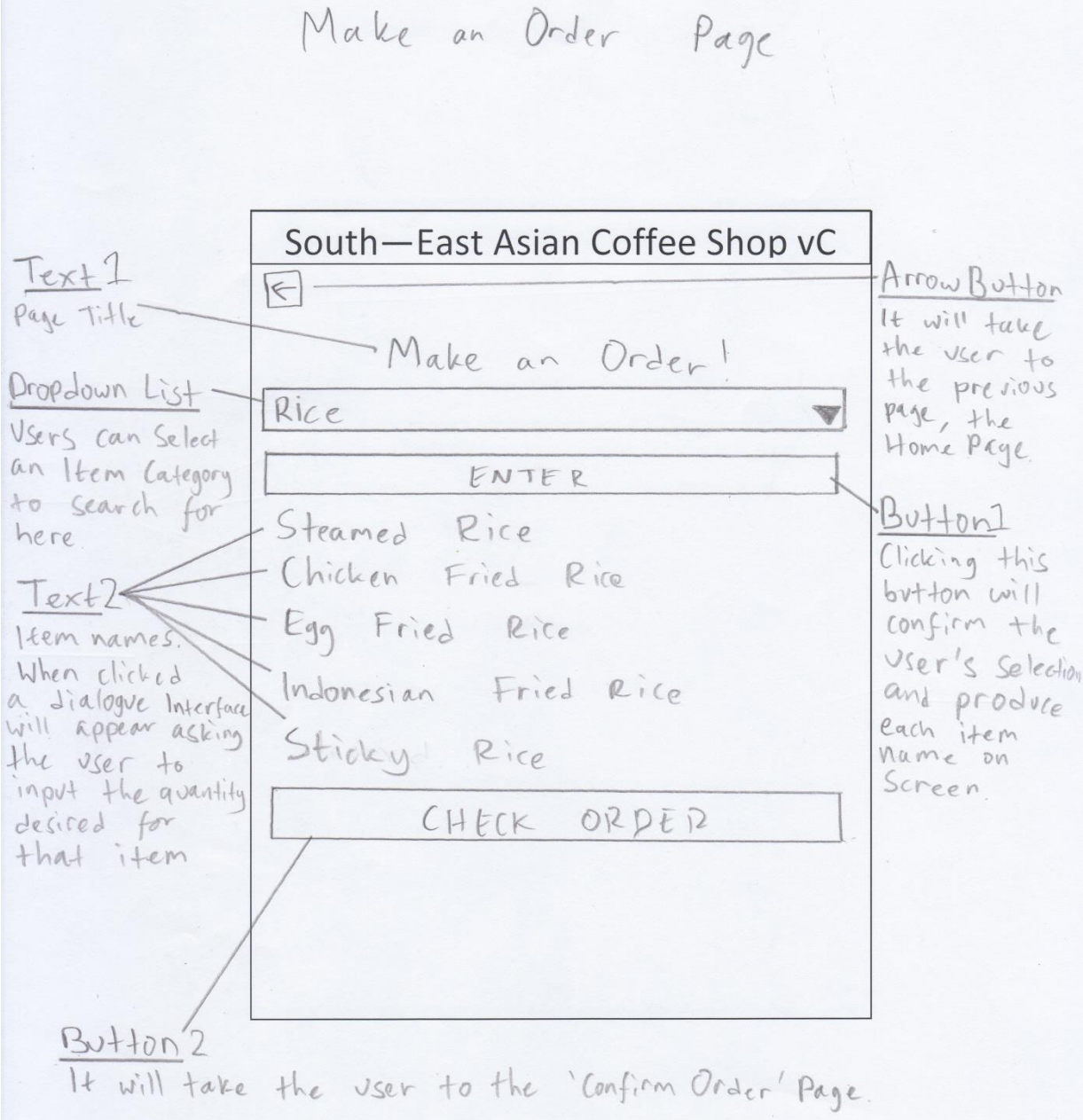


Text – Font size 24sp, centralized

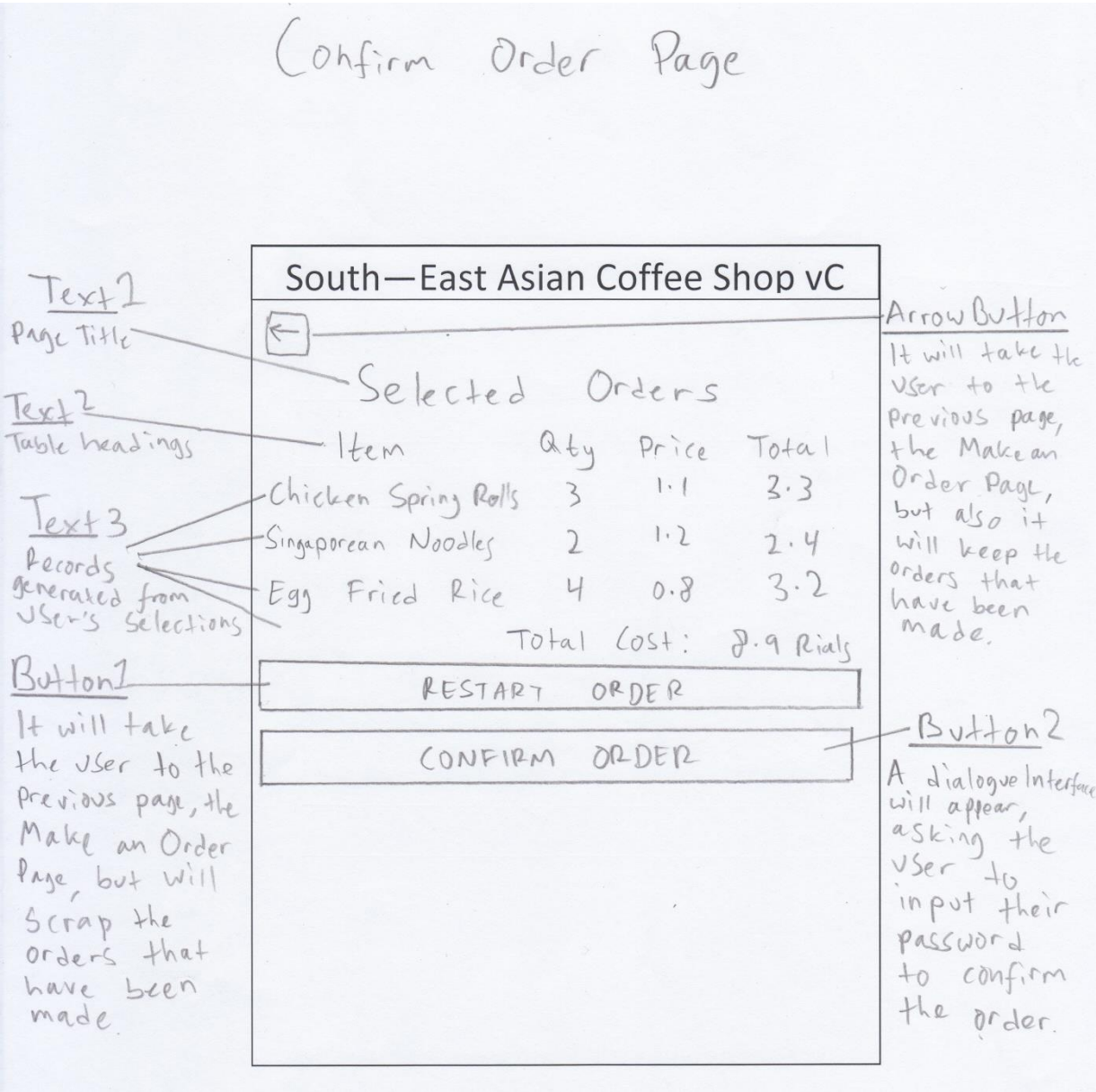
If users change page to the Home page from any other pages that is not the log in page, this will not appear.



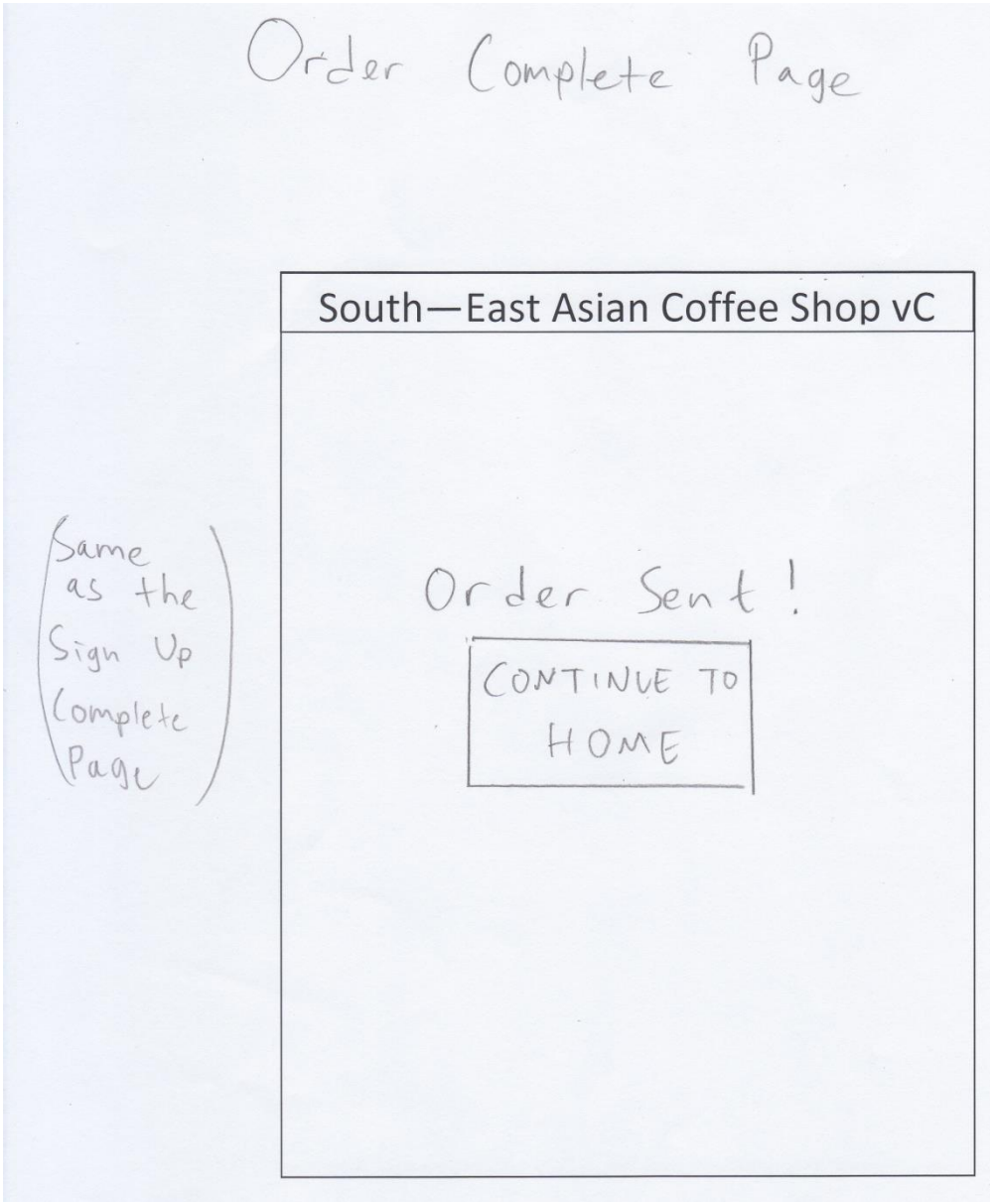
Text – Font size 24sp, centralized
Buttons – Font size 24sp, centralised



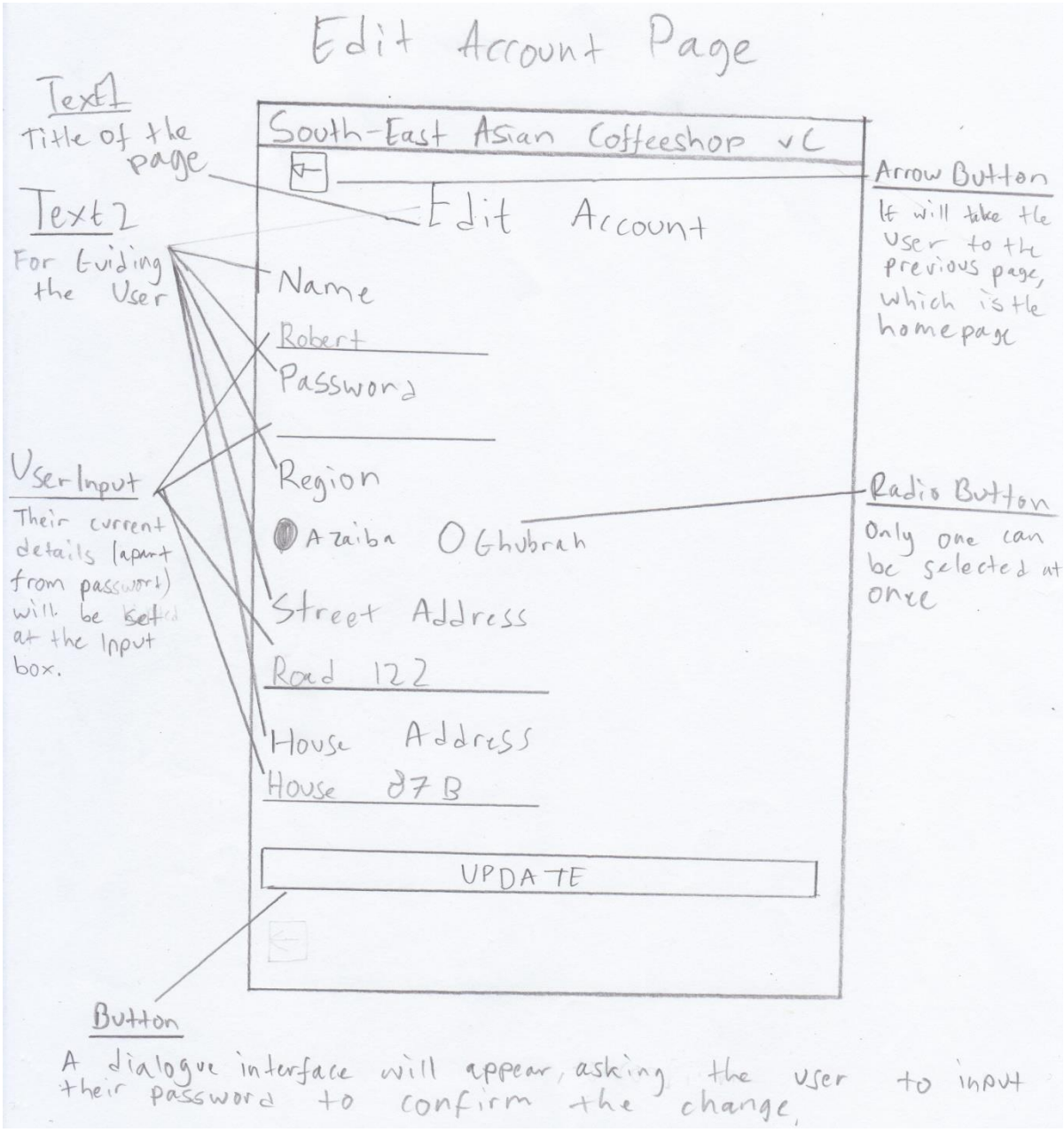
Text1 – Font size 24sp, centralized
Text2 – Font size 18sp, left aligned
Button1 – Font size 14sp, centralized
Button2 – Font size 14sp, centralized
Should there be an error a popup will appear



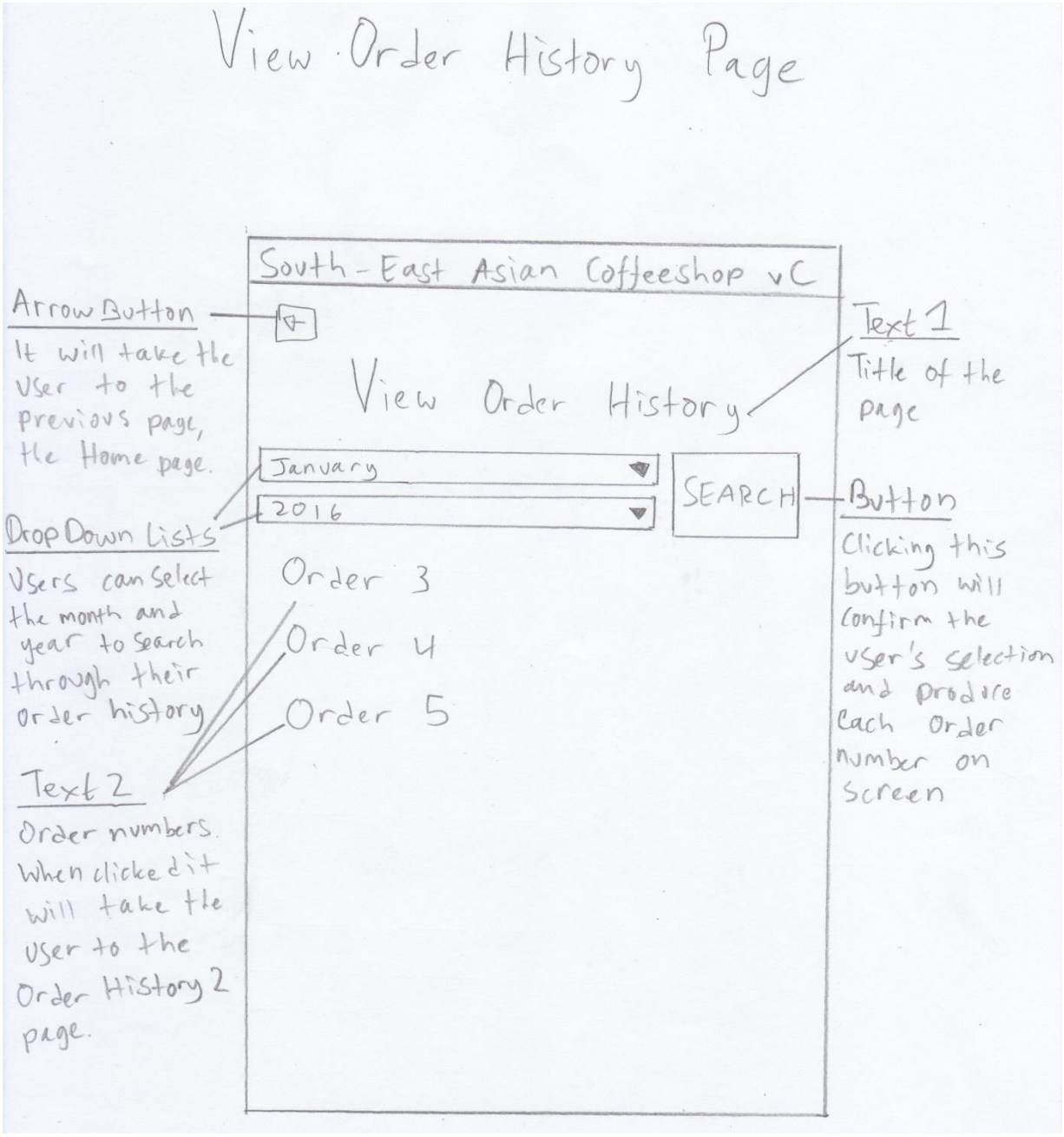
Text1 – Font size 24sp, centralized
Text2 – Font size 18sp, centralised
Text3 – Font size 14sp, centralized (except the names of item, they are left aligned)
Table – Four columns, the last row all four columns are merged into one
Button1 – Font size 14sp, centralized, this button will empty the order list
Button2 – Font size 14sp, centralized



Exactly the same as the sign up page!



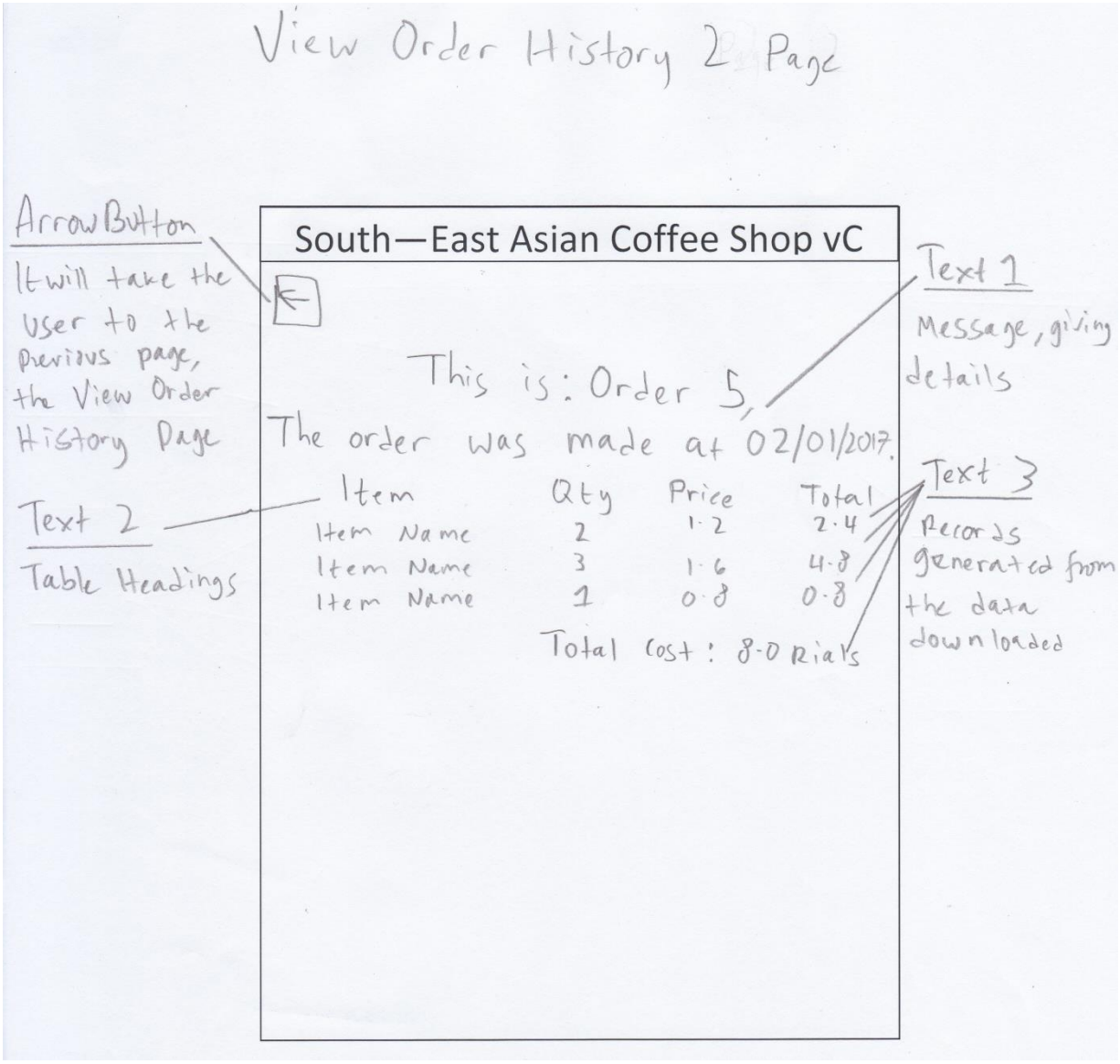
Text1 – Font size 24sp, centralized
Text2 – Font size 18sp, left aligned
UserInput – Font size 14sp, left aligned, set the text as normal (except for password, leave it blank)
RadioButton – Font size 14sp, left aligned, set the radio group orientation to 'horizontal'
Button – Font size 14sp, centralized
Should there be an error a popup will appear



Text1 – Font size 24sp, centralized

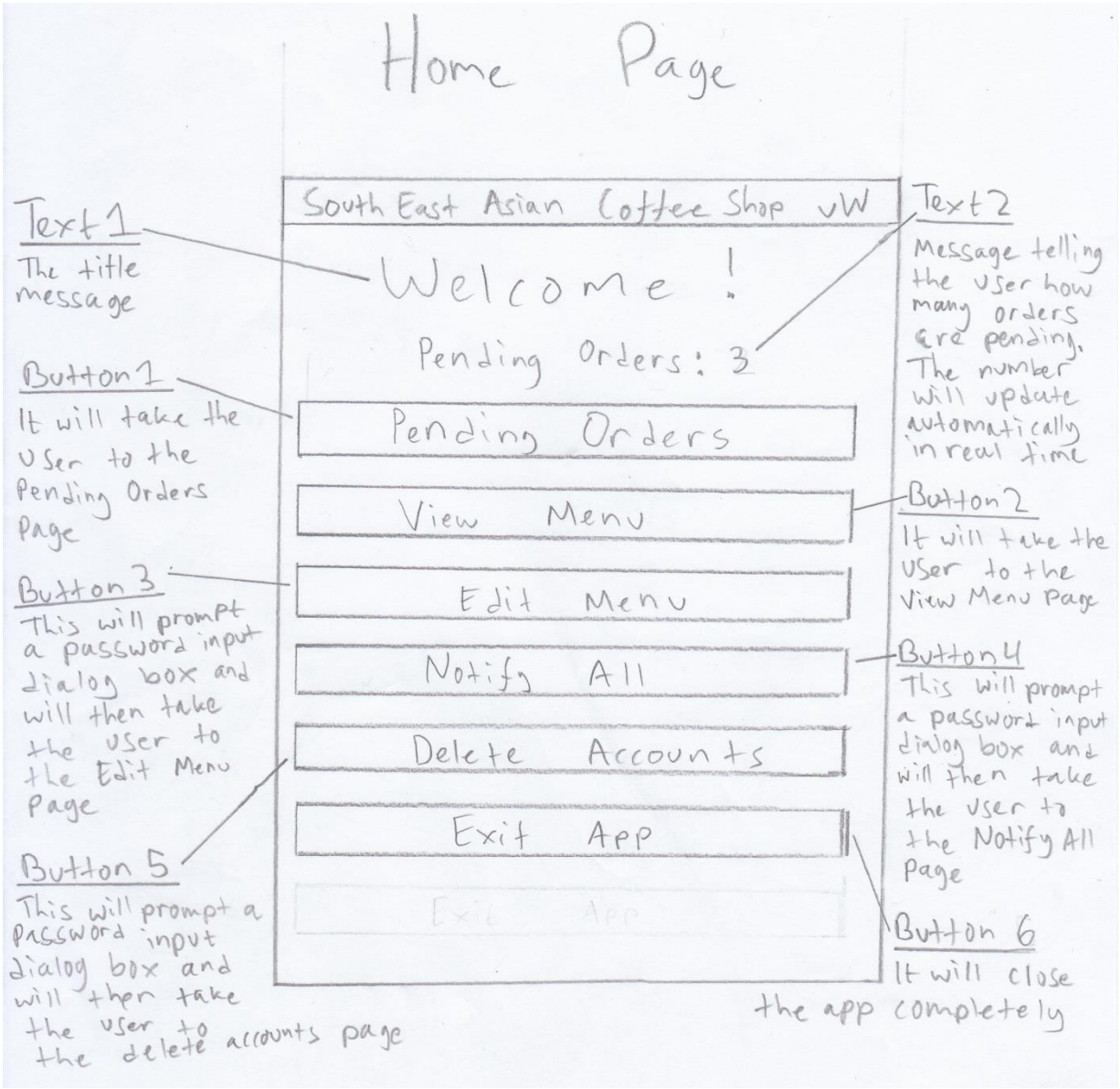
Text2 – Font size 23sp, left aligned, clickable

Button – Font size 14sp

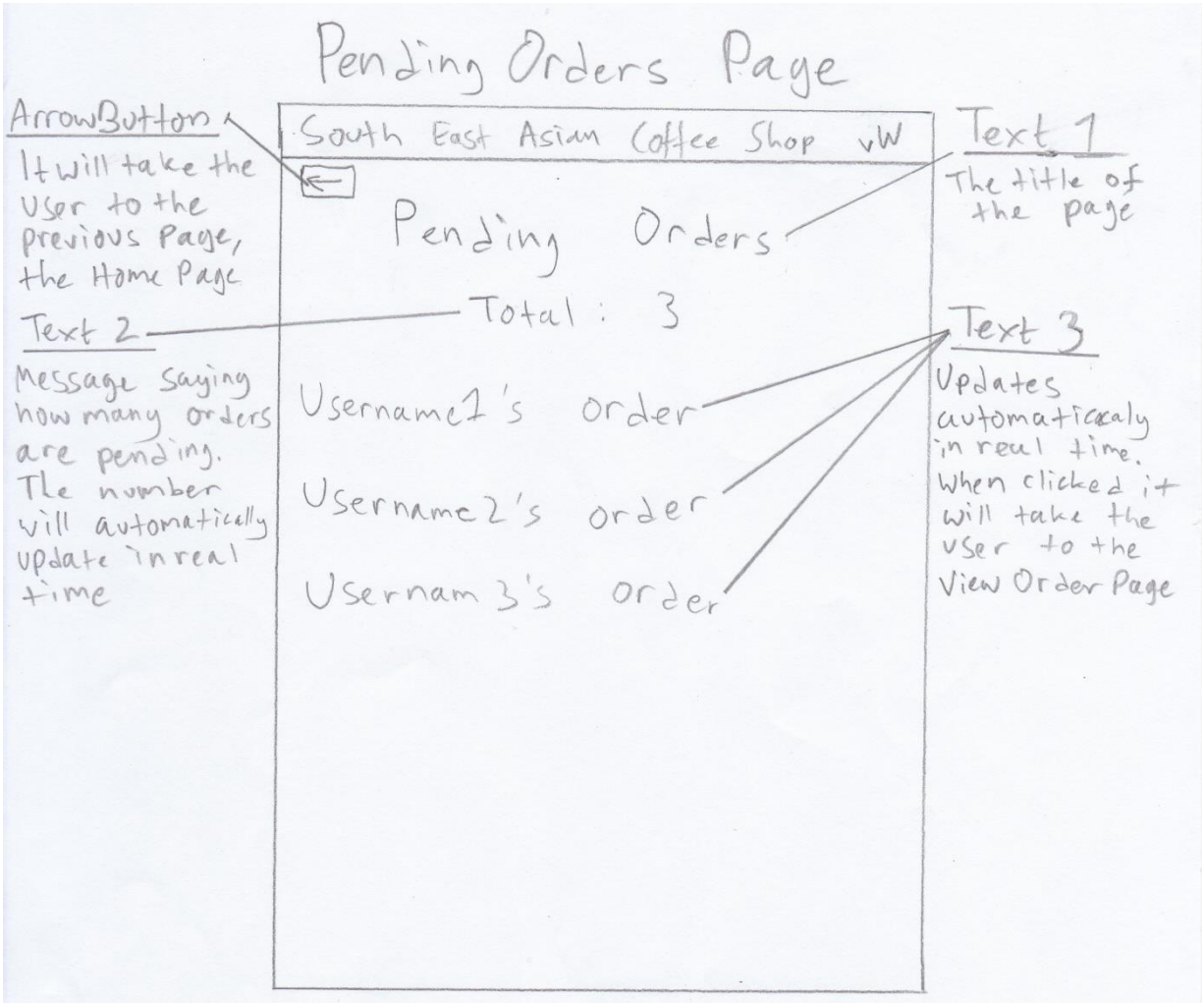


Text1 – Font size 24sp, centralized
Text2 – Font size 18sp, centralized
Text3 – Font size 14sp, centralized (except the item name is left aligned and the total cost is right aligned)
Table – Four columns, the last row all four columns are merged into one

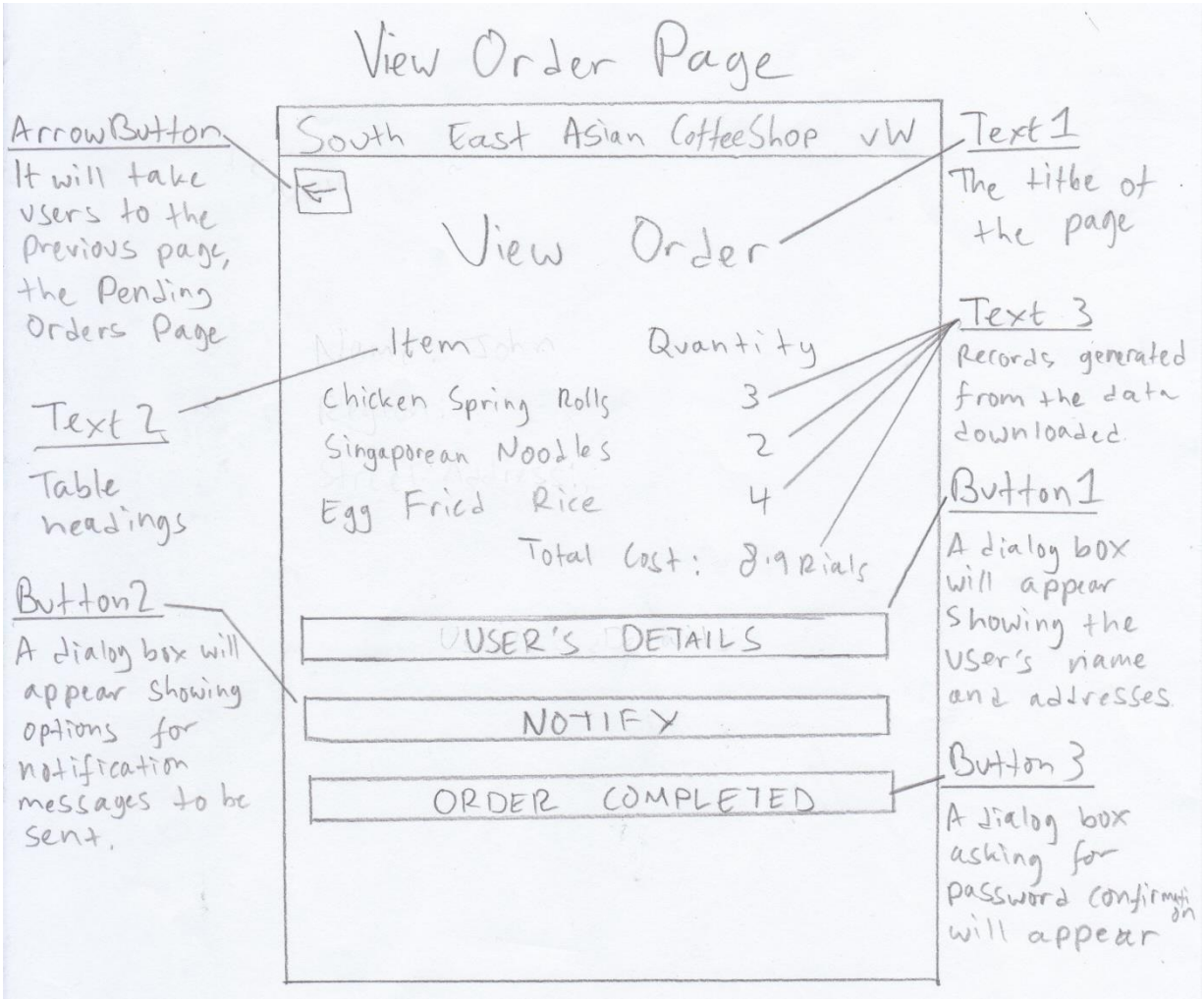
Here are the UI diagrams for the Worker App:



Text1 – Font size 24sp, centralized
Text2 – Font size 18sp, centralized
Buttons – Font size 14sp, centralized
Should there be an error a popup will appear



Text1 – Font size 24sp, centralized
Text2 – Font size 18sp, centralized
Text3 – Font size 23sp, left alignment



Text1 – Font size 24sp, centralized

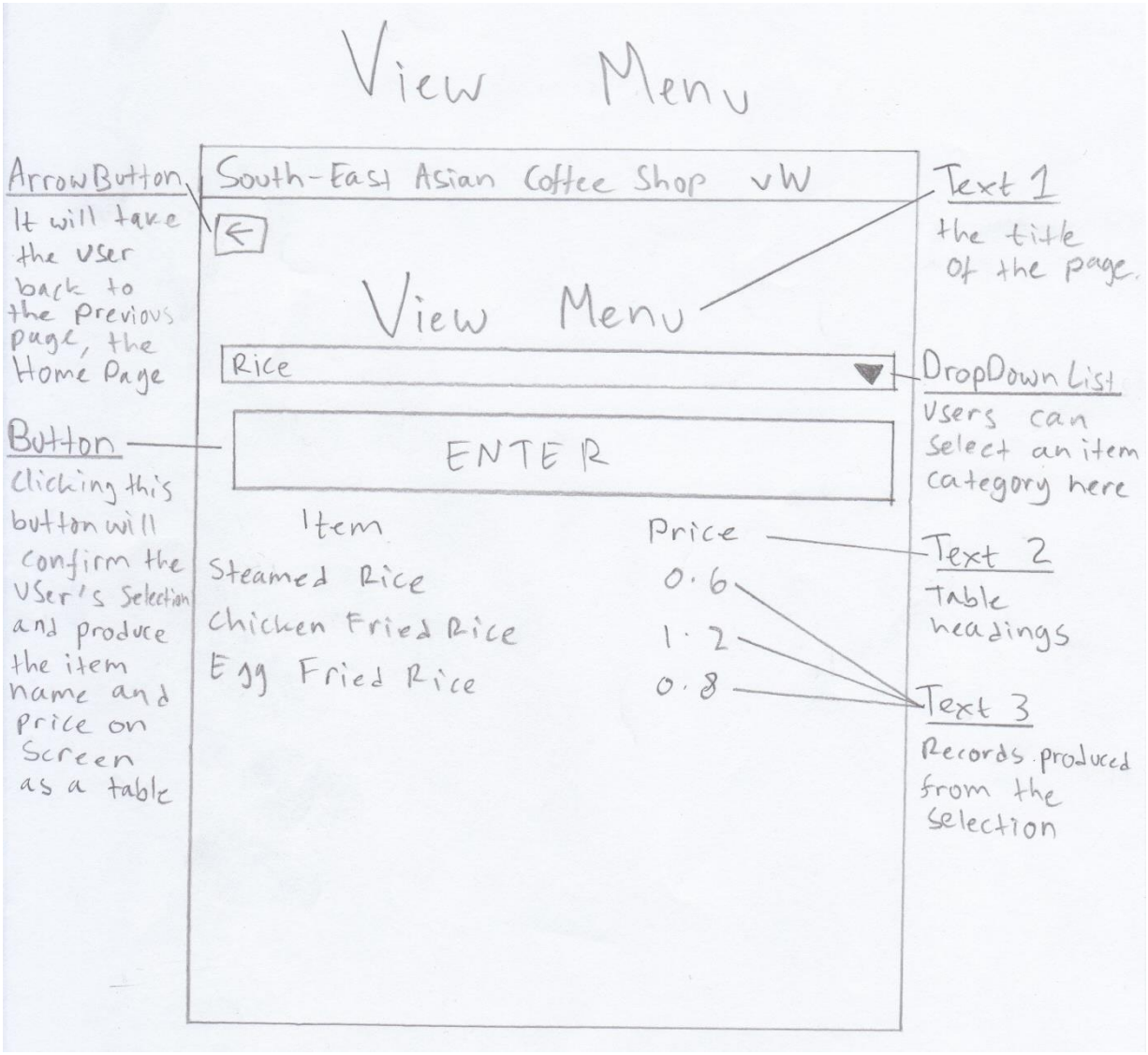
Text2 – Font size 18sp, centralized

Text3 – Font size 14sp, item name left aligned, quantity centralized, and total cost right aligned

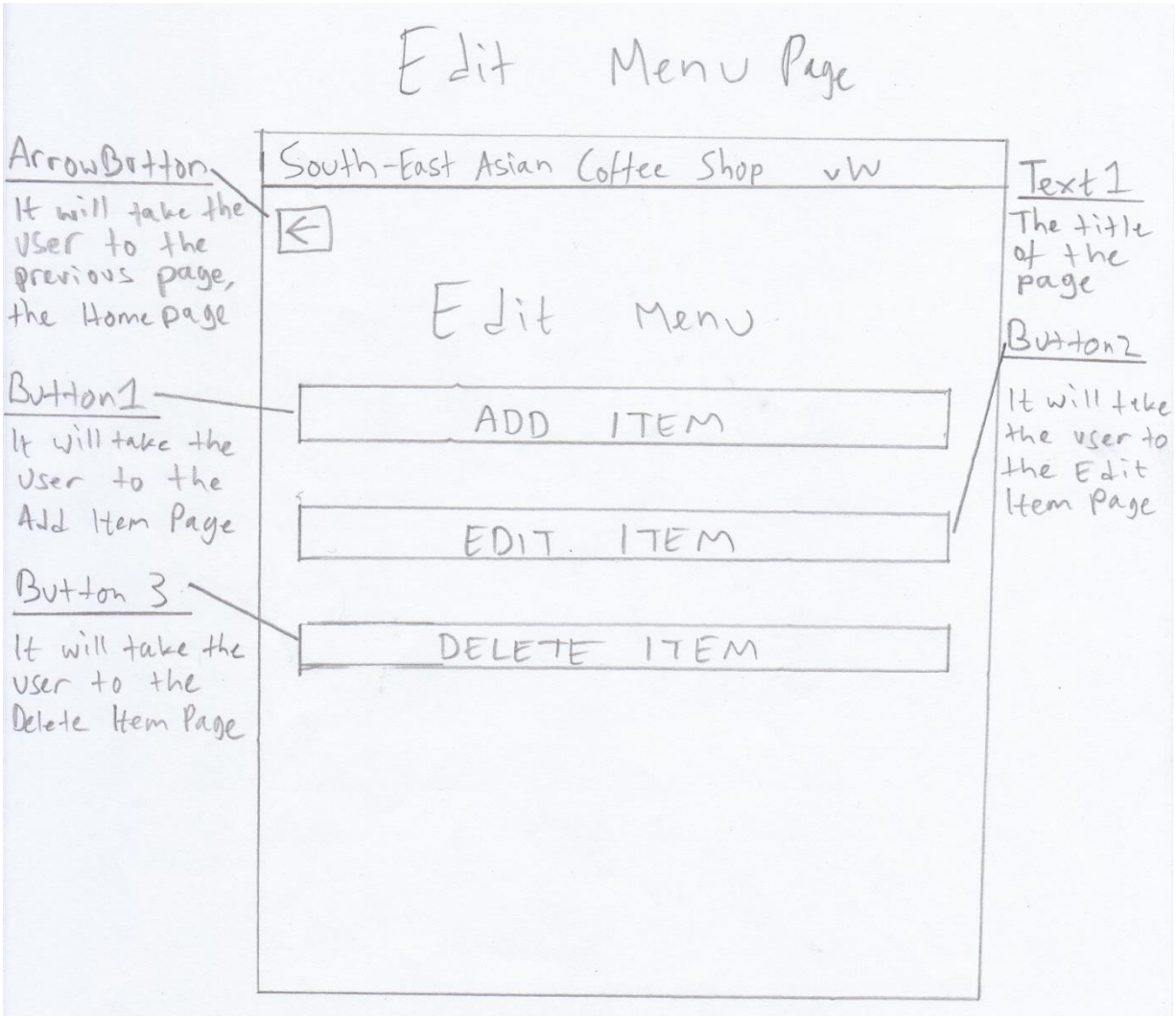
Table – Two columns, last row the columns are merged into one

Buttons – Font size 14sp, centralized

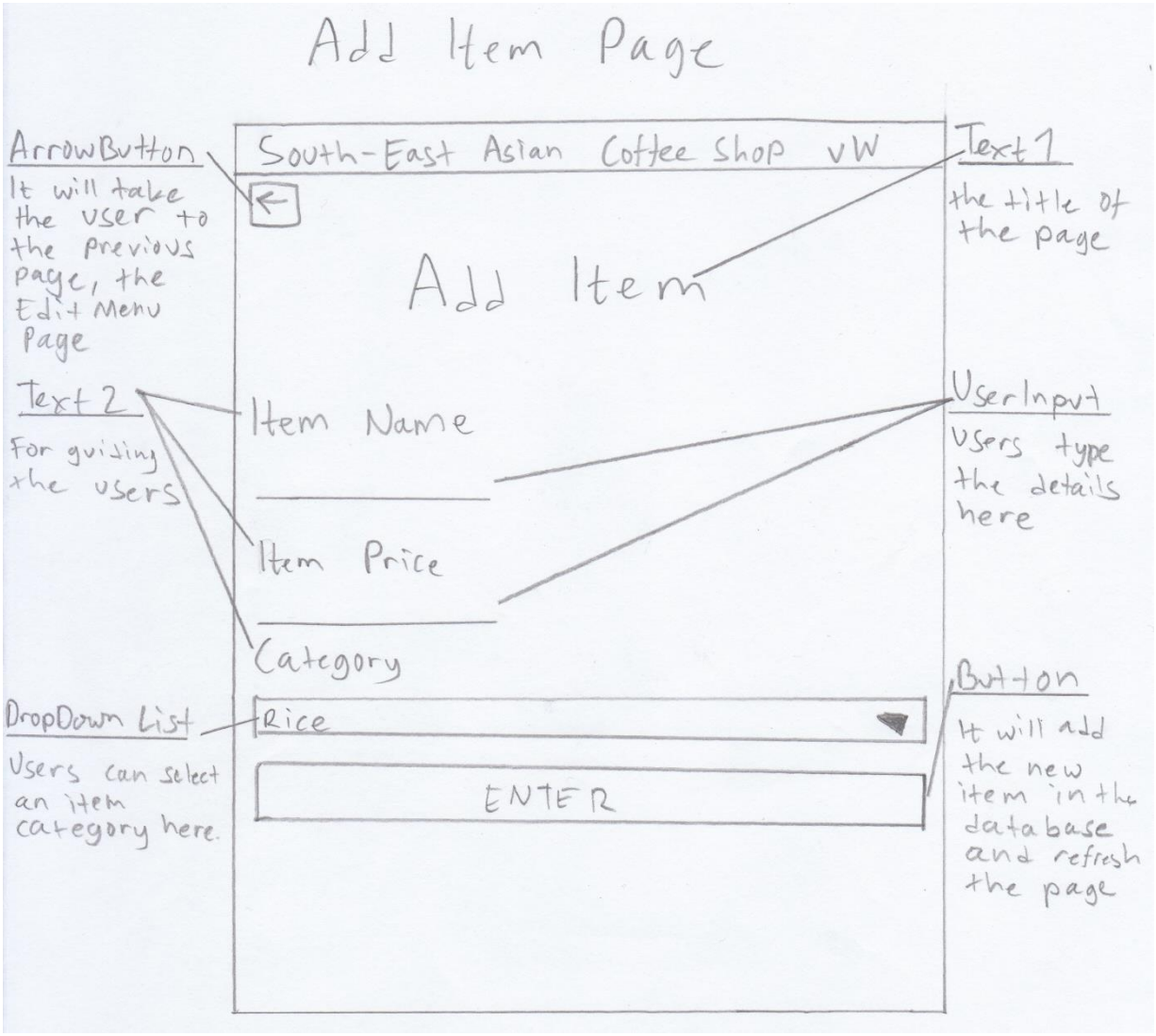
Should there be an error a popup will appear



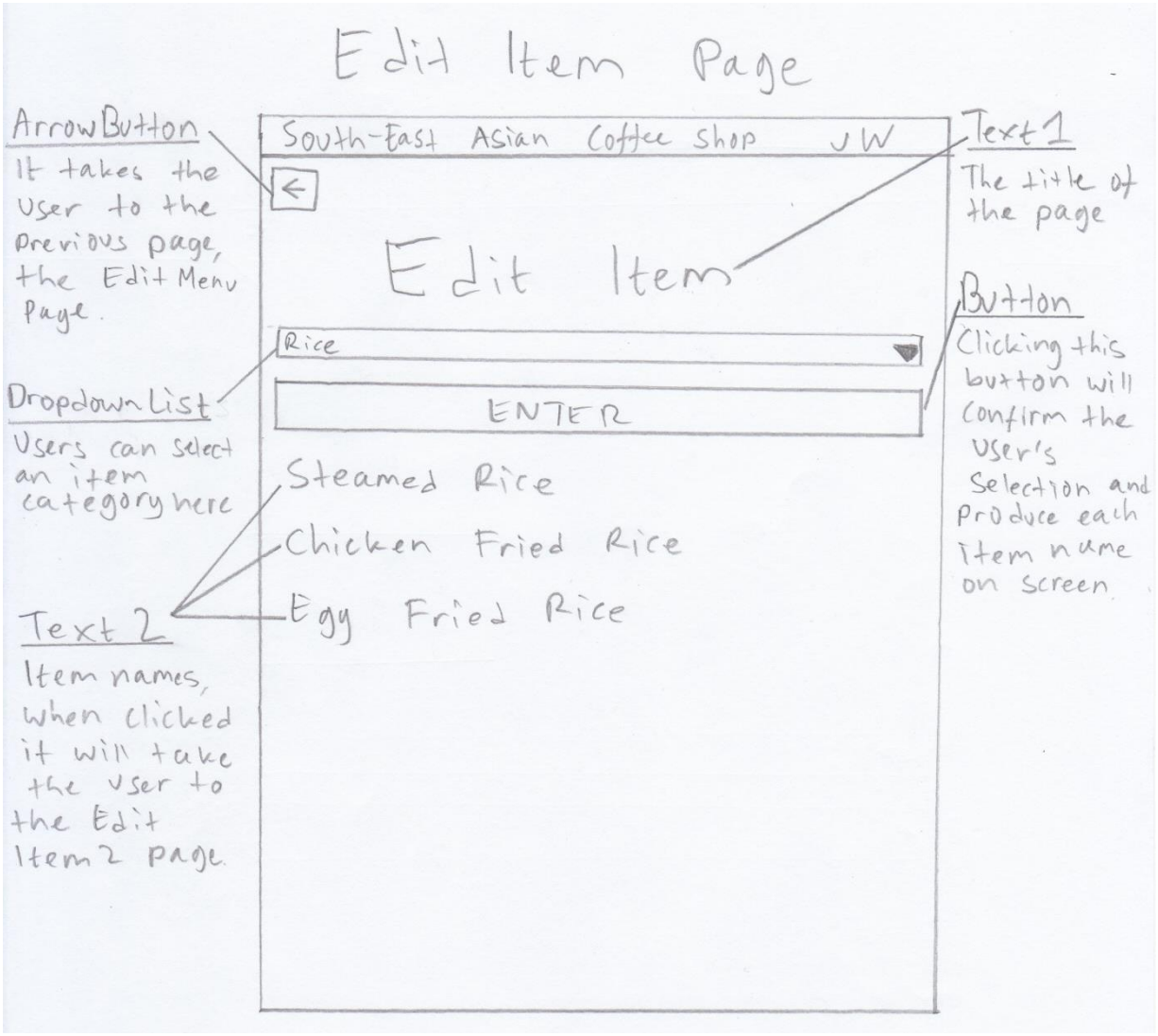
Text1 – Font size 24sp, centralized
Text2 – Font size 18sp, centralized
Text3 – Font size 14sp, item names left aligned and price numbers centralized
Table – Two columns
Button – Font size 14sp, centralized



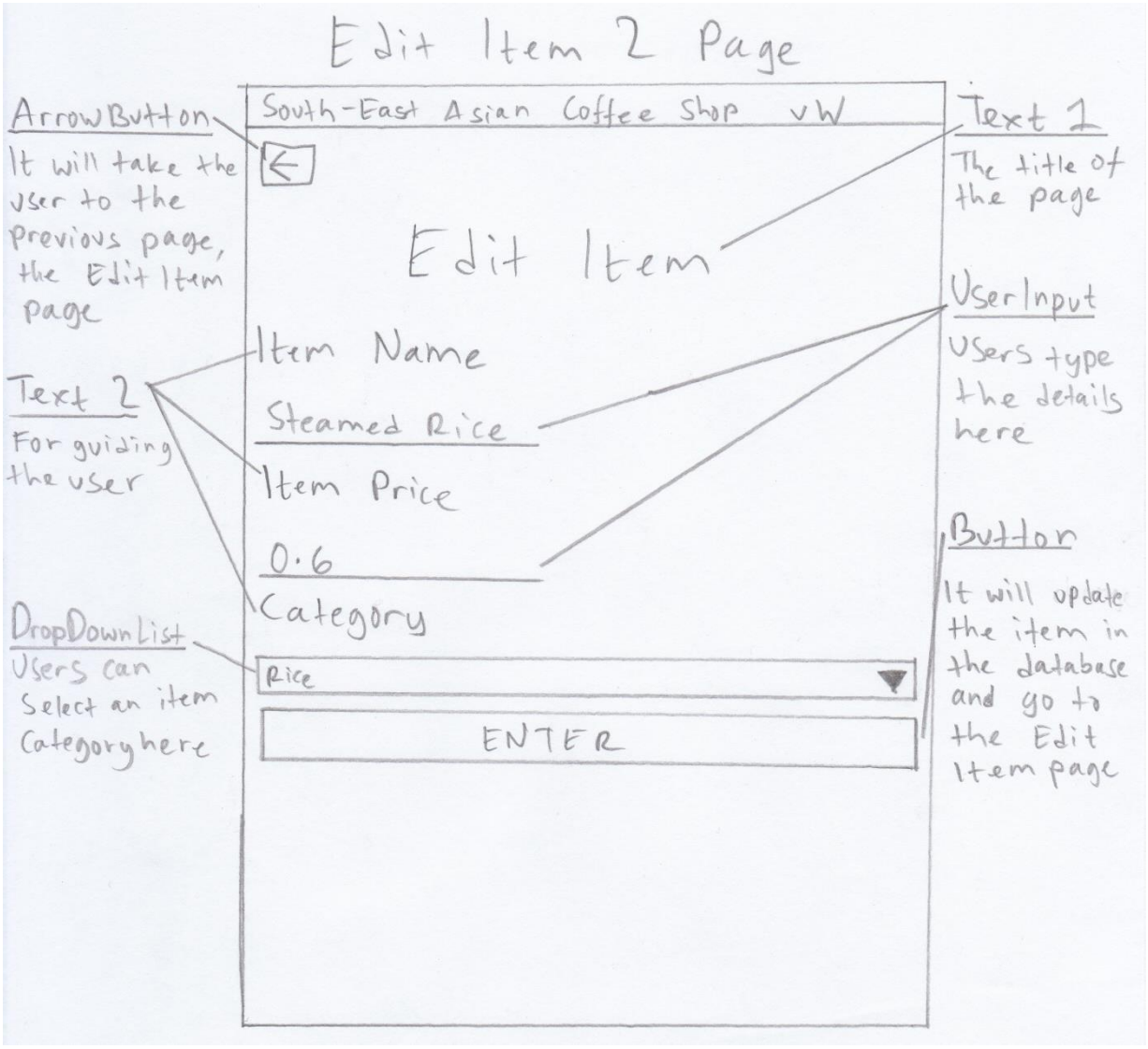
Text1 – Font size 24sp, centralized
Buttons – Font size 14sp, centralized



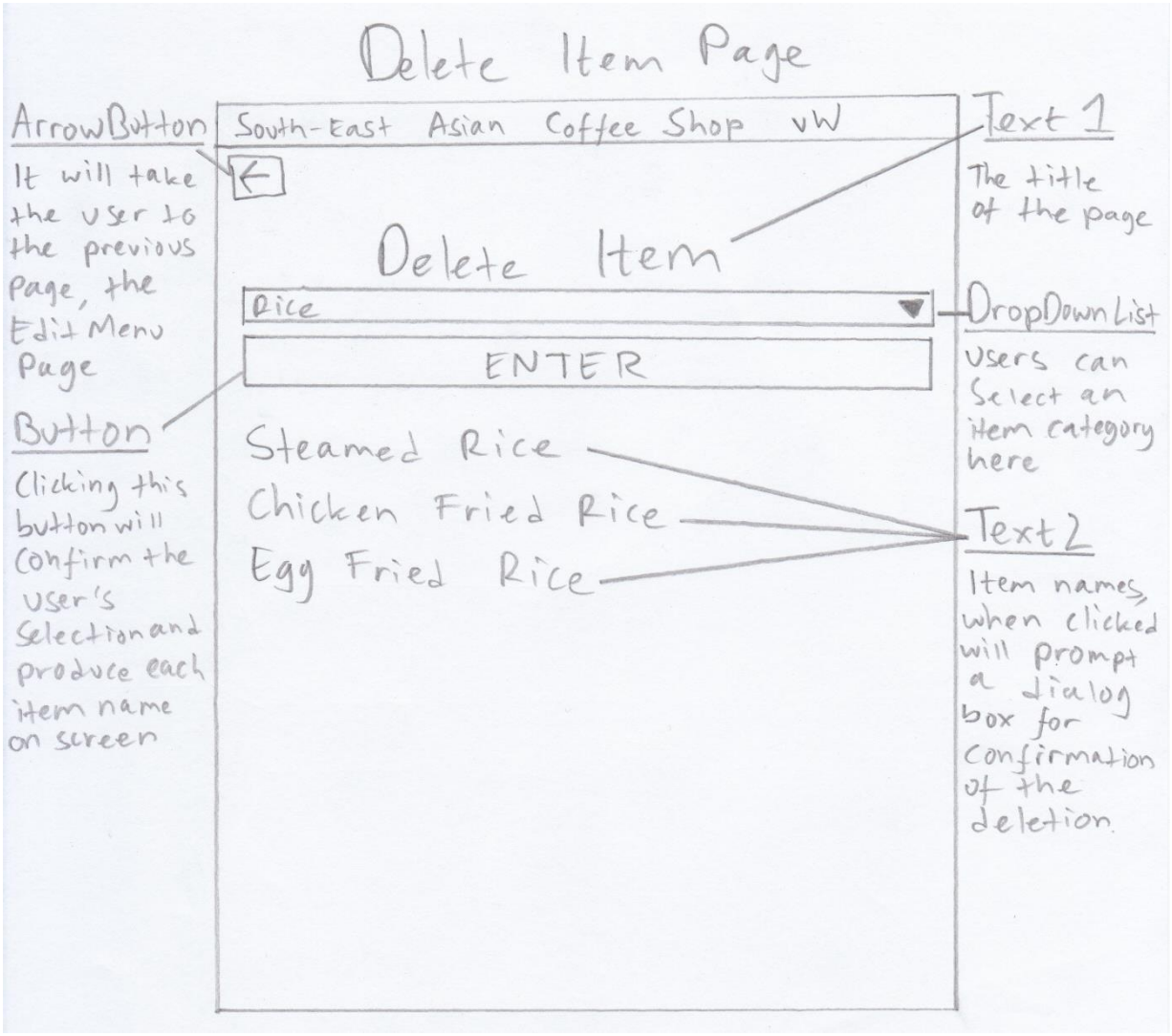
Text1 – Font size 24sp, centralized
Text2 – Font size 18sp, left aligned
UserInput – Font size 14sp, left aligned, no set text
Button – Font size 14sp, centralized
Should there be an error a popup will appear



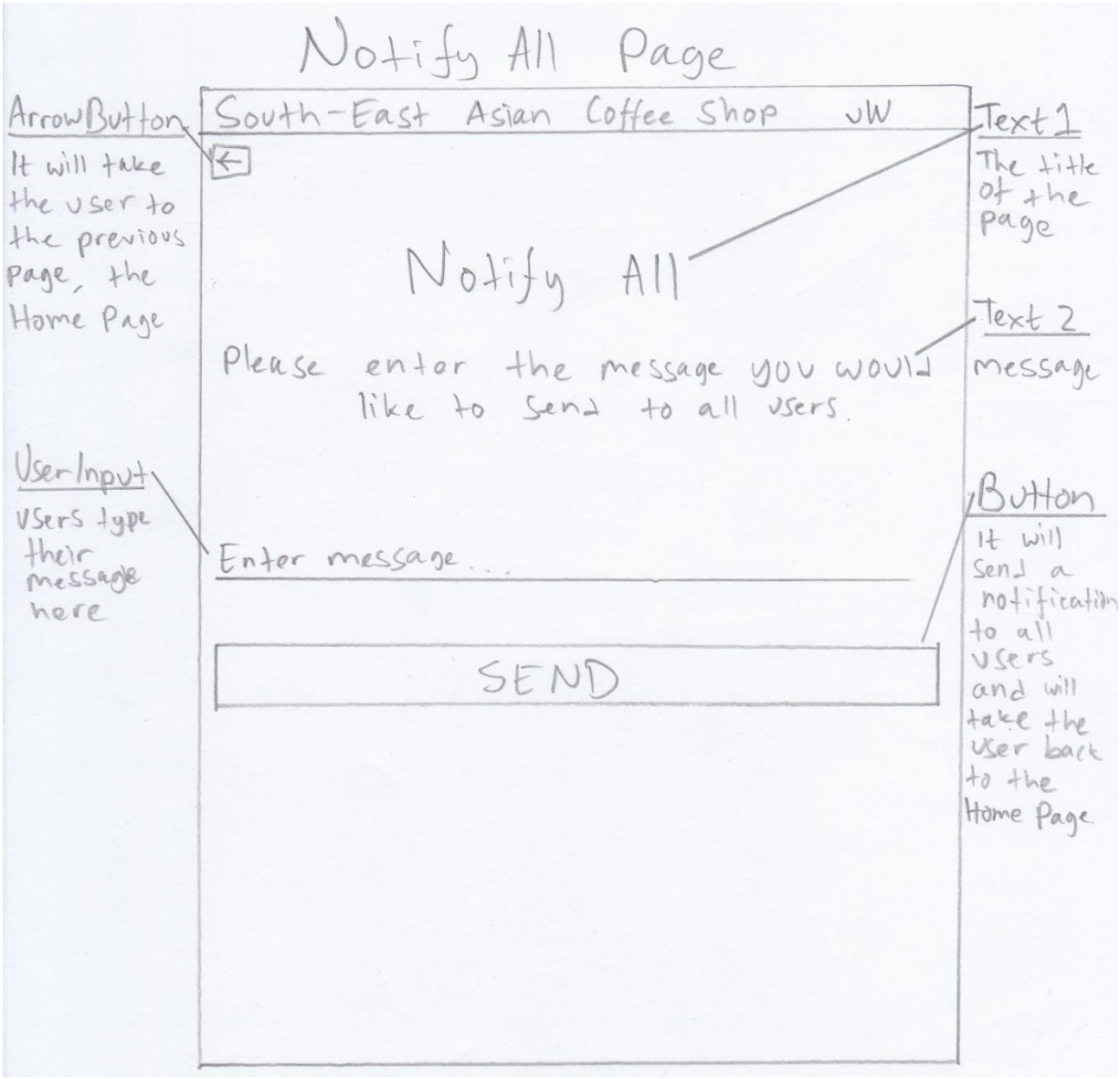
Text1 – Font size 24sp, centralized
Text2 – Font size 23sp, left aligned
Button – Font size 14sp, centralized



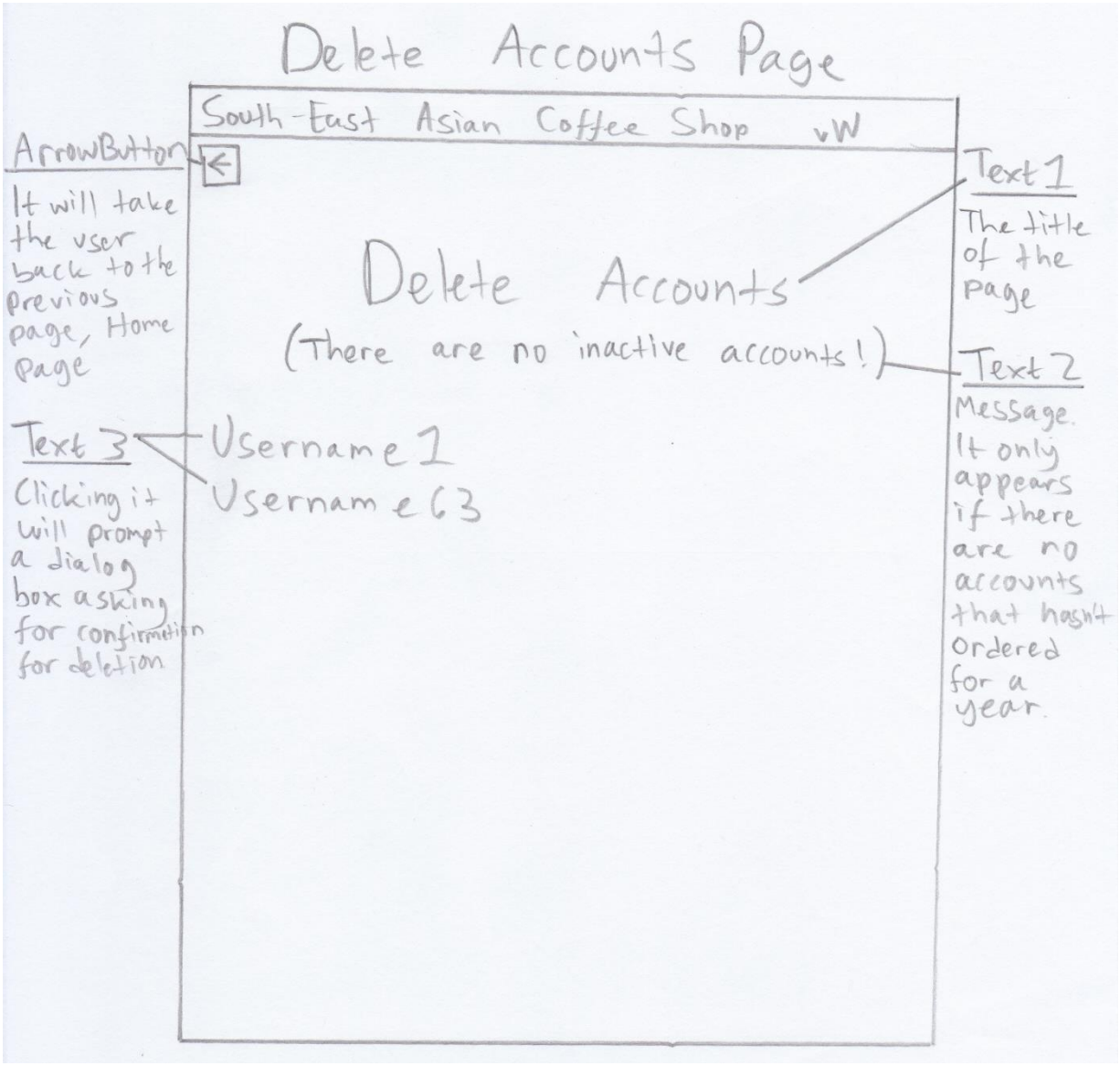
Text1 – Font size 24sp, centralized
Text2 – Font size 18sp, left aligned
UserInput – Font size 14sp, left aligned, the original values are set as the text
Button – Font size 14sp, centralized
Should there be an error a popup will appear



Text1 – Font size 24sp, centralized
Text2 – Font size 23sp, left aligned
Button – Font size 14sp, centralized



Text1 – Font size 24sp, centralized
Text2 – Font size 18sp, centralized
UserInput – Font size 14sp, left aligned
Button – Font size 14sp, centralized
Should there be an error a popup will appear



Text1 – Font size 24sp, centralized
Text2 – Font size 18sp, centralized
Text3 – Font size 23sp, left aligned

Data Entry Selections

I have used the following options for users to enter data for a reason.

Entry Design/Objects	Reasoning
Radio Button (Option Buttons)	This will limit the input the users can make, so

	that there is also less room for errors.
Spinners (Drop Down Lists)	This will also limit the input the users can make, however with spinners I can have many options and the layout will still look professional and nice.
Buttons	Buttons make a good navigation button for something to happen or for a process to occur.
EntryText (Input Field)	This will allow the user to type what they wish, which is sometimes required for example when typing their name, because there is no way of limiting what name a user can have since it varies around the world. However, some EntryTexts will have validations when for example only numbers can be typed.

Font Selections

I have used the following fonts for a reason.

Description	Reasoning
Sizes	
24sp – Titles of pages	This is the biggest font size in all of the layouts and will be given to the title pages. The title being the biggest font size makes it stand out and ensures that the user is in the certain page.
23sp – Text generated programatically that are clickable	It is slightly smaller than title because these texts need to be easily read and easily clicked by the user, if it is too small they may click the wrong text by accident.
18sp – Field Titles and messages	This is one step smaller from the titles and are the next thing that will catch the user's eyes.
14sp – Entry fields and Buttons	These texts will have hints telling them to input something and they are given the smallest size

	as it is something they will have to change.
Styles	
Sans Serif	Only one font style is used to keep everything consistent and it looks professional.
Colours	
Green – Heading bar	These two colours were selected to represent the sign outside of the coffee shop. The sign has a green background with yellow text. Using the same colour scheme will create a sense of familiarity with the users.
Yellow – Text in the heading bar	
Dark Green – Above the heading bar	A darker shade of the heading bar is used to differentiate that this is a seperate area.
Black – All other texts	Black and white will be used for simplicity. Black text on white background works well as it is a very strong contrast, making it easy for users to read the text. Having too many colours can also have a clash and make it hard for users to read, hence the simplicity.
White – The main screen	

Data Validation

Validation will ensure that the user inputs are what I want them to be. They will exclude errors and unwanted data.

Validation Check Type	Description
Range	Checks that the inputted data is within the specified range
Presence	Checks that all the data needed is present
List	This will only allow certain values to be selected

Activity	Validation	Description	Error Message	Valid	Erroneous

	Check			Data	Data
Changing the user Password or creating a new account and typing the password	Range	The string has to be grater than or equal to 6 characters	The password has to be at least 6 characters!	Banana	star
Typing the quantity desired for ordering	Range	The quantity has to be greater than 0	The quantity has to be greater than 0!	5	0
Making a new account	Presence	Make sure every field has been entered in	You have missed out on a field!	“Trevor”	[empty]
Selecting item categories or selecting date for order history	List	Makes sure that the users can only select what is on the list	N/A	“Fish”	N/A

Data Structure

Using an XML file, values of strings are stored in a seperate resource file. This resource file will contain all the strings that will be represented in the app in buttons, textviews, spinners and so on. Main ones such as titles of pages like “Pending Orders” are stored here, and also messages such as “Insert Item Name”. String arrays will also be stored here such as the names of each item category and also the options for sending notifications. Having the strings stored in a resource files is more efficient than hard coding it in the layout file, as for example, titles with the same name will automatically changed if they are used in more than one layout when you change it from the resource file.

I also have a resource file for colors that are used.

Security Measures

To prevent unauthorised access to other user’s password, the password that is stored will be hashed. Hashing provides a greater security than encrypting the value as it is one way. If I were to use encryption, if somehow an unauthorised person gets access to the key, they can then decrypt all the user’s password. Hence why hashing is used instead. I will use the hash SHA-256 as it is proven to be one of the strongest hashing algorithms available.

Pseudo Code

Here are the custom methods I will use in my code:

Calculating the 15% Discount

```
GET UserInput
DIVIDE UserInput BY 100
MULTIPLY UserInput BY 15
ROUND UserInput TO ONE DECIMAL PLACE
RETURN UserInput
```

HASH password with SHA-256

```
GET UserInput
USE SHA-256 Hash ON UserInput
CHANGE UserInput TO ARRAYS OF BYTES
FOR EACH SET OF BYTES:
   CHANGE each set to a HEXADECIMAL STRING
   IF LENGTH OF HEXADECIMAL STRING equals 1:
      ADD zero to the HEXADECIMAL STRING
   END IF
   ADD HEXADECIMAL STRING to an empty variable
END FOR
CONVERT THE variable to a STRING
RETURN STRING
```

Changing Page

```
GET PageDestination
IF needed:
   PASS Variables to PageDestintion
END IF
OPEN PageDestination
CLOSE CurrentPage
```

Dialog Box for Password Confirmation

```
GET message
MAKE DialogBox
SET message TO DialogBox
MAKE UserInputField
SET UserInputField TO DialogBox
SET InputType FOR UserInputField TO Password
SET EnterButton TO DialogBox
```

```
WHEN EnterButton CLICKED:
   GET UserInput
   COMPARE UserInput WITH PasswordCheck
   IF EQUAL:
      PERFORM OPERATION
   ELSE:
      DISPLAY ErrorMessage
   END IF
   CLOSE DialogBox
SET CancelButton TO DialogBox
WHEN CancelButton CLICKED:
   CLOSE DialogBox
DISPLAY DialogBox
```

User Interface Designs

Key	
Diagram Language	**XML Language**
UserInput	EntryText
Text	TextView
Dropdown List	Spinner
Radio Button	RadioButton
Popup	Toast
Button	Button
ArrowButton, InfoButton	ImageButton

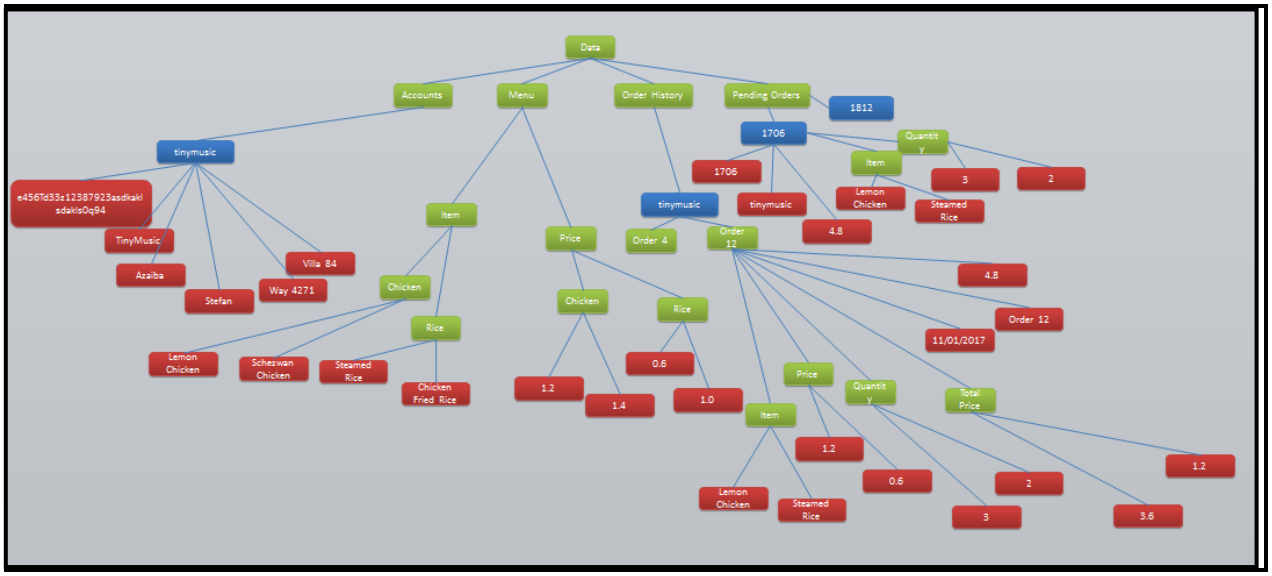
Database Structure

A JSON Tree database structure will be used for this project. The data is stored neatly in their respective branches, for example all things regarding the user's details is stored under the Accounts branch.

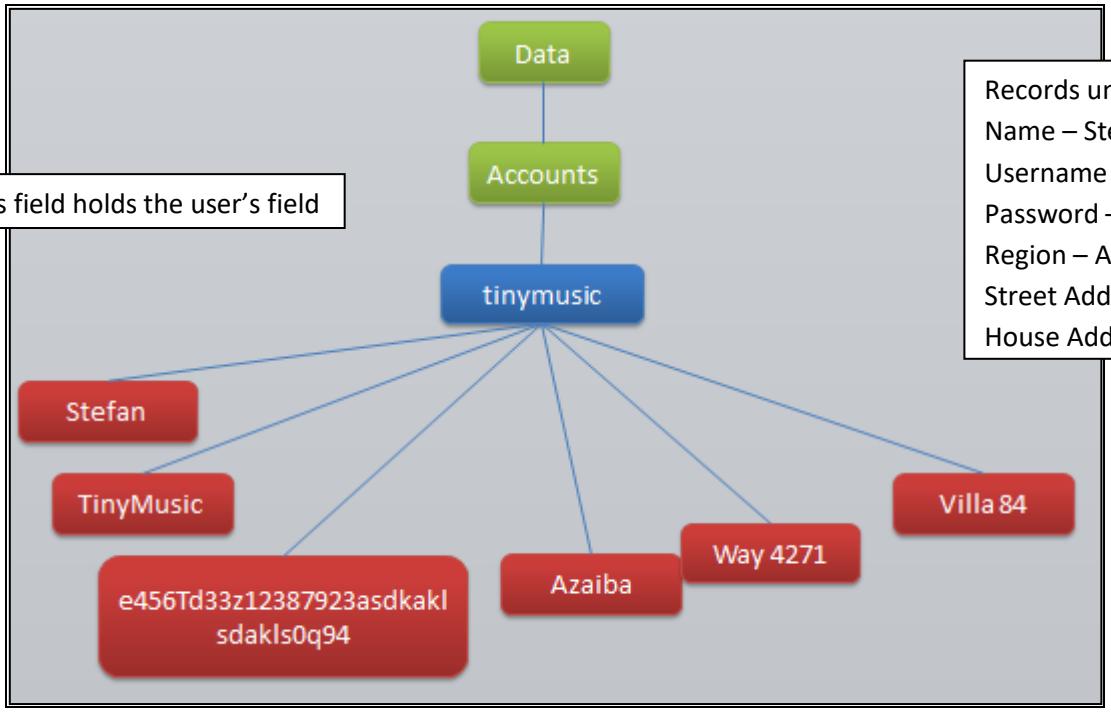
There will be two temporary storages, one is titled “Public Notification” which is directly under Data and “Order Status” which is under the username of the account. These will disappear after some time and only exists as temporary messages to the users.

This is an overview of the structure:

Key	
Blue	Field dependent on User
Green	Field
Red	Record



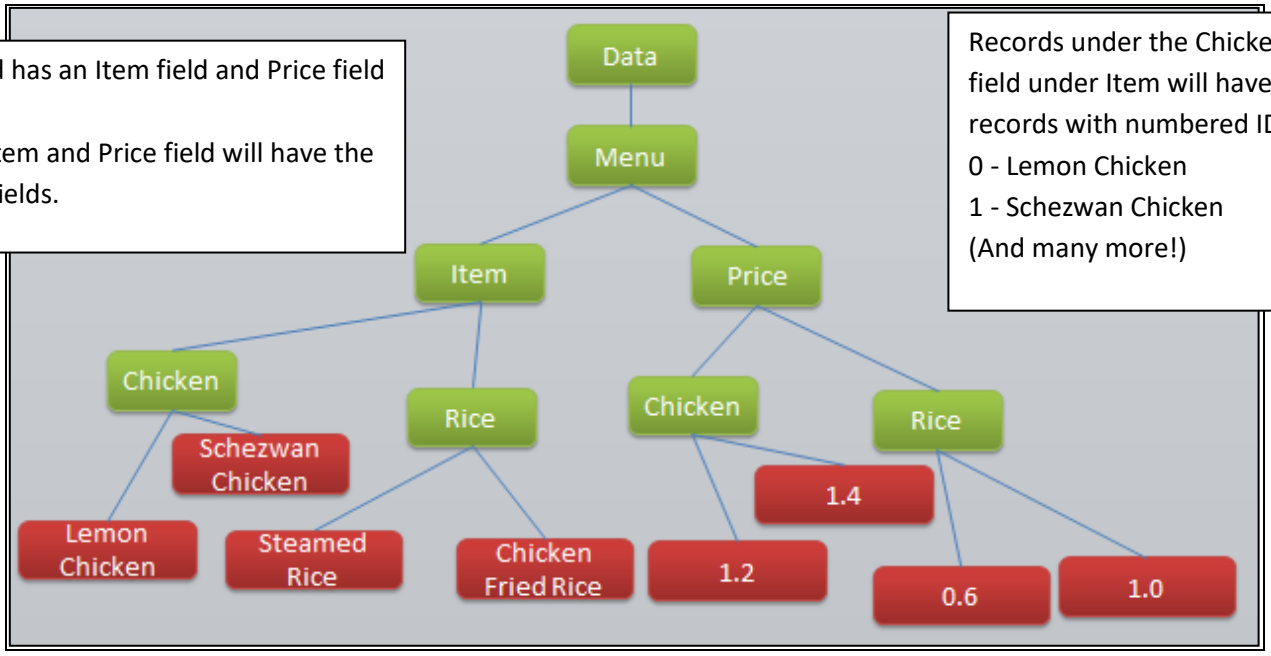
Accounts:



Accounts field holds the user's field

Records under the Field:
Name – Stefan
Username – TinyMusic
Password – [Hashed Value]
Region – Azaiba
Street Address – Way 4271
House Address – Villa 84

Menu:

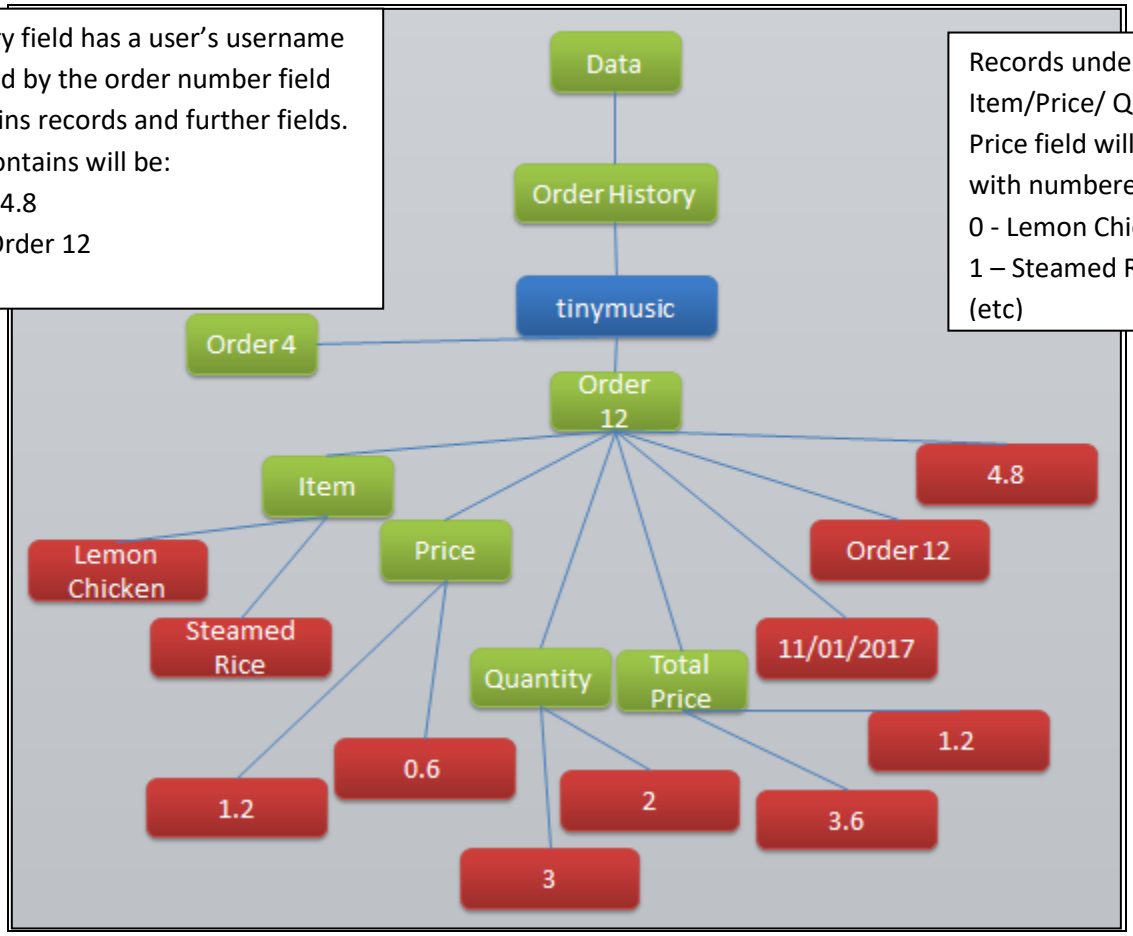


Menu field has an Item field and Price field under it.
Both the Item and Price field will have the Category fields.

Records under the Chicken field under Item will have records with numbered IDs:
0 - Lemon Chicken
1 - Schezwan Chicken
(And many more!)

Order History:

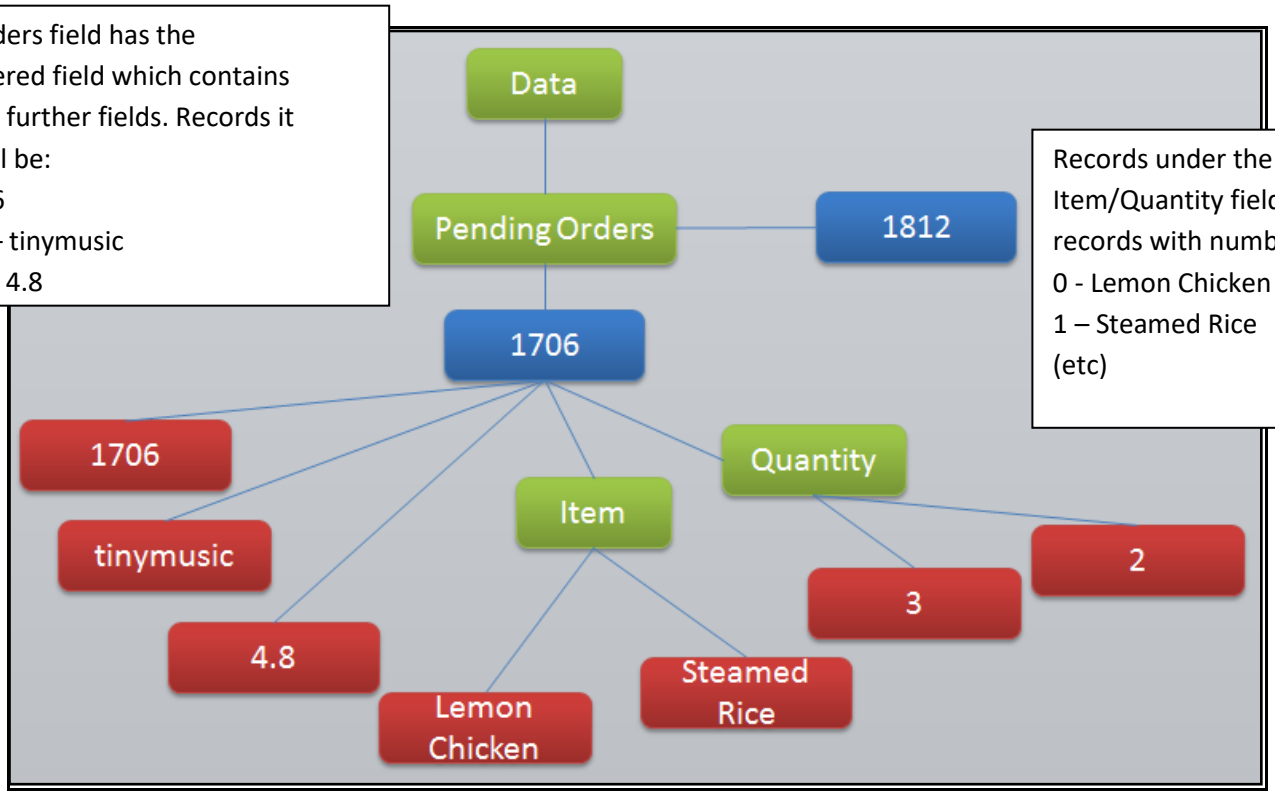
Order History field has a user's username field followed by the order number field which contains records and further fields.
Records it contains will be:
Total Cost – 4.8
OrderNo – Order 12



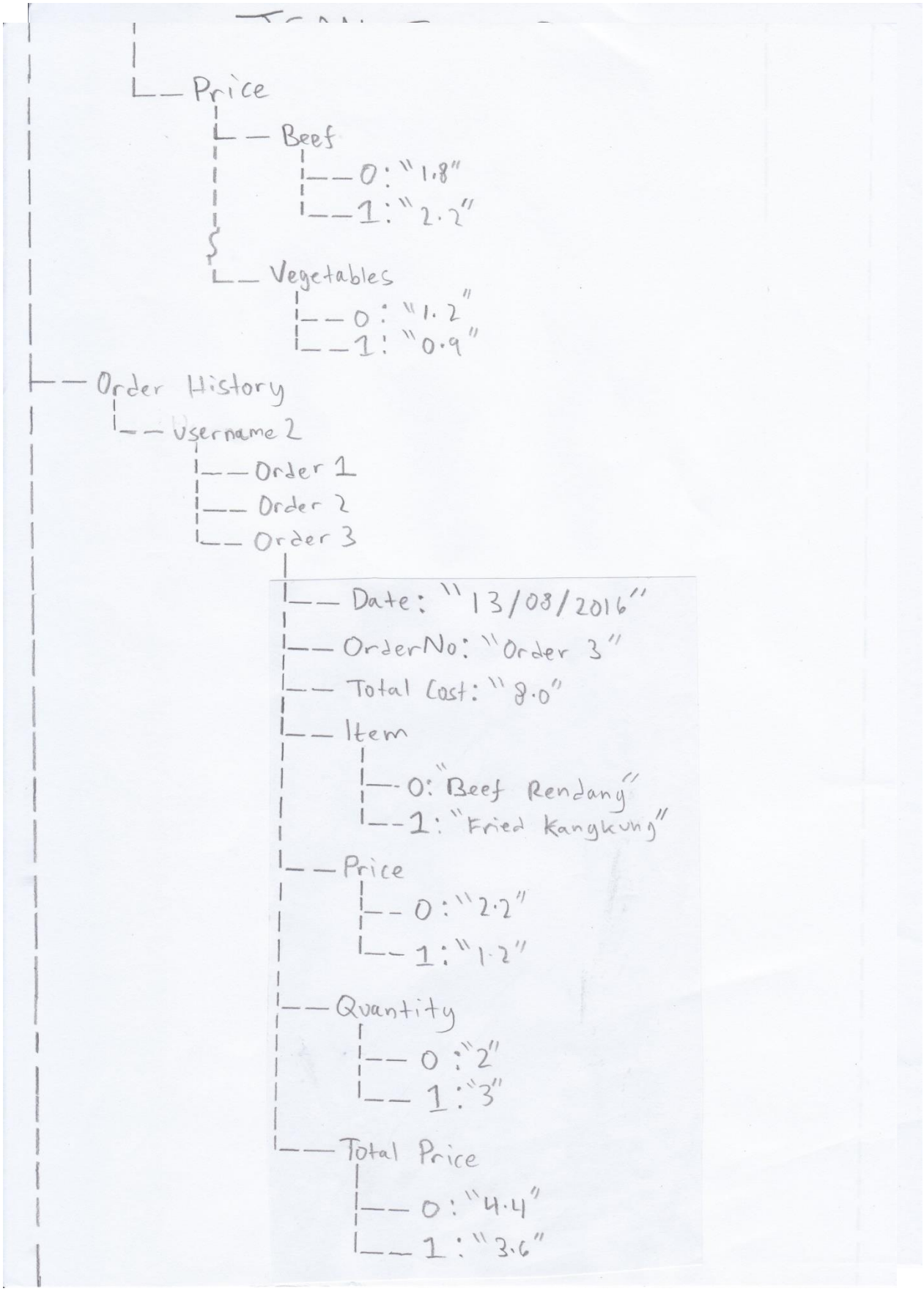
Records under the Item/Price/ Quantity/Total Price field will have records with numbered IDs:
0 - Lemon Chicken
1 – Steamed Rice
(etc)

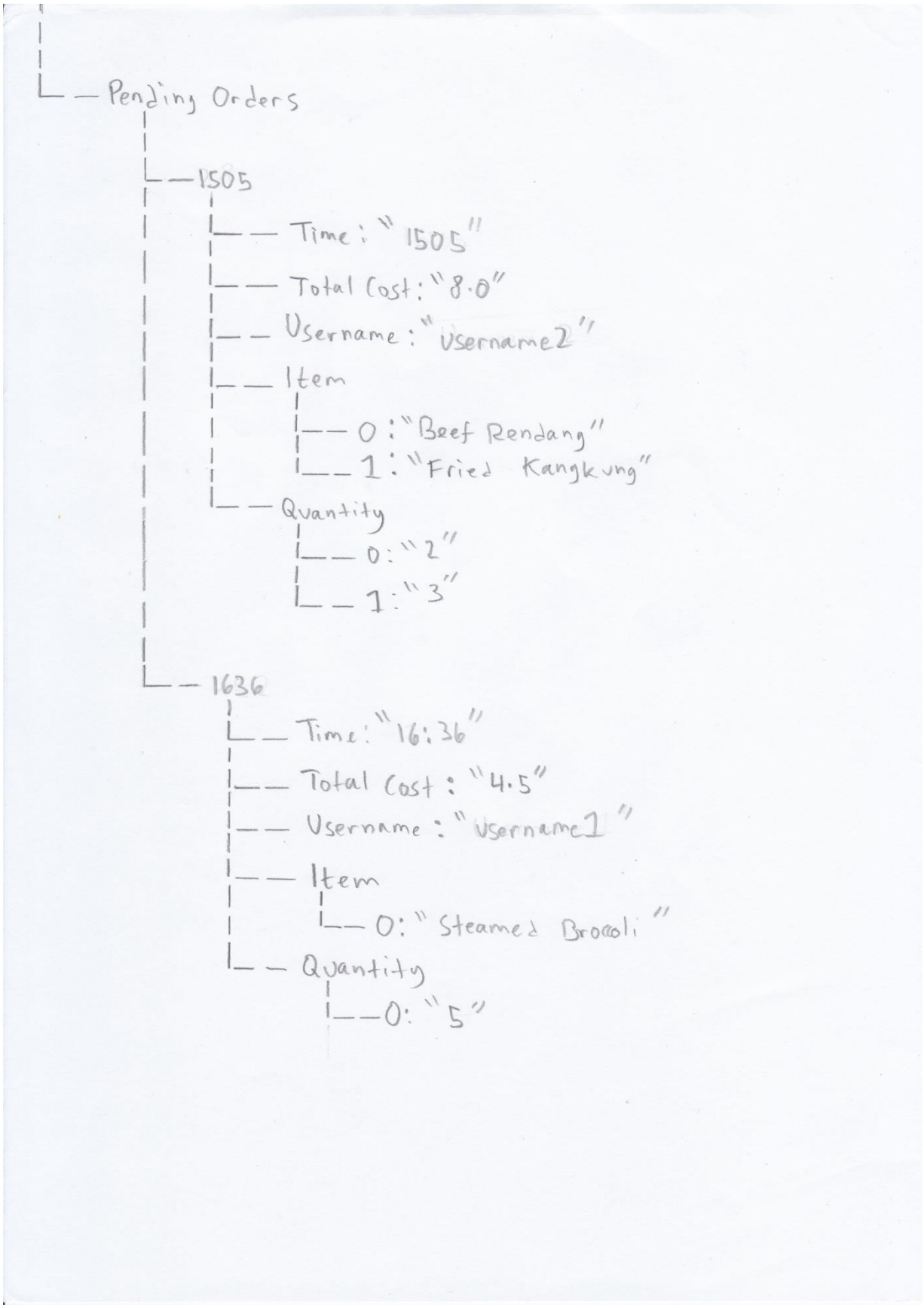
Pending Orders:

Pending Orders field has the TimeOfOrdered field which contains records and further fields. Records it contains will be:
Time – 1706
Username – tinymusic
Total Cost – 4.8



Records under the Item/Quantity field will have records with numbered IDs:
0 - Lemon Chicken
1 – Steamed Rice
(etc)


Price
Beef
0: "1.8"
1: "2.2"
Vegetables
0: "1.2"
1: "0.9"
Order History
Username 2
Order 1
Order 2
Order 3
Date: "13/03/2016"
OrderNo: "Order 3"
Total Cost: "8.0"
Item
0: "Beef Rendang"
1: "Fried Kangkung"
Price
0: "2.2"
1: "1.2"
Quantity
0: "2"
1: "3"
Total Price
0: "4.4"
1: "3.6"


Pending Orders
1505
Time: "1505"
Total Cost: "8.0"
Username: "Username2"
Item
0: "Beef Rendang"
1: "Fried Kangkung"
Quantity
0: "2"
1: "3"
1636
Time: "16:36"
Total Cost: "4.5"
Username: "Username1"
Item
0: "Steamed Broccoli"
Quantity
0: "5"

Test Plan

Before the app is released for the public to use, it must be tested.

Customer App

Log In Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
1	To test the entry fields function properly (upon pressing the log in button)	Expected Data	Correct username (regardless of upper/lower case) and correct password input	Logging in success, taken to home page
		Erroneous Data	Correct username and incorrect password input	Error Message, saying either username or password is incorrect
		Erroneous Data	Incorrect username and correct password input	Error Message, saying either username or password is incorrect
		Erroneous Data	Incorrect username and incorrect password input	Error Message, saying either username or password is incorrect
		Erroneous Data	One or more fields is left blank	Error Message, saying that a field was left blank
2	To test the navigation buttons works	Expected Data	Short/Long press	Log In: Takes user to the Home Page Sign Up: Takes user to the Sign Up Page

Sign Up Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
3	To test the entry fields function properly (upon pressing the sign up button)	Expected Data	Every field (including radio button) is filled, password field is at least 6 characters	Account created and stored in database, taken to sign up complete page
		Erroneous Data	1 or more fields is left blank	Error Message, saying that a field was left blank
		Erroneous Data	Password field is filled with 5 characters or less	Error Message, saying the password must at least be 6 characters
		Erroneous	Type a username that	Error Message, saying that

		Data	already exists on the database	the username is already in use
4	To test the radio buttons function properly	Expected Data	Option A is selected	Option A will be selected
		Expected Data	Option B is selected after Option A was selected	Option B will be selected and Option A will be unselected
5	To test the info buttons function properly	Expected Data	Info button is pressed	Password info should say "minimum of 6 characters required". House and Street info should give examples. Region info should say why only the two options is availabl.
6	To test the navigation buttons functions	Expected Data	Short/Long press	Arrow: Takes user back to the Log In page Sign Up: Takes user to the Sign Up Complete Page

Sign Up Complete Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
7	To test the navigation button functions	Expected Data	Short/Long press	Continue to Log In: Takes user to the Log In Page

Home Page

Test Plan	Purpose	Test Type	Test Data	Expected Outcome
8	To test exiting the app completely works	Expected Data	Short/Long press on button	Closes app
9	To test all navigation buttons function	Expected Data	Short/Long press on button	Make an Order: Takes users to the Make an Order Page Edit Account: Takes users to the Edit Account Page View Order History: Takes

				users to the View Order History Page
10	To test that you can only press the Make an Order button between 9AM to 9PM	Expected Data	Short/Long press on button, between 9AM and 9PM	Takes users to the Make an Order Page
		Erroneous Data	Short/Long press on button, Before 9AM and after 9PM	Error Message, saying that you can't order at this hour
11	To test the welcome message uses the user's name in it	Expected Data	Log in with a few accounts	For each account their name that was stored in the database should appear
12	To test that the "Logging In Success" message doesn't appear when coming from pages that is not the Log In Page	Expected Data	Change page to the Home Page from the 3 avaliable pages	The message will not appear.

Edit Account Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
13	To test that clicking the UPDATE button will summon a Dialog Box	Expected Data	Short/Long press	A dialog box appearing asking for password confirmation
14	To test that the CANCEL and UPDATE button in the notify dialog	Expected Data	Short/Long press	CANCEL – Closes dialog box
		Expected Data	Short/Long press, password typed correctly	UPDATE – Popup saying the account has been updated, takes user back to the home

	box works			page
		Erroneous Data	Short/Long press, password typed incorrectly	UPDATE – Popup saying the password was incorrect
15	To test the entry fields function properly (upon pressing the UPDATE button)	Expected Data	Every field (including radio button) is filled, password field is at least 6 characters	Database values are updated
		Erroneous Data	1 or more fields is left blank	Error Message, saying that a field was left blank
		Erroneous Data	Password field is filled with 5 characters or less	Error Message, saying the password must at least be 6 characters
16	To test the radio buttons function properly	Expected Data	Option A is selected	Option A will be selected
		Expected Data	Option B is selected after Option A was selected	Option B will be selected and Option A will be unselected
17	To test the entry fields has the user's current data written in them (except for password, including radio button option)	Expected Data	Log In with a few accounts	Each field should have the user's data written in them
18	To test all navigation buttons function	Expected Data	Short/Long press on button	Arrow/Update: Takes users to the Home Page

Make an Order Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
19	To test the navigation button functions	Expected Data	Short/Long press	Arrow: Takes users to the Home Page Check Order: Takes users to the Confirm Order Page

20	To test the spinner selection is correct (upon pressing the enter button)	Expected Data	Select a couple of categories, cross-reference to the database	Each category chosen should get a list of every food item in that category
21	To test if users can proceed to the Check Orders Page	Expected Data	Made 1 or more selection of orders and then press button	Taken to the Confirm Order Page with selected Orders
		Erroneous Data	Made no selections at all and then press button	Error Message, saying that the user had not made any selections
22	To test if pressing an item name will create a dialog box	Expected Data	Press a few item names	A dialog box appearing
23	To test that the food item has the correct price alongside it	Expected Data	Open up with a few food items, cross check to the database	The correct price is retrieved for each one
24	To test that users can only type integers when asked for how much portion is wanted (upon pressing OK)	Expected Data	Typing a number such as 7	7 is typed in the entry field and the item is added to the list
		Erroneous Data	Typing a character that's not a number	Not Applicable as the keyboard will be default to numbers only
		Extreme Data	Typing a float value such as 5.6	Not Applicable as the keyboard will not have a full stop
		Extreme Data	Typing a negative number such as -5	Not Applicable as the keyboard will not have a minus sign
		Extreme	Typing 0	Error Message, saying you can't order zero portions of

		Data		something
25	To test that the CANCEL and OK button in the dialog box works	Expected Data	Short/Long press	OK – Closes dialog box, saves selected item to list, popup saying it has been added CANCEL – Closes dialog box

Confirm Order Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
26	To test the navigation button functions	Expected Data	Short/Long press	Arrow: Takes users to the Make an Order Page Restart Order: Takes users to the Make an Order Page
27	To test the arrow button keeps the user's selected orders	Expected Data	Press arrow button and then press check order	List of selected orders still present
28	To test the Restart Order button gets rid of the user's selected orders	Expected Data	Press the restart order button and then press check order	Error Message, saying that the user had not made any selections
29	To test that the calculations are done correctly	Expected Data	View page	Quantity * Price = Total. Sum of each Total = Total Cost.
30	To test that the Confirm Order button will create a dialog box	Expected Data	Short/Long press button	A dialog box asking for password confirmation appearing
31	To test that typing the password for confirmation	Expected Data	Correct password input	Order will be uploaded to database
		Erroneous Data	Incorrect password input	Error Message, saying password is incorrect

	functions properly			
32	To test that the CANCEL and SEND ORDER button in the dialog box works	Expected Data	Short/Long press	SEND ORDER – Uploads order to database, takes user to the Order Complete Page CANCEL – Closes dialog box

Order Complete Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
33	To test the navigation button functions properly	Expected Data	Short/Long press	Continue to Home: Takes user to the Home Page

View Order History Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
34	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page
35	To test the spinner selection is correct (upon pressing the search button)	Expected Data	Select a month and year which the user have ordered in	Orders from that time period should be listed
		Erroneous Data	Select a month and year which the user have not ordered in	Error Message, saying the user has not ordered anything in the month and year chosen
36	To test that an order can be clicked	Expected Data	Short/Long press	Takes the user to the View Order History 2 Page

37	To test that when it is a new year it is automatically updated in the spinner	Expected Data	View the spinner selections	There should be options from 2013 to the current year.

View Order History 2 Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
38	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the View Order History Page
39	To test that the data downloaded is all correct	Expected Data	Do this with several orders, cross-check in database	All data correct

Worker App

Home Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
40	To test exiting the app completely works	Expected Data	Short/Long press on button	Closes app
41	To test all navigation buttons function	Expected Data	Short/Long press on button	Pending Orders: Takes users to the pending orders page View Menu: Takes users to the view menu page
42	Clicking the edit menu, notify all and delete accounts button should	Expected Data	Short/Long press on button	A dialog box appearing asking to input password

	prompt a password confirmation			
43	To test that the CANCEL and CONFIRM button in the dialog box works	Expected Data	Short/Long press	CANCEL – Closes dialog box
		Expected Data	Short/Long press, password typed correctly	CONFIRM – Takes users to the respective page, dialog box closes
		Erroneous Data	Short/Long press, password typed incorrectly	CONFIRM – Popup saying the password was incorrect
44	Text stating how many orders is pending should update in real time	Expected Data	Have home page screen open at the same time as someone making an order on the customer app	The number should increase by one
		Expected Data	Have home page screen open at the same time as someone completing a pending order on the worker app	The number should decrease by one

Pending Orders Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
45	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page User's order: Takes user to the View Order Page
46	Text stating how many orders is pending should	Expected Data	Have pending order screen open at the same time as someone making an order on the	The number should increase by one

	update in real time		customer app	
		Expected Data	Have pending order screen open at the same time as someone completing a pending order on the worker app	The number should decrease by one
47	Clickable text of a user's orders should update in real time	Expected Data	Have pending order screen open at the same time as someone making an order on the customer app	A new text should be added at the bottom of the existing list
		Expected Data	Have pending order screen open at the same time as someone completing a pending order on the worker app	That order should disappear from the list

View Order Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
48	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page
49	To test that the item, quantity and cost retrieved is correct	Expected Data	Cross-check to the database	Should be correct
50	The user's details button should prompt a dialog box stating the user's name and address only	Expected Data	Short/Long press	Name, Region, Street Address and House Address should be displayed

51	To test that the GOTCHA button in the user's details dialog box works	Expected Data	Short/Long press	GOTCHA – Closes dialog box
52	The notify button should prompt a dialog box for sending notification	Expected Data	Short/Long press	Default messages and a custom message with an entry field should be present
53	To test that the CANCEL button in the notify dialog box works	Expected Data	Short/Long press	CANCEL – Closes dialog box
54	To test that pressing the message options will send a notification to the user	Expected Data	Short/Long press, watch the phone of the user who made the order	A notification appearing in the user's phone
		Erroneous Data	When pressing the custom message but the entry field is empty	Error message saying there is no message written
55	The order completed button should prompt a dialog box asking for password confirmation	Expected Data	Short/Long press	A dialog box appearing with an entry field for the password of the user who made the order
56	To test that the CANCEL and CONFIRM button in the notify dialog box works	Expected Data	Short/Long press	CANCEL – Closes dialog box
		Expected Data	Short/Long press, password typed correctly	CONFIRM – Popup saying order has been completed, takes user back to the pending order page, the order that was completed should have disappeared from the list
		Erroneous	Short/Long press,	CONFIRM – Popup saying the

		Data	password typed incorrectly	password was incorrect

View Menu Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
57	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page
58	To test that selecting an item category from the spinner will produce the correct list of items	Expected Data	Select category and press the ENTER button	List of items from the correct category produced along with its price

Edit Menu Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
59	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page Add Item: Takes user to the Add Item Page Edit Item: Takes user to the Edit Item Page Delete Item: Takes user to the Delete Item Page

Add Item Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
60	To test the navigation button	Expected	Short/Long press	Arrow: Takes user to the Edit

	functions properly	Data		Menu Page
61	To test the entry fields function properly (upon pressing the confirm button)	Expected Data	All entry fields are filled in (inluding the spinner option)	Popup saying the new item is added, added to the database, the page is then refreshed
		Erroneous Data	1 or more entry field is not filled in	Popup saying that 1 or more field is missing

Edit Item Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
62	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Edit Menu Page
63	To test that selecting an item category from the spinner will produce the correct list of items	Expected Data	Select category and press the ENTER button	List of items from the correct category produced
64	To test that the produced list of items is clickable	Expected Data	Short/Long press	User will be taken to the Edit Item 2 page with the correct details of the item selected

Edit Item 2 Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
65	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Edit Item Page
66	To test that the entry field has the current data for the	Expected Data	Repeat a few times with different items and	The correct item name and price should be text values in

	item as text		cross-check in database	the entry field
67	To test that the confirm button works	Expected Data	Short/Long press	The database is updated with the new values, the user will be taken to the Edit Item Page and a popup saying it was a success should appear
		Erroneous Data	Short/Long press but one or more field is empty	Error message saying a field was left blank

Delete Item Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
68	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Edit Menu Page
69	To test that selecting an item category from the spinner will produce the correct list of items	Expected Data	Select category and press the ENTER button	List of items from the correct category produced
70	To test that the produced list of items is clickable	Expected Data	Short/Long press	A dialog box will prompt asking for confirmation of the deletion
71	To test that the CANCEL and CONFIRM button in the dialog box works	Expected Data	Short/Long press	CONFIRM – a popup saying the item was deleted will appear, and the dialog box will close CANCEL – Closes dialog box

Notify All Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
72	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page
73	To test that entering nothing will prompt an error message	Expected Data	Leave the entry field blank and press SEND	An error message appearing saying there was nothing written
74	To test that all the customers gets the notification	Expected Data	Type a message and press the SEND button	All users should get the notification

Delete Account Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
75	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page
76	To test the page functions properly	Expected Data	If an account has not ordered in more than one year	That account should appear on the page and is clickable for deletion
		Expected Data	If there is no account that has not ordered more than one year	A text should say that there is no inactive accounts
77	To test that clicking on a name will prompt a dialog box	Expected Data	Short/Long press	A dialog box appearing
78	To test that the CONFIRM and CANCEL buttons in the dialog box	Expected Data	Short/Long press	CONFIRM – Deletes the account from the database, account details and order history, popup saying the

	works			deletion is a success and the page is refreshed CANCEL – closes the dialog box

Technical Solution

Table of Contents

My solution has met all my objectives.

Order App vW Resources

Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.nadramon.orderappvw">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="South-East Asian Coffee Shop vW"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".HomePage"
            android:configChanges="orientation"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".PendingOrders"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity android:name=".ViewMenu"
```

```
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity android:name=".EditMenu"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity android:name=".NotifyAll"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity android:name=".DeleteAccounts"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity android:name=".ViewOrder"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity android:name=".AddItem"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity android:name=".EditItem"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity android:name=".DeleteItem"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity android:name=".EditItem2"
            android:configChanges="orientation"
            android:screenOrientation="portrait"></activity>
    </application>

</manifest>
```

Colors

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#228B22</color>
    <color name="colorPrimaryDark">#008000</color>
    <color name="colorAccent">#FF4081</color>
    <color name="titleColor">#FFFF00</color>
    <color name="black">#000000</color>

</resources>
```

Strings

```
<resources>
    <string name="app_name">Order App vW</string>

    <string name="welcome">Welcome!</string>
    <string name="pending_orders">Pending Orders</string>
    <string name="view_menu">View Menu</string>
    <string name="edit_menu">Edit Menu</string>
    <string name="notify_all">Notify All</string>
    <string name="delete_Accounts">Delete Accounts</string>
    <string name="exit_app">Exit App</string>
```

```
    <string name="item">Item</string>
    <string name="price">Price</string>

    <string name="add_item">Add Item</string>
    <string name="edit_item">Edit Item</string>
    <string name="delete_item">Delete Item</string>

    <string name="item_name">Item Name</string>
    <string name="item_price">Item Price</string>
    <string name="insert_item_name">Insert Item Name</string>
    <string name="insert_item_price">Insert Item Price</string>
    <string name="category">Category</string>

    <string name="view_order">View Order</string>
    <string name="users_details">User\'s Details</string>
    <string name="notify">Notify</string>
    <string name="order_completed">Order Completed</string>

    <string name="enter">Enter</string>
    <string name="send">Send</string>
    <string name="processing">Currently Processing....</string>
    <string name="no_inactive_accounts">There are no inactive accounts!</string>
    <string name="enter_message">Please enter the message you would like to send
to all users.</string>

    <string-array name="Categories">
        <item>Starters</item>
        <item>Soup</item>
        <item>Beef</item>
        <item>Chicken</item>
        <item>Duck</item>
        <item>Fish</item>
        <item>Vegetables</item>
        <item>Rice</item>
        <item>Noodles</item>
        <item>Desert</item>
        <item>Beverages</item>
    </string-array>

    <string-array name="Processes">
        <item>Viewed</item>
        <item>Cooking</item>
        <item>Delivering</item>
        <item>Arrived</item>
        <item>Custom Message</item>

    </string-array>


</resources>
```

Styles

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
```

```
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <style name="myDialog" parent="Theme.AppCompat.Dialog">
        <item name="android:windowNoTitle">true</item>
    </style>

</resources>
```

Order App vW Gradle Scripts

Build.Gradle (Project: OrderAppvW)

```
// Top-level build file where you can add configuration options common to all
sub-projects/modules.

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.2.0'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
        classpath 'com.google.gms:google-services:3.0.0'
    }
}

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Build.Gradle (Module: App)

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion "24.0.3"
    defaultConfig {
        applicationId "com.nadramon.orderappvw"
        minSdkVersion 18
```

```
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:24.2.1'
    compile 'com.google.firebase:firebase-database:9.6.0'
    testCompile 'junit:junit:4.12'
}

apply plugin: 'com.google.gms.google-services'
```

This allows me to connect to the firebase database, hence achieving objective 3a.

Order App vW Layout

Home Page

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_home_page"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.orderappvw.HomePage">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/welcome"
        android:id="@+id/Welcome"
        android:textColor="@color/black"
        android:textAlignment="center"
        android:layout_alignParentEnd="true"
        android:layout_alignParentStart="true"
        android:textSize="24sp" />

    <TextView
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_below="@+id/Welcome"
        android:layout_marginTop="21dp"
        android:textColor="@color/black"
        android:text="@string/processing"
        android:id="@+id/LiveCount"
        android:layout_alignParentEnd="true"
        android:layout_alignParentStart="true"
        android:textAlignment="center"
        android:textSize="18sp" />

    <Button
        android:text="@string/pending_orders"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="26dp"
        android:id="@+id/button"
        android:onClick="toPO"
        android:layout_below="@+id/LiveCount"
        android:layout_alignParentEnd="true"
        android:layout_alignParentStart="true" />

    <Button
        android:text="@string/view_menu"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="18dp"
        android:id="@+id/button2"
        android:onClick="toVM"
        android:layout_below="@+id/button"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />

    <Button
        android:text="@string/edit_menu"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="18dp"
        android:id="@+id/button3"
        android:onClick="toEM"
        android:layout_below="@+id/button2"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />

    <Button
        android:text="@string/notify_all"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="15dp"
        android:id="@+id/button4"
        android:onClick="toNA"
        android:layout_below="@+id/button3"
        android:layout_alignStart="@+id/button3"
        android:layout_alignParentEnd="true" />

    <Button
        android:text="@string/delete_Accounts"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_marginTop="16dp"
        android:id="@+id/button5"
        android:onClick="toDA"
        android:layout_below="@+id/button4"
        android:layout_alignStart="@+id/button4"
        android:layout_alignParentEnd="true" />

    <Button
        android:text="@string/exit_app"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button6"
        android:onClick="Shutdown"
        android:layout_alignParentBottom="true"
        android:layout_alignStart="@+id/button5"
        android:layout_marginBottom="12dp"
        android:layout_alignParentEnd="true" />
</RelativeLayout>
```

Pending Orders Page

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:id="@+id/activity_pending_orders"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.orderappvw.PendingOrders">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:id="@+id/POarrow"
        android:onClick="Back"/>

    <TextView
        android:text="@string/pending_orders"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/POtitle"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:textSize="24sp" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/processing"
        android:id="@+id/POlivecount"
        android:textAlignment="center"
```

```
        android:textColor="@color/black"
        android:textSize="18sp" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/POlinearLayout"
            android:orientation="vertical" />
    </ScrollView>
</LinearLayout>
```

View Order Page

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:id="@+id/activity_view_order"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.orderappvw.ViewOrder"
    android:weightSum="1">


    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:onClick="Back"
        android:id="@+id/VOarrow" />

    <TextView
        android:text="@string/view_order"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/VOtitle"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:textSize="24sp" />

    <TableLayout
        android:id="@+id/VOtableLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
    </TableLayout>

    <Button
```

```
        android:text="@string/users_details"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/VOuserdetails"
        android:onClick="UsersDetails" />

    <Button
        android:text="@string/notify"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/VOnotify"
        android:onClick="Notify" />

    <Button
        android:text="@string/order_completed"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/VOcomplete"
        android:onClick="Completed" />
</LinearLayout>
```

View Menu Page

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:id="@+id/activity_view_order"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.orderappvw.ViewMenu">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:onClick="Back"
        android:id="@+id/VMarrow" />

    <TextView
        android:text="@string/view_menu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:id="@+id/VMtitle"
        android:textAlignment="center"
        android:textSize="24sp" />

    <Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/VMspinner" />
```

```
    <Button
        android:text="@string/enter"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="GetItems"
        android:id="@+id/VMenter" />

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:id="@+id/VMtableLayout">
    </TableLayout>
</LinearLayout>
```

Edit Menu Page

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_edit_menu"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.orderappvw.EditMenu">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:id="@+id/EMarrow"
        android:onClick="Back" />

    <TextView
        android:text="@string/edit_menu"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/EMtitle"
        android:textColor="@color/black"
        android:textAlignment="center"
        android:textSize="24sp"
        android:layout_below="@+id/EMarrow"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />

    <Button
        android:text="@string/add_item"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/EMtitle"
```

```
        android:layout_marginTop="25dp"
        android:id="@+id/EMadd"
        android:onClick="AddItem"
        android:layout_alignParentEnd="true"
        android:layout_alignParentStart="true" />

    <Button
        android:text="@string/edit_item"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/EMadd"
        android:layout_alignParentStart="true"
        android:layout_marginTop="26dp"
        android:id="@+id/EMedit"
        android:layout_alignParentEnd="true"
        android:onClick="EditItem" />

    <Button
        android:text="@string/delete_item"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/EMedit"
        android:layout_alignParentStart="true"
        android:layout_marginTop="29dp"
        android:id="@+id/EMdelete"
        android:layout_alignParentEnd="true"
        android:onClick="DeleteItem" />
</RelativeLayout>
```

Add Item Page

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_add_item"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.orderappvw.AddItem">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:id="@+id/AIarrow"
        android:onClick="Back" />

    <TextView
        android:text="@string/add_item"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:textColor="@color/black"
        android:id="@+id/AItitle"
        android:textAlignment="center"
        android:textSize="24sp"
        android:layout_below="@+id/AIarrow"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />

    <TextView
        android:text="@string/item_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/AItitle"
        android:layout_alignParentStart="true"
        android:layout_marginTop="16dp"
        android:textColor="@color/black"
        android:id="@+id/AIitemnameTV"
        android:textSize="18sp"
        android:textAlignment="viewStart"
        android:layout_alignEnd="@+id/AIitemnameET" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:id="@+id/AIitemnameET"
        android:textSize="14sp"
        android:hint="@string/insert_item_name"
        android:layout_below="@+id/AIitemnameTV"
        android:layout_alignParentStart="true" />

    <TextView
        android:text="@string/item_price"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/AIitempriceTV"
        android:layout_below="@+id/AIitemnameET"
        android:textColor="@color/black"
        android:layout_alignParentStart="true"
        android:textSize="18sp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:layout_below="@+id/AIitempriceTV"
        android:layout_alignParentStart="true"
        android:id="@+id/AIitempriceET"
        android:layout_alignEnd="@+id/AIitemnameET"
        android:textSize="14sp"
        android:hint="@string/insert_item_price" />

    <TextView
        android:text="@string/category"
        android:textColor="@color/black"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_below="@+id/AIitempriceET"
        android:layout_alignParentStart="true"
        android:id="@+id/AIcategoryTV"
        android:layout_alignEnd="@+id/AIitempriceET"
        android:textSize="18sp" />

    <Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="13dp"
        android:id="@+id/AIspinner"
        android:layout_below="@+id/AIcategoryTV"
        android:layout_alignParentStart="true" />

    <Button
        android:text="@string/enter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/AIspinner"
        android:layout_alignParentStart="true"
        android:layout_marginTop="22dp"
        android:id="@+id/AIbutton"
        android:layout_alignParentEnd="true"
        android:onClick="NewEntry" />


</RelativeLayout>
```

Edit Item Page

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:id="@+id/activity_edit_item"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.orderappvw.EditItem">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:id="@+id/EIarrow"
        android:onClick="Back" />

    <TextView
        android:text="@string/edit_item"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/EItitle"
```

```
        android:textColor="@color/black"
        android:textAlignment="center"
        android:textSize="24sp" />

    <Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/EIspinner" />

    <Button
        android:text="@string/enter"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/EIbutton"
        android:onClick="GetItems" />
</LinearLayout>
```

Edit Item 2 Page

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_edit_item2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.orderappvw.EditItem2">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:onClick="Back"
        android:id="@+id/EI2arrow" />

    <TextView
        android:text="@string/edit_item"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/EI2title"
        android:textColor="@color/black"
        android:textAlignment="center"
        android:textSize="24sp"
        android:layout_below="@+id/EI2arrow"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />

    <TextView
        android:text="@string/item_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:textColor="@color/black"
        android:layout_below="@+id/EI2title"
        android:layout_alignParentStart="true"
        android:layout_marginTop="16dp"
        android:id="@+id/EI2itemnameTV"
        android:textSize="18sp"
        android:textAlignment="viewStart"
        android:layout_alignEnd="@+id/EI2itemnameET" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:id="@+id/EI2itemnameET"
        android:textSize="14sp"
        android:hint="@string/insert_item_name"
        android:layout_below="@+id/EI2itemnameTV"
        android:layout_alignParentStart="true" />

    <TextView
        android:text="@string/item_price"
        android:layout_width="wrap_content"
        android:textColor="@color/black"
        android:layout_height="wrap_content"
        android:id="@+id/EI2itempriceTV"
        android:layout_below="@+id/EI2itemnameET"
        android:layout_alignParentStart="true"
        android:textSize="18sp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:layout_below="@+id/EI2itempriceTV"
        android:layout_alignParentStart="true"
        android:id="@+id/EI2itempriceET"
        android:layout_alignEnd="@+id/EI2itemnameET"
        android:textSize="14sp"
        android:hint="@string/insert_item_price" />

    <TextView
        android:text="@string/category"
        android:layout_width="wrap_content"
        android:textColor="@color/black"
        android:layout_height="wrap_content"
        android:layout_below="@+id/EI2itempriceET"
        android:layout_alignParentStart="true"
        android:id="@+id/EI2categoryTV"
        android:layout_alignEnd="@+id/EI2itempriceET"
        android:textSize="18sp" />

    <Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="13dp"
        android:id="@+id/EI2spinner"
        android:layout_below="@+id/EI2categoryTV"
```

```
        android:layout_alignParentStart="true" />

    <Button
        android:text="@string/enter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/EI2spinner"
        android:layout_alignParentStart="true"
        android:layout_marginTop="22dp"
        android:id="@+id/EI2button"
        android:layout_alignParentEnd="true"
        android:onClick="Update" />
</RelativeLayout>
```

Delete Item Page

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:id="@+id/activity_delete_item"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.orderappvw.DeleteItem">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:id="@+id/DIarrow"
        android:onClick="Back" />

    <TextView
        android:text="@string/delete_item"
        android:textColor="@color/black"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/DItitle"
        android:textAlignment="center"
        android:textSize="24sp" />

    <Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/DIspinner" />

    <Button
        android:text="@string/enter"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/DIenter"
```

```
        android:onClick="GetItems" />
</LinearLayout>
```

Notify All Page

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_notify_all"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.orderappvw.NotifyAll">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:id="@+id/NAarrow"
        android:onClick="Back" />

    <TextView
        android:text="@string/notify_all"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/NAarrow"
        android:id="@+id/NAtitle"
        android:textColor="@color/black"
        android:layout_alignParentEnd="true"
        android:layout_alignParentStart="true"
        android:textAlignment="center"
        android:textSize="24sp" />

    <TextView
        android:text="@string/enter_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:id="@+id/NAmessageTV"
        android:textSize="18sp"
        android:textColor="@color/black"
        android:textAlignment="center"
        android:layout_below="@+id/NAtitle"
        android:layout_alignParentStart="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:ems="10"
        android:layout_below="@+id/NAmessageTV"
```

```
        android:layout_alignParentStart="true"
        android:layout_marginTop="27dp"
        android:id="@+id/NAmessageET"
        android:layout_alignParentEnd="true"
        android:hint="Enter message..." />

    <Button
        android:text="@string/send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/NAmessageET"
        android:layout_alignParentStart="true"
        android:layout_marginTop="26dp"
        android:id="@+id/NAsend"
        android:layout_alignParentEnd="true"
        android:onClick="Send" />
</RelativeLayout>
```

Delete Accounts Page

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_delete_accounts"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.orderappvw.DeleteAccounts"
    android:weightSum="1">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:id="@+id/DAarrow"
        android:onClick="Back" />

    <TextView
        android:text="@string/delete_Accounts"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:id="@+id/VAtitle"
        android:textAlignment="center"
        android:textSize="24sp" />

    <Space
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0.04" />

    <TextView
```

```
        android:text="@string/processing"
        android:layout_width="match_parent"
        android:textColor="@color/black"
        android:layout_height="wrap_content"
        android:id="@+id/VAdescription"
        android:textAlignment="center"
        android:textSize="18sp" />
</LinearLayout>
```

Order App vW Code

Home Page

```
package com.nadramon.orderappvw;

import android.content.DialogInterface;
import android.content.Intent;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.view.ContextThemeWrapper;
import android.text.InputType;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class HomePage extends AppCompatActivity {

    DatabaseReference DReference = FirebaseDatabase.getInstance().getReference();
//Gets the database reference
    DatabaseReference DPending = DReference.child("Pending Orders"); //Gets the
reference for the Pending Orders field

    TextView LiveCounter; //Defining the variable type

    @Override
    public void onBackPressed() { //Prevents users from pressing the back button
on their phone
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home_page); //Creates the page with the
layout
        LiveCounter = (TextView) findViewById(R.id.LiveCount); //Gives the
variable type its XML reference
        DPending.addValueEventListener(new ValueEventListener() { //Check
database every time there is a change in the Pending Orders field
            //If there is a new order, or an order was completed, the code within
```

```
this method will run, regardless of which page is currently opened
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                Long count = dataSnapshot.getChildrenCount(); //Count how many
things are stored under the pending orders field
                String countString = Long.toString(count); //Convert the number
to a string
                String Message = "There are currently: " + countString + " orders
pending."; //Append the number to a sentence
                LiveCounter.setText(Message); //Set the text in the layout as the
sentence above
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });

    }

    //Procedure to send the user to the specified page
    public void ChangePage(Class page) {//Needing a class variable
        Intent Change = new Intent(this, page);
        startActivity(Change); //Open the designated page
        finish(); //Close the current page
    }

    //Procedure to get a password confirmation dialog box
    public void SummonDialogBox(final String password, final Class page)
{//Needing a string and a class variable
        AlertDialog.Builder dialogBox = new AlertDialog.Builder(new
ContextThemeWrapper(this, R.style.myDialog)); //gets the style for the dialog box
        dialogBox.setMessage("Please enter the password to access this
feature:"); //sets the message in the dialog box
        final EditText Confirmation = new EditText(this); //creates a new
EditText
        Confirmation.setInputType(InputType.TYPE_TEXT_VARIATION_PASSWORD); //sets
the input type to password so it will be hidden
        dialogBox.setView(Confirmation); //Adds the EditText to the dial
        dialogBox.setPositiveButton("ENTER", new
DialogInterface.OnClickListener() { //Upon clicking the ENTER button...
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String ConfirmPassword = Confirmation.getText().toString(); //Get
the string in the EditText
                if (password.equals(ConfirmPassword)) {//Compare the actual
password with the string from the EditText
                    dialog.dismiss();//Close dialog box
                    ChangePage(page); //Run the ChangePage procedure with
specified variable
                }
                else { //If it doesn't match
                    Toast.makeText(getApplicationContext(),"Password is
incorrect.", Toast.LENGTH_SHORT).show(); //Create a toast
                    dialog.dismiss(); //Close dialog box
                }
            }
        });
        dialogBox.setNegativeButton("CANCEL", new
DialogInterface.OnClickListener() {//Upong clicking the CANCEL button...
```

This achieves objective 1hi, 1kiv, 1miii

```
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();//Close the dialog box
            }
        });
        dialogBox.create().show(); //Create the dialog box and make it appear on
screen
    }

    //Upong clicking the Pending Orders button, run the ChangePage procedure with
specified variable
    public void toPO(View view) {
        ChangePage(PendingOrders.class);
    }
    //Upong clicking the View Menu button, run the ChangePage procedure with
specified variable
    public void toVM(View view) {
        ChangePage(ViewMenu.class);
    }
    //Upong clicking the Pending Orders button, run the SummonDialogBox procedure
    public void toEM(View view) {
        SummonDialogBox("edit123", EditMenu.class);
    }
    //Upong clicking the Pending Orders button, run the SummonDialogBox procedure
    public void toNA(View view) {
        SummonDialogBox("notify123", NotifyAll.class);
    }
    //Upong clicking the Pending Orders button, run the SummonDialogBox procedure
    public void toDA(View view) {
        SummonDialogBox("delete123", DeleteAccounts.class);
    }

    //If the "Exit App" button is pressed, completely shuts down the App
    public void Shutdown(View view) {
        finish(); //closes the page
        System.exit(0); //shuts app down
    }
}
```

This achieves objective 1ni

Pending Orders Page

```
package com.nadramon.orderappvw;

import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
```

```
import com.google.firebase.database.ValueEventListener;

public class PendingOrders extends AppCompatActivity {

    DatabaseReference DReference = FirebaseDatabase.getInstance().getReference();
//Gets the database reference
    DatabaseReference DPending = DReference.child("Pending Orders"); //Gets the
reference for the Pending Orders field

    //Defining variable types
    TextView liveCounter;
    LinearLayout Linearlayout;

    @Override
    public void onBackPressed() { //Prevents users from pressing the back button
on their phone
    }

    public void Back(View view) {//Upon pressing the arrow button...
        Intent Back = new Intent(this, HomePage.class);//Destination: Home Page
        startActivity(Back); //Open the new page
        finish(); //Close current page

    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pending_orders); //Creates the
the layout

        //Gives the variable types its reference in the XML file
        liveCounter = (TextView) findViewById(R.id.POlivecount);
        Linearlayout = (LinearLayout) findViewById(R.id.POlinearLayout);

        DPending.addValueEventListener(new ValueEventListener() {//Check database
every time there is a change in the Pending Orders field
            //If there is a new order, or an order was completed, the code within
this method will run, regardless of which page is currently opened
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                Long count = dataSnapshot.getChildrenCount();//Count how many
things are stored under the pending orders field
                String countString = Long.toString(count);//Convert the number to
a string
                String Message = "Total: " + countString; //Append the number to
a sentence
                liveCounter.setText(Message); //Set the text in the layout as the
sentence above

                int s = Linearlayout.getChildCount(); //Gets number of objects in
the layout
                if (s > 0) { //Because initially there nothing in the layout
                    //this is more for when the button is being pressed after the
first time
                    for (int x = 0; x < s; x++) {
                        //It will get rid of all the objects from previous button
click
                        Linearlayout.removeViewAt(0);
```

This achieves objective 1ji, 1jii, 1jiii

```
                    }
                }

                //For each child node under the pending order field
                for (final DataSnapshot postSnapshot: dataSnapshot.getChildren())
{
                    String username = (String)
postSnapshot.child("Username").getValue();
//Text objects are dynamically created

                    TextView Order = new TextView(PendingOrders.this); //Creating
a new view of TextView
                    String text = username + "'s order";
                    Order.setText(text); //Gives the view the text of the Item
                    Order.setTextSize(23); //Sets the font size to 23
                    Order.setTextColor(Color.BLACK); //Set the text colour to
black
                    Order.setClickable(true); //Allows it to be clicked on by
users
                    Linearlayout.addView(Order); //Adds the new view to the
layout

                    Order.setOnClickListener(new View.OnClickListener() {//Upon
clicking the text...
                        @Override
                        public void onClick(View v) {
                            String time = (String)
postSnapshot.child("Time").getValue(); //Get the time of the order from the
database
                            Intent Change = new Intent(PendingOrders.this,
ViewOrder.class);
                            Change.putExtra("time", time);//Pass this value to
the next page
                            startActivity(Change); //Open the next page
                            finish(); //Close this page
                        }
                    });

                    //Creates an image view
                    ImageView Divider = new ImageView(PendingOrders.this);
                    LinearLayout.LayoutParams div = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, 5);
                    div.setMargins(15, 15, 15, 15); //sets the margins
                    Divider.setLayoutParams(div);
                    Linearlayout.addView(Divider); //add it to the layout
                    //It's an empty image, just to create spacing between the
items
                }
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });
    }
}
```

View Order Page

```
package com.nadramon.orderappvw;

import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.os.Handler;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.view.ContextThemeWrapper;
import android.text.InputType;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TableRow.LayoutParams;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.security.MessageDigest;
import java.util.ArrayList;

public class ViewOrder extends AppCompatActivity {

    DatabaseReference DReference = FirebaseDatabase.getInstance().getReference();
//Get the database reference
    DatabaseReference DAccounts = DReference.child("Accounts"); //Get the
reference for the accounts field
    DatabaseReference DPending = DReference.child("Pending Orders"); //Get the
reference for the pending orders field
    DatabaseReference DorderID; //define variable type

    //Define variable types
    ArrayList<String> items;
    ArrayList<String> quantities;
    ArrayList<String> messages;
    String cost;
    String username;
    String time;
    String customerPassword;
    String customerName;
    String customerRegion;
    String customerStreet;
    String customerHouse;
    String text;

//Method for hashing password
    public static String sha256(String base) {
        try{
            MessageDigest digest = MessageDigest.getInstance("SHA-256"); //Get
```

```
the SHA-256 hash
            byte[] hash = digest.digest(base.getBytes("UTF-8")); //convert input
string to series of bytes
            StringBuffer hexString = new StringBuffer(); //make the string
modifiable

            for (int i = 0; i < hash.length; i++) { //for each set of bytes
                String hex = Integer.toHexString(0xff & hash[i]); //Convert the
bytes sequence to a hex string
                if (hex.length() == 1) hexString.append('0');
                hexString.append(hex);//add each one to one string

            }
            return hexString.toString(); //convert it back to a regular string
and return the value
        }
        catch (Exception e){
            throw new RuntimeException(e);
        }
    }

    //Procedure for sending notification
    private void SendNotification(String text) {
        DAccounts.child(username).child("Order Status").setValue(text);
    }


    //Procedure for making dialog box showing user details
    private void DetailsDialogBox(String name, String region, String street,
String house) { //Needing 4 string variables
        //create new dialog box
        AlertDialog.Builder dialogBox= new AlertDialog.Builder(new
ContextThemeWrapper(ViewOrder.this, R.style.myDialog));
        //Set the messae, \n will make it into a new line
        dialogBox.setMessage("Here are the details of the customer:" + "\nName: "
+ name + "\nRegion: " + region + "\nStreet Address: " + street + "\nHouse
Address: " + house);
        dialogBox.setPositiveButton("GOTCHA!", new
DialogInterface.OnClickListener() { //Set the GOTCHA button
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss(); //Close the dialog box
            }
        });
        dialogBox.create().show();//Create the dialog box and make it appear
    }

    //Procedure to make a dialog box that sends notifications
    private void NotificationDialogBox() {
        //create new dialog box
        AlertDialog.Builder dialogBox = new AlertDialog.Builder(new
ContextThemeWrapper(ViewOrder.this, R.style.myDialog));
        dialogBox.setTitle("What is the status of the order?"); //Set the title
message
        final EditText customMessage = new EditText(ViewOrder.this); //Make a new
edit text
        dialogBox.setItems(R.array.Processes, new
DialogInterface.OnClickListener() { //Set a clickable list in the dialog box
            @Override
```

```
            public void onClick(DialogInterface dialog, int which) {
                text = messages.get(which); //Get the clicked item
                if (text.equals(messages.get(4))) { //If the clicked item is the
custom message
                    text = customMessage.getText().toString(); //Get the string
from the edit text
                    if (text.equals("")) { //Error trap incase the user typed
nothing
                        Toast.makeText(getApplicationContext(), "You didn't type
a custom message!", Toast.LENGTH_SHORT).show(); //Error pop up
                    }
                    else {
                        SendNotification(text); //run send notification procedure
                    }
                }
                else {
                    SendNotification(text);//run send notification procedure
                }
                dialog.dismiss();//Close dialog box
            }
        });
        dialogBox.setView(customMessage); //Add the edit text to the dialog box
        dialogBox.setNegativeButton("CANCEL", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss(); //Close dialog box
            }
        });
        dialogBox.create().show(); //create dialog box and make it appear on
screen

    }

    //Procedure to get a password confirmation dialog box
    private void PasswordDialogBox(final String password) {
        AlertDialog.Builder dialogBox = new AlertDialog.Builder(new
ContextThemeWrapper(ViewOrder.this, R.style.myDialog));
        dialogBox.setMessage("Please input the customer's password."); //set
message
        final EditText confirmation = new EditText(ViewOrder.this); //make a new
edit text
        confirmation.setInputType(InputType.TYPE_TEXT_VARIATION_PASSWORD); //set
the input to a password so it will be hidden
        dialogBox.setView(confirmation); //Add the edit text to the dialog box
        dialogBox.setPositiveButton("ENTER", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String confirmPassword = confirmation.getText().toString(); //Get
the inputted value
                if (password.equals(sha256(confirmPassword))) {//Run hash
function on the inputted value and compare to actual value from database
                    //If the same
                    DorderID.removeValue(); //Remove the order from the pending
order field
                    Intent Change = new Intent(ViewOrder.this,
PendingOrders.class);
                    Toast.makeText(getApplicationContext(),"The order has been
```

This achieves objective 1ki, 1kii

```
completed!", Toast.LENGTH_SHORT).show();// success pop up
                    dialog.dismiss();//close dialog box
                    startActivity(Change); //Open new page
                    finish(); //Close current page
                }
                else {
                    Toast.makeText(getApplicationContext(),"Password is
incorrect", Toast.LENGTH_SHORT).show(); //Error pop up
                    dialog.dismiss(); //close dialog box
                }
            }
        });
        dialogBox.setNegativeButton("CANCEL", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();// close dialog box
            }
        });
        dialogBox.create().show(); //make dialog box appear
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_order); //create page with
specified layout

        Intent intent = getIntent(); //get values from previous page
        time = intent.getExtras().getString("time");

        DorderID = DPending.child(time); //get reference of the order id

        //New lists
        items = new ArrayList<>();
        quantities = new ArrayList<>();


        DorderID.addListenerForSingleValueEvent(new ValueEventListener() {//Check
the database just this once
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {

                Long itemCount = dataSnapshot.child("Item").getChildrenCount();
//get the number of records of the items

//Text objects and table rows are dynamically created

                //for each item
                for (int x = 0; x < itemCount; x++) {
                    String y = Integer.toString(x); //convert the number to
string
                    //Get the item and price from the database
                    String item = (String)
dataSnapshot.child("Item").child(y).getValue();
                    String quantity = (String)
dataSnapshot.child("Quantity").child(y).getValue();
                    //Add the item and price to its respective lists
                    items.add(item);
```

```
                quantities.add(quantity);
            }

            //Get the user name and total cost value
            username = (String) dataSnapshot.child("Username").getValue();
            cost = (String) dataSnapshot.child("Total Cost").getValue();

            //Create a table layout
            TableLayout tableLayout = (TableLayout)
findViewById(R.id.VOtableLayout);
            //Set the parameters
            LayoutParams textLayout = new
LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT, 1);

            //Create the first row
            TableRow FirstRow = new TableRow(ViewOrder.this);
            //Set layout parameters
            FirstRow.setLayoutParams(new
LayoutParams(LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
            FirstRow.setWeightSum(2); //Make it have two columns
            FirstRow.setOrientation(TableRow.VERTICAL); //So that it will be
for columns

            //New textviews
            TextView ItemTitle = new TextView(ViewOrder.this);
            TextView QuantityTitle = new TextView(ViewOrder.this);

            //Set the text, size, color, layout parameters, and alignment
            ItemTitle.setText("Item");
            ItemTitle.setTextSize(18);
            ItemTitle.setTextColor(Color.BLACK);
            ItemTitle.setLayoutParams(textLayout);
            ItemTitle.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
            ItemTitle.setRight(1);

            //Set the text, size, color, layout parameters, and alignment
            QuantityTitle.setText("Quantity");
            QuantityTitle.setTextSize(18);
            QuantityTitle.setTextColor(Color.BLACK);
            QuantityTitle.setLayoutParams(textLayout);
            QuantityTitle.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
            QuantityTitle.setRight(1);

            //add the columns to the row
            FirstRow.addView(ItemTitle);
            FirstRow.addView(QuantityTitle);
            tableLayout.addView(FirstRow); //add the row to the table layout

            int listLength = items.size(); //Get the size of the item list

            //For each item in the list
            for (int x = 0; x < listLength; x++) {

                //Same process as above
                TableRow rows = new TableRow(ViewOrder.this);
                rows.setLayoutParams(new
LayoutParams(LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
                rows.setWeightSum(2);
                rows.setOrientation(TableRow.VERTICAL);
```

This achieves objective 1jiv

```
                    TextView itemTV = new TextView(ViewOrder.this);
                    TextView qtyTV = new TextView(ViewOrder.this);

                    itemTV.setText(items.get(x));
                    itemTV.setTextSize(14);
                    itemTV.setTextColor(Color.BLACK);
                    itemTV.setLayoutParams(textLayout);
                    itemTV.setTextAlignment(View.TEXT_ALIGNMENT_TEXT_START);
                    itemTV.setRight(1);

                    qtyTV.setText(quantities.get(x));
                    qtyTV.setTextSize(14);
                    qtyTV.setTextColor(Color.BLACK);
                    qtyTV.setLayoutParams(textLayout);
                    qtyTV.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
                    qtyTV.setRight(1);

                    rows.addView(itemTV);
                    rows.addView(qtyTV);
                    tableLayout.addView(rows);
                }

                //Same process as above, for the last row
                TableRow lastRow = new TableRow(ViewOrder.this);
                lastRow.setLayoutParams(new
LayoutParams(LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
                lastRow.setWeightSum(1); //Only one column though
                lastRow.setOrientation(TableRow.VERTICAL);

                TextView costTV = new TextView(ViewOrder.this);

                String totalCost = "Total Cost: " + cost + " Rials";

                costTV.setText(totalCost);
                costTV.setTextSize(14);
                costTV.setTextColor(Color.BLACK);
                costTV.setTextAlignment(View.TEXT_ALIGNMENT_TEXT_END);
                costTV.setLayoutParams(textLayout);
                costTV.setRight(1);

                lastRow.addView(costTV);
                tableLayout.addView(lastRow);
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });
        //Messages for notification
        messages = new ArrayList<>();
        messages.add("Your order has been viewed.");
        messages.add("Your order is currently being cooked.");
        messages.add("Your order is currently being delivered.");
        messages.add("Your order has arrived at the designated address!");
        messages.add("Custom Message:");
    }


    public void UsersDetails(View view) {
```

```
        String usernameLowered = username.toLowerCase();
        DAccounts.child(usernameLowered).addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                //get the user's details values from database
                customerName = dataSnapshot.child("Name").getValue().toString();
                customerRegion =
dataSnapshot.child("Region").getValue().toString();
                customerStreet = dataSnapshot.child("Street
Address").getValue().toString();
                customerHouse = dataSnapshot.child("House
Address").getValue().toString();

                //Run the details procedure
                DetailsDialogBox(customerName, customerRegion, customerStreet,
customerHouse);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });
    }

    //Run the notification dialog box procedure upon clicking the notify button
    public void Notify(View view) {
        NotificationDialogBox();
    }

    public void Completed(View view) {
        String usernameLowered = username.toLowerCase(); //make sure username is
lowercased
        DAccounts.child(usernameLowered).addListenerForSingleValueEvent(new
ValueEventListener() {//Check the database just this once
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                customerPassword =
dataSnapshot.child("Password").getValue().toString(); //get password from
database
                PasswordDialogBox(customerPassword); //run password dialog
procedure
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });
    }

    public void Back(View view) {
        Intent Back = new Intent(this, PendingOrders.class);
        startActivity(Back);//open new page
        finish(); //close current page
    }
}
```

This achieves objective 1hv, 1li

This achieves objective 1hvi

This achieves objective 1hvii

View Menu Page

```
package com.nadramon.orderappvw;

import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

public class ViewMenu extends AppCompatActivity {


    DatabaseReference DReference = FirebaseDatabase.getInstance().getReference();
//Get the database reference
    DatabaseReference DMenu = DReference.child("Menu"); //Gets the reference for
the menu field

    //Define variable types
    Spinner Categories;
    ArrayList<String> items;
    ArrayList<String> prices;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_menu);//Creates the page with the
layout

        //Creating the spinner list of item categories
        Categories = (Spinner) findViewById(R.id.VMspinner);
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.Categories,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        Categories.setAdapter(adapter);



    }

    public void GetItems(View view) {//Upon clicking the ENTER button
        //Set the list to be empty (for more than one clicks)
        items = new ArrayList<>();
```

This achieves objective 1ii

```
        prices = new ArrayList<>();

        final String ChosenCategory = Categories.getSelectedItem().toString();
//Gets the selected item in the spinner

        final TableLayout tableLayout = (TableLayout)
findViewById(R.id.VMtableLayout);
        int s = tableLayout.getChildCount(); //Gets number of Views in the layout
        if (s > 0) { //Because initially there is nothing in the table
            tableLayout.removeAllViews(); //Removes all views
        }

        DMenu.addListenerForSingleValueEvent(new ValueEventListener() {//Check
the database just this once
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {

                Long categoryLength =
dataSnapshot.child("Item").child(ChosenCategory).getChildrenCount(); //Get the
number of things there are in the chosen category field
                for (int x = 0; x < categoryLength; x++) {//For every record
under that field
                    String y = Integer.toString(x); //Convert the number to a
string for the ID
                    String item =
dataSnapshot.child("Item").child(ChosenCategory).child(y).getValue().toString();
//Get the name of the item
                    String price =
dataSnapshot.child("Price").child(ChosenCategory).child(y).getValue().toString();
//Get the price of the item

                    items.add(item); //Add the item to the item list
                    prices.add(price); //Add the price to the price list
                }

                TableRow.LayoutParams textLayout = new
TableRow.LayoutParams(TableRow.LayoutParams.FILL_PARENT,
TableRow.LayoutParams.WRAP_CONTENT, 1);//Set parameters

                TableRow FirstRow = new TableRow(ViewMenu.this); //Create a new
view of a table row
                FirstRow.setLayoutParams(new
TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT,
TableRow.LayoutParams.WRAP_CONTENT)); //Set layout parameters
                FirstRow.setWeightSum(2); //Set the columns to 2
                FirstRow.setOrientation(TableRow.VERTICAL); // Vertical so that
it is columns

                TextView ItemTitle = new TextView(ViewMenu.this); //New textview
for item heading
                TextView PriceTitle = new TextView(ViewMenu.this); //New textview
for price heading

                ItemTitle.setText(R.string.item); //Set the text for the textview
                ItemTitle.setTextSize(18); //Set the font size
                ItemTitle.setTextColor(Color.BLACK); //Set the colour of the text
                ItemTitle.setLayoutParams(textLayout); //Set the paramaters
                ItemTitle.setTextAlignment(View.TEXT_ALIGNMENT_CENTER); //Set the
alignment
```

This achieves objective 1iii

This achieves objective 1iiii

```
                ItemTitle.setRight(1);

                PriceTitle.setText(R.string.price); //Set the text for the
textview
                PriceTitle.setTextSize(18); //Set the font size
                PriceTitle.setTextColor(Color.BLACK); //Set the colour of the
text
                PriceTitle.setLayoutParams(textLayout); //Set the paramaters
                PriceTitle.setTextAlignment(View.TEXT_ALIGNMENT_CENTER); //Set
the alignment
                PriceTitle.setRight(1);

                FirstRow.addView(ItemTitle); //Add the textview to the first
column
                FirstRow.addView(PriceTitle); //Add the textview to the second
column
                tableLayout.addView(FirstRow); //Add the row to the table

                int listLength = items.size(); //Get the number of items in the
list

                for (int x = 0; x < listLength; x++) { //For each item

                    TableRow rows = new TableRow(ViewMenu.this);//Create a new
view of a table row
                    rows.setLayoutParams(new
TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT,
TableRow.LayoutParams.WRAP_CONTENT));//Set layout parameters
                    rows.setWeightSum(2); //Set the columns to 2
                    rows.setOrientation(TableRow.VERTICAL); // Vertical so that
it is columns

                    TextView itemTV = new TextView(ViewMenu.this); //New textview
for item
                    TextView priceTV = new TextView(ViewMenu.this);//New textview
for price


                    itemTV.setText(items.get(x));//Get the item name from the
list
                    itemTV.setTextSize(14);//Set the font size
                    itemTV.setTextColor(Color.BLACK);//Set the colour of the text
                    itemTV.setLayoutParams(textLayout);//Set the paramaters
                    itemTV.setTextAlignment(View.TEXT_ALIGNMENT_TEXT_START);//Set
the alignment
                    itemTV.setRight(1);

                    priceTV.setText(prices.get(x));//Get the item price from the
list
                    priceTV.setTextSize(14);//Set the font size
                    priceTV.setTextColor(Color.BLACK);//Set the colour of the
text
                    priceTV.setLayoutParams(textLayout);//Set the paramaters
                    priceTV.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);//Set
the alignment
                    priceTV.setRight(1);

                    rows.addView(itemTV);//Add the textview to the first column
                    rows.addView(priceTV);//Add the textview to the second column
                    tableLayout.addView(rows);//Add the row to the table
```

```
                }
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });
    }
    @Override
    public void onBackPressed() {//Prevents users from pressing the back button
on their phone

    }

    public void Back(View view) {//Upon pressing the arrow button
        Intent Back = new Intent(this, HomePage.class);
        startActivity(Back);//Open the new page
        finish(); //Close current page
    }
}
```

Edit Menu Page

```
package com.nadramon.orderappvw;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class EditMenu extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_menu);
    }

    //Procedure to send the user to the specified page
    public void ChangePage(Class page) {//Needing a class variable
        Intent Change = new Intent(this, page);
        startActivity(Change); //Open the new page
        finish(); //Close the current page
    }

    //Upong clicking the Add Item button, run the ChangePage procedure with
specified variable
    public void AddItem(View view) {
        ChangePage(AddItem.class);
    }
    //Upong clicking the Edit Item button, run the ChangePage procedure with
specified variable
    public void EditItem(View view) {
        ChangePage(EditItem.class);
    }
```

```
    //Upong clicking the Delete Item button, run the ChangePage procedure with
specified variable
    public void DeleteItem(View view) {
        ChangePage(DeleteItem.class);
    }
    //Upong clicking the arrow button, run the ChangePage procedure with
specified variable
    public void Back(View view) {
        ChangePage(HomePage.class);
    }

    @Override
    public void onBackPressed() {//Prevents users from pressing the back button
on their phone
    }
}
```

Add Item Page

This achieves objective 1hiv

```
package com.nadramon.orderappvw;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class AddItem extends AppCompatActivity {

    DatabaseReference DReference =
FirebaseDatabase.getInstance().getReference();//Gets the database reference
    DatabaseReference DMenu = DReference.child("Menu"); //Gets the reference for
the menu field
    Spinner Categories; //Sets variable type


    //Procedure to send the user to the specified page
    public void ChangePage(Class page) {//Needing a class variable
        Intent Change = new Intent(this, page);
        startActivity(Change); //Open the new page
        finish();//Close current page
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_item); //Create the page with the
layout

        //Creating the spinner list of item categories
        Categories = (Spinner) findViewById(R.id.AIspinner);
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.Categories,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        Categories.setAdapter(adapter);
    }

    public void NewEntry(View view) {//Upon pressing the ENTER button...

        EditText itemNameET = (EditText) findViewById(R.id.AIitemnameET);//Gives
the variable type its XML reference
        EditText itemPriceET = (EditText)
findViewById(R.id.AIitempriceET);//Gives the variable type its XML reference

        final String itemName = itemNameET.getText().toString(); //Get the
inputted value
        final String itemPrice = itemPriceET.getText().toString();//Get the
inputted value
        final String chosenCategory = Categories.getSelectedItem().toString();
//Gets the selected item in the spinner

        if (itemName.equals("") || itemPrice.equals("")) {//Error check, if the
user didn't type anything
            Toast.makeText(getApplicationContext(), "A field was left empty!",
Toast.LENGTH_SHORT).show(); //Pop an error message
        }
        else {//Otherwise...
            DMenu.addListenerForSingleValueEvent(new ValueEventListener()
{//Check the database just this once
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    Long counter =
dataSnapshot.child("Item").child(chosenCategory).getChildrenCount(); //Get the
number of things under the item field
                    String counterS = Long.toString(counter); //Convert the
number to a string for the ID

DMenu.child("Item").child(chosenCategory).child(counterS).setValue(itemName);
//Add the new item to the database

DMenu.child("Price").child(chosenCategory).child(counterS).setValue(itemPrice);//
Add the new price to the database
                    Toast.makeText(getApplicationContext(), "The item has been
added!", Toast.LENGTH_SHORT).show();//Pop a success message
                    ChangePage(AddItem.class);//Refresh the page
                }
                @Override
                public void onCancelled(DatabaseError databaseError) {
                }
            });
        }
```

```
    }

    //Run the ChangePage procedure upon clicking the arrow button with the
specified variable
    public void Back(View view) {
        ChangePage(EditMenu.class);
    }
}
```

Edit Item Page

```
package com.nadramon.orderappvw;

import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TextView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class EditItem extends AppCompatActivity {

    DatabaseReference DReference =
FirebaseDatabase.getInstance().getReference();//Gets the database reference
    DatabaseReference DMenu = DReference.child("Menu");//Gets the reference for
the menu field

    Spinner Categories;//Sets variable type

    @Override
    public void onBackPressed() { //Prevents users from pressing the back button
on their phone
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_item);//Create the page with the
layout

        //Creating the spinner list of item categories
        Categories = (Spinner) findViewById(R.id.EIspinner);
        ArrayAdapter<CharSequence> adapter =
```

```
ArrayAdapter.createFromResource(this, R.array.Categories,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        Categories.setAdapter(adapter);
    }

    public void GetItems(View view) {

        final LinearLayout linearLayout = (LinearLayout)
findViewById(R.id.activity_edit_item);//Gives the variable type its XML reference
        int s = linearLayout.getChildCount(); //Gets number of Views in the
layout
        if (s > 4) { //Because initially there is the back button, text, spinner
and button
            //this is more for when the button is being pressed after the first
time
            for (int x = 4; x < s; x++) {
                //It will get rid of the views after the 4 views
                linearLayout.removeViewAt(4);
            }
        }

        final String ChosenCategory = Categories.getSelectedItem().toString();
//Gets the selected item in the spinner
        DMenu.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                Long counter =
dataSnapshot.child("Item").child(ChosenCategory).getChildrenCount(); //Gets
number of items of that category
                for (int x = 0; x < counter; x++) {
                    final String y = Integer.toString(x);//Convert the number to
a string for the ID
                    final String item = (String)
dataSnapshot.child("Item").child(ChosenCategory).child(y).getValue();//Get the
item from the database
                    final String price = (String)
dataSnapshot.child("Price").child(ChosenCategory).child(y).getValue();//Get the
price from the database
                    TextView Items = new TextView(EditItem.this); //Creating a
new view of TextView
                    Items.setText(item); //Gives the view the text of the Item
                    Items.setTextSize(23); //Sets the font size to 23
                    Items.setTextColor(Color.BLACK); //Set the text colour to
black
                    Items.setClickable(true); //Allows it to be clicked on by
users
                    linearLayout.addView(Items); //Adds the new view to the
layout

                    Items.setOnClickListener(new View.OnClickListener() {//Upon
clicking the item...
                        @Override
                        public void onClick(View v) {
                            Intent Change = new Intent(EditItem.this,
EditItem2.class);
                            //Pass these values to the next page
                            Change.putExtra("Item", item);
```

```
                            Change.putExtra("Price", price);
                            Change.putExtra("ID", y);
                            Change.putExtra("Category", ChosenCategory);
                            startActivity(Change);//Open the new page
                            finish();//Close current page
                        }
                    });

                    //Creates an image view
                    ImageView divider = new ImageView(EditItem.this);
                    LinearLayout.LayoutParams div = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, 5);
                    div.setMargins(15, 15, 15, 15); //sets the margins
                    divider.setLayoutParams(div);
                    linearLayout.addView(divider); //add it to the layout
                    //It's an empty image, just to create spacing between the
items
                }
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });

    }
    public void Back(View view) {//Upon clicking the arrow button
        Intent Change = new Intent(this, EditMenu.class);
        startActivity(Change);//Open the new page
        finish();//Close current page
    }
}
```

Edit Item 2 Page

This achieves objective 1hii

```
package com.nadramon.orderappvw;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class EditItem2 extends AppCompatActivity {

    DatabaseReference DReference =
```

```
FirebaseDatabase.getInstance().getReference();//Gets the reference of the
database
    DatabaseReference DMenu = DReference.child("Menu");//Gets the reference for
the menu field

    //Defines variable types
    String Item;
    String Price;
    String Category;
    String ID;
    int IDinteger;
    Spinner Categories;
    EditText itemET;
    EditText priceET;

    @Override
    public void onBackPressed() { //Prevents users from pressing the back button
on their phone
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_item2); //Creates the page with the
specified layout

        Intent intent = getIntent();
        //Retrieves values from previous activity
        Item = intent.getExtras().getString("Item");
        Price = intent.getExtras().getString("Price");
        Category = intent.getExtras().getString("Category");
        ID = intent.getExtras().getString("ID");

        IDinteger = Integer.parseInt(ID);//Convert string to integer

        //Give the variables its XML reference
        itemET = (EditText) findViewById(R.id.EI2itemnameET);
        priceET = (EditText) findViewById(R.id.EI2itempriceET);

        //Creating the spinner list of item categories
        Categories = (Spinner) findViewById(R.id.EI2spinner);
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.Categories,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        Categories.setAdapter(adapter);

        int position = adapter.getPosition(Category); //Get the position of the
item category

        //Set the text of the editText and spinner
        itemET.setText(Item);
        priceET.setText(Price);
        Categories.setSelection(position);
    }

    //Procedure for changing pages
    public void ChangePage(Class page) {//Needing a class variable
```

```
        Intent Change = new Intent(this, page);
        startActivity(Change);//Open the new page
        finish();//Close the current page
    }

    public void Update(View view) {//Upon clicking the update button

        //Get the values of the edit text and spinner
        final String item = itemET.getText().toString();
        final String price = priceET.getText().toString();
        final String category = Categories.getSelectedItem().toString();

        if (!category.equals(Category)) {//If the category is different

            DMenu.addListenerForSingleValueEvent(new ValueEventListener()
{//Check the database just this once
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {

                    if (item.equals("") || price.equals("")) {//Error check, if
the user didn't type anything
                        Toast.makeText(getApplicationContext(), "A field was left
empty!", Toast.LENGTH_SHORT).show(); //Pop an error message
                    }

                    else {//Otherwise...
                        Long counter =
dataSnapshot.child("Item").child(category).getChildrenCount(); //Get the number
of records in that category of items
                        String counterS = Long.toString(counter); //Convert that
number to a string

DMenu.child("Item").child(category).child(counterS).setValue(item);//Set the new
item with the ID from the counter

DMenu.child("Price").child(category).child(counterS).setValue(price);//Set the
new price with the ID from the counter


DMenu.child("Item").child(Category).child(ID).removeValue();//Remove the item
from its original position

DMenu.child("Price").child(Category).child(ID).removeValue();//Remove the price
from its original position

                        Long counter2 =
dataSnapshot.child("Item").child(Category).getChildrenCount(); //Get the number
of records in the old category

                        for (int x = IDinteger + 1; x < counter2; x++) { //For
each records there that was after the original item

                            String y = Integer.toString(x); //Convert number to
string
                            //Get the item and price values
                            String itemDown =
dataSnapshot.child("Item").child(Category).child(y).getValue().toString();
                            String priceDown =
dataSnapshot.child("Price").child(Category).child(y).getValue().toString();
```

```
                            //Minus the number by one to get a new ID
                            int z = x - 1;
                            String w = Integer.toString(z);//Convert that to
string

                            //Move the item and price down by 1 ID

DMenu.child("Item").child(Category).child(w).setValue(itemDown);

DMenu.child("Price").child(Category).child(w).setValue(priceDown);

                            //For the last item
                            if (x == counter2-1) {
                                //Delete the item and price

DMenu.child("Item").child(Category).child(y).removeValue();

DMenu.child("Price").child(Category).child(y).removeValue();
                            }
                        }
                        Toast.makeText(getApplicationContext(), "The item has
been updated!", Toast.LENGTH_SHORT).show(); //Popup for success
                        ChangePage(EditItem.class);//Run the change page
procedure with the specified variable
                    }
                }
                @Override
                public void onCancelled(DatabaseError databaseError) {
                }
            });
        }
        else {//If category is the same

            if (item.equals("") || price.equals("")) {//Error check, if the user
didn't type anything
                Toast.makeText(getApplicationContext(), "A field was left
empty!", Toast.LENGTH_SHORT).show(); //Pop an error message
            }
            else {
                //Simply update the values
                DMenu.child("Item").child(category).child(ID).setValue(item);
                DMenu.child("Price").child(category).child(ID).setValue(price);
                Toast.makeText(getApplicationContext(), "The item has been
updated!", Toast.LENGTH_SHORT).show();//Pop a success message
                ChangePage(EditItem.class);//Run the change page procedure with
the specified variable
            }
        }
    }
    //Run the change page procedure with the specified variable
    public void Back(View view) {
        ChangePage(EditItem.class);
    }
}
```

Delete Item Page

This achieves objective 1hiii

```
package com.nadramon.orderappvw;

import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.view.ContextThemeWrapper;
import android.text.InputType;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class DeleteItem extends AppCompatActivity {

    DatabaseReference DReference =
FirebaseDatabase.getInstance().getReference();//Get the database reference
    DatabaseReference DMenu = DReference.child("Menu"); //Get the reference for
the menu field
    Spinner Categories; //Sets variable type

    //Procedure to send the user to the specified page
    public void ChangePage(Class page) { //Needing a class variable
        Intent Change = new Intent(this, page);
        startActivity(Change);//Open the new page
        finish();//Close the current page
    }

    @Override
    public void onBackPressed() { //Prevents users from pressing the back button
on their phone
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_delete_item);//Create the page with the
specified layout

        //Creating the spinner list of item categories
        Categories = (Spinner) findViewById(R.id.DIspinner);
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.Categories,
```

```
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        Categories.setAdapter(adapter);
    }


    public void GetItems(View view) {//Upon pressing the ENTER button...

        final LinearLayout linearLayout = (LinearLayout)
findViewById(R.id.activity_delete_item);
        int viewCount = linearLayout.getChildCount(); //Gets number of Views in
the layout
        if (viewCount > 4) { //Because initially there is the back button, text,
spinner and button
            //this is more for when the button is being pressed after the first
time
            for (int x = 4; x < viewCount; x++) {
                //It will get rid of the views after the 4 views
                linearLayout.removeViewAt(4);
            }
        }

        final String ChosenCategory = Categories.getSelectedItem().toString();
//Gets the selected item in the spinner
        DMenu.addListenerForSingleValueEvent(new ValueEventListener() {//Check
the database just this once
            @Override
            public void onDataChange(final DataSnapshot dataSnapshot) {
                final Long counter =
dataSnapshot.child("Item").child(ChosenCategory).getChildrenCount(); //Gets
number of items of that category
                for (int x = 0; x < counter; x++) {
                    final String y = Integer.toString(x); //convert the number to
string
                    final String item = (String)
dataSnapshot.child("Item").child(ChosenCategory).child(y).getValue();//Get the
value from database
                    TextView Items = new TextView(DeleteItem.this); //Creating a
new view of TextView
                    Items.setText(item); //Gives the view the text of the Item
                    Items.setTextSize(23); //Sets the font size to 23
                    Items.setTextColor(Color.BLACK); //Set the text colour to
black
                    Items.setClickable(true); //Allows it to be clicked on by
users
                    linearLayout.addView(Items); //Adds the new view to the
layout

                    final int currentID = x; //Current ID for items below the
current item

                    Items.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) { //Upon clicking the text...

                            //make a new alert dialog box
                            AlertDialog.Builder dialogBox = new
AlertDialog.Builder(new ContextThemeWrapper(DeleteItem.this, R.style.myDialog));
```

```
                            dialogBox.setMessage("Are you sure you wish to delete
this item from the menu?"); //set the message
                            dialogBox.setPositiveButton("YES", new
DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int
which) {//Set the YES button

                                    //Remove the intended item and price

DMenu.child("Item").child(ChosenCategory).child(y).removeValue();

DMenu.child("Price").child(ChosenCategory).child(y).removeValue();

                                    //For each item and price under the intended
ones
                                    for (int z = currentID + 1; z < counter; z++
) {

                                        String currentIDs = Integer.toString(z);
//Convert number to string
                                        //Get the current values
                                        String itemDown =
dataSnapshot.child("Item").child(ChosenCategory).child(currentIDs).getValue().toS
tring();
                                        String priceDown =
dataSnapshot.child("Price").child(ChosenCategory).child(currentIDs).getValue().to
String();

                                        //Minus the number by one to get a new ID
                                        int newID = z - 1;
                                        String newIDs = Integer.toString(newID);
//Convert the number to string

                                        //Move the ID for the values down

DMenu.child("Item").child(ChosenCategory).child(newIDs).setValue(itemDown);

DMenu.child("Price").child(ChosenCategory).child(newIDs).setValue(priceDown);

                                        //For the last item
                                        if (z == counter-1) {
                                            //Remove item and price

DMenu.child("Item").child(ChosenCategory).child(currentIDs).removeValue();

DMenu.child("Price").child(ChosenCategory).child(currentIDs).removeValue();
                                        }
                                    }
                                    dialog.dismiss(); //Close dialog box
                                    ChangePage(DeleteItem.class); //Run the
Change Page procedure with the specified variable

                                }
                            });
                            dialogBox.setNegativeButton("NO", new
DialogInterface.OnClickListener() {//Set the NO button
                                @Override
                                public void onClick(DialogInterface dialog, int
```

```
which) {
                                        dialog.dismiss();//Close the dialog
                                    }
                                });
                                dialogBox.create().show();//Create the dialog box and
make it appear
                            }
                        });

                        //Creates an image view
                        ImageView divider = new ImageView(DeleteItem.this);
                        LinearLayout.LayoutParams div = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, 5);
                        div.setMargins(15, 15, 15, 15); //sets the margins
                        divider.setLayoutParams(div);
                        linearLayout.addView(divider); //add it to the layout
                        //It's an empty image, just to create spacing between the
items
                    }
                }
                @Override
                public void onCancelled(DatabaseError databaseError) {
                }
            });
    }
    //Run the changepage procedure with the specified variable
    public void Back(View view) {
        ChangePage(EditMenu.class);
    }
}
```

Notify All Page

This achieves objective 1kiii

```
package com.nadramon.orderappvw;

import android.content.Intent;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class NotifyAll extends AppCompatActivity {

    //Gets the database reference
    DatabaseReference DReference = FirebaseDatabase.getInstance().getReference();

    //Define variable type
    EditText editText;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_notify_all);//Create page with the
specified layout

        editText = (EditText) findViewById(R.id.NAmessageET); //Give the variable
it's XML reference

    }

    //Procedure to send the user to the specified page
    public void ChangePage(Class page) {//Needing a class variable
        Intent Change = new Intent(this, page);
        startActivity(Change);//Open the new page
        finish();//Close current page
    }

    public void Send(View view) {//Upon pressing the SEND button
       String message = editText.getText().toString();//Get the value from the
edittext
        if (message.equals("")) {//Error trap incase they typed nothing
            Toast.makeText(getApplicationContext(), "You didn't type anything!",
Toast.LENGTH_SHORT).show();//error popup
        }
        else {//Otherwise...
            DReference.child("Public Notification").setValue(message);//Get
reference for public notification field
            ChangePage(HomePage.class);//Run the changepage procedure with the
specified variable
        }
    }
    @Override
    public void onBackPressed() { //Prevents users from pressing the back button
on their phone
    }
    //Run the changepage procedure with the specified variable
    public void Back(View view) {
        ChangePage(HomePage.class);
    }
}
```

Delete Accounts Page

This achieves objective 1mi

```
package com.nadramon.orderappvw;

import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.view.ContextThemeWrapper;
import android.view.View;
```

```
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.text.SimpleDateFormat;
import java.util.Date;

public class DeleteAccounts extends AppCompatActivity {

    DatabaseReference DReference =
FirebaseDatabase.getInstance().getReference();//Get the database reference
    DatabaseReference DOrderHistory = DReference.child("Order History"); //Get
the reference for the order history field
    DatabaseReference DAccounts = DReference.child("Accounts"); //Get the
reference for the accounts field

    //Procedure to send the user to the specified page
    public void ChangePage(Class page) { //Needing a class variable
        Intent Change = new Intent(this, page);
        startActivity(Change);//Open the new page
        finish();//Close the current page
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_delete_accounts);//Create page with the
specified layout

        DOrderHistory.addListenerForSingleValueEvent(new ValueEventListener()
{//Check the database just this once
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {

                Date now = new Date(); //Get the current date
                int counter = 0; //Set counter to 0

                SimpleDateFormat JustYear = new SimpleDateFormat("yyyy"); //Get
just the year
                SimpleDateFormat JustMonth = new SimpleDateFormat("MM"); //Get
just the month
                SimpleDateFormat JustDay = new SimpleDateFormat("dd"); //Get just
the day
                //Convert them to string
                String CurrentYear = JustYear.format(now);
                String CurrentMonth = JustMonth.format(now);
                String CurrentDay = JustDay.format(now);
                //Convert them to integer
                int CurrentYearINT = Integer.parseInt(CurrentYear);
                int CurrentMonthINT = Integer.parseInt(CurrentMonth);
                int CurrentDayINT = Integer.parseInt(CurrentDay);
```

```
                LinearLayout linearLayout = (LinearLayout)
findViewById(R.id.activity_delete_accounts); //Gets the XML reference for the
layout
                TextView message = (TextView) findViewById(R.id.VAdescription);
//Gets the XML reference for the view

                //For every single order history
                for (DataSnapshot snapshot: dataSnapshot.getChildren()) {
                    Long orderCounter = snapshot.getChildrenCount(); //Get the
number of orders for each account
                    String orderCounterS = Long.toString(orderCounter); //Convert
that to string
                    String orderName = "Order " + orderCounterS; //To get the
last order
                    String date =
snapshot.child(orderName).child("Date").getValue().toString(); //Get the date of
the last order
                    String year = date.substring(6,10); //Get just the year
                    String month = date.substring(3,5); //Get just the month
                    String day = date.substring(0,2); //Get just the day
                    //Convert them to integer values
                    int yearINT = Integer.parseInt(year);
                    int monthINT = Integer.parseInt(month);
                    int dayINT = Integer.parseInt(day);

                    if (yearINT == CurrentYearINT) {
                        //pass
                    }
                    else if (CurrentYearINT == yearINT+1 && CurrentMonthINT >
monthINT) {
                        //pass
                    }
                    else if (CurrentYearINT == yearINT+1 && CurrentMonthINT ==
monthINT && CurrentDayINT > dayINT) {
                        //pass
                    }
                    else { //If the order is more than one year old from the
current date

                        final String username = snapshot.getKey(); //Get the
username
                        TextView Accounts = new TextView(DeleteAccounts.this);
//Creating a new view of TextView
                        Accounts.setText(username); //Gives the view the text of
the Item
                        Accounts.setTextSize(23); //Sets the font size to 23
                        Accounts.setTextColor(Color.BLACK); //Sets the font
colour to black
                        Accounts.setClickable(true); //Allows it to be clicked on
by users
                        linearLayout.addView(Accounts); //Adds the new view to
the layout


                        Accounts.setOnClickListener(new View.OnClickListener() {
                            @Override
                            public void onClick(View v) {//Upon clicking the text
```

```
                                //Create a new dialog box
                                AlertDialog.Builder dialogBox = new
AlertDialog.Builder(new ContextThemeWrapper(DeleteAccounts.this,
R.style.myDialog));
                                dialogBox.setMessage("Are you sure you wish to
completely remove this account from the database?"); //Set the message
                                dialogBox.setPositiveButton("CONFIRM", new
DialogInterface.OnClickListener() {
                                    @Override
                                    public void onClick(DialogInterface dialog,
int which) {//Set the CONFIRM button

                                        //If the user clicks the CONFIRM button

DOrderHistory.child(username).removeValue(); //Remove all traces of their order
history
                                        DAccounts.child(username).removeValue();
//Remove all traces of their account details
                                        dialog.dismiss();//Close dialog box
                                        //Success pop up
                                        Toast.makeText(getApplicationContext(),
"The account has been successfully removed.", Toast.LENGTH_SHORT).show();
                                        ChangePage(DeleteAccounts.class); //Run
Changepage procedure with the specified variable
                                    }
                                });
                                dialogBox.setNegativeButton("CANCEL", new
DialogInterface.OnClickListener() {//set the CANCEL button
                                    @Override
                                    public void onClick(DialogInterface dialog,
int which) {
                                        dialog.dismiss();//close dialog box
                                    }
                                });
                                dialogBox.create().show(); //Create the dialog
box and make it appear
                            }
                        });

                        //Creates an image view
                        ImageView divider = new ImageView(DeleteAccounts.this);
                        LinearLayout.LayoutParams div = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, 5);
                        div.setMargins(15, 15, 15, 15); //sets the margins
                        divider.setLayoutParams(div);
                        linearLayout.addView(divider); //add it to the layout
                        //It's an empty image, just to create spacing between the
items

                        counter ++; //Increment counter
                    }
                }
                if (counter == 0) { //If no accounts were 1 year old
                    message.setText(R.string.no_inactive_accounts); //Set text
                }
                else { //Otherwise
                    String counterS = Integer.toString(counter); //Convert
counter to string
                    message.setText("There are " + counterS + " accounts eligible
```

This achieves objective 1mii

```
for removal."); //Set text
                    }
                }
                @Override
                public void onCancelled(DatabaseError databaseError) {

                }
            });
    }

    @Override
    public void onBackPressed() { //Prevents users from pressing the back button
on their phone
    }

    public void Back(View view) {
        ChangePage(HomePage.class);
    }//Run change page procedure with specified variable
}
```

Order App vC Resources

Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.nadramon.test92">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:configChanges="orientation"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".Main2Activity"
            android:configChanges="orientation"
```

```
            android:screenOrientation="portrait" />
        <activity
            android:name=".signupcomplete"
            android:configChanges="orientation"
            android:screenOrientation="portrait" />
        <activity
            android:name=".homepagev2"
            android:configChanges="orientation"
            android:screenOrientation="portrait" />
        <activity
            android:name=".MakeAnOrder"
            android:configChanges="orientation"
            android:screenOrientation="portrait" />
        <activity
            android:name=".OrderHistory"
            android:configChanges="orientation"
            android:screenOrientation="portrait" />
        <activity android:name=".CheckOrder"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity android:name=".AfterOrderTransition"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity android:name=".ViewHistory"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
        <activity android:name=".EditAccount"
            android:configChanges="orientation"
            android:screenOrientation="portrait"/>
    </application>

</manifest>
```

Colours

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#228B22</color>
    <color name="colorPrimaryDark">#008000</color>
    <color name="colorAccent">#FF4081</color>
    <color name="titleColor">#FFFF00</color>
    <color name="black">#000000</color>
</resources>
```

Strings

```
<resources>
    <string name="app_name">South-East Asian Coffeeshop vC</string>
    //Log In and Sign Up screen
    <string name="insert_username">Insert Username</string>
    <string name="insert_password">Insert Password</string>
    <string name="insert_name">Insert Name</string>
    <string name="insert_street_address">Insert Street Address</string>
    <string name="insert_house_address">Insert House Address</string>
    <string name="select_region">Select Region</string>
```

```
<string name="name">Name</string>
<string name="username">Username</string>
<string name="password">Password</string>
<string name="region">Region</string>
<string name="street_address">Street Address</string>
<string name="house_address">House Address</string>
<string name="sign_up">Sign Up</string>
<string name="log_in">Log In</string>
<string name="or">Or</string>
<string name="azaiba">Azaiba</string>
<string name="ghubrah">Ghubrah</string>
<string name="make_order">Make an Order!</string>
<string name="check_order">Check Order</string>
<string name="edit_account">Edit Account</string>
<string name="view_history">View Order History</string>
<string name="exit_app">Exit App</string>
<string name="update_details">Update</string>
<string name="order_sent">Order Sent!</string>
<string name="return">Return</string>
<string name="search">Search</string>
<string name="sign_up_complete">Sign Up Complete!</string>
<string name="selected_orders">Selected Orders</string>
<string name="login_success">Logging in Success!</string>

<string name="continue_login">Continue to Log In!</string>
<string name="continue_home">Continue to Home!</string>

<string name="item">Item</string>
<string name="qty"> Qty </string>
<string name="price">Price</string>
<string name="total">Total</string>

<string name="restart_order">Restart Order</string>
<string name="confirm_order">Confirm Order</string>

<string name="enter">Enter</string>
<string-array name="Categories">
    <item>Starters</item>
    <item>Soup</item>
    <item>Beef</item>
    <item>Chicken</item>
    <item>Duck</item>
    <item>Fish</item>
    <item>Vegetables</item>
    <item>Rice</item>
    <item>Noodles</item>
    <item>Desert</item>
    <item>Beverages</item>
</string-array>

<string-array name="months">
    <item>January</item>
    <item>February</item>
    <item>March</item>
    <item>April</item>
    <item>May</item>
    <item>June</item>
    <item>July</item>
    <item>August</item>
```

```
        <item>September</item>
        <item>October</item>
        <item>November</item>
        <item>December</item>
    </string-array>
</resources>
```

Styles

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <style name="myDialog" parent="Theme.AppCompat.Dialog">
        <item name="android:windowNoTitle">true</item>
    </style>

</resources>
```

Order App vC Gradle Scripts

Build.Gradle (Project: test92) [[[[#I need to change name of project to OrderAppvC]]]]

```
// Top-level build file where you can add configuration options common to all
sub-projects/modules.

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.2.0'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
        classpath 'com.google.gms:google-services:3.0.0'
    }
}

allprojects {
    repositories {
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Build.Gradle (Module: App)

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion "24.0.3"
    defaultConfig {
        applicationId "com.nadramon.test92"
        minSdkVersion 17
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
    productFlavors {
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:24.2.1'
    compile 'com.google.firebase:firebase-database:9.6.0'
    testCompile 'junit:junit:4.12'
}


apply plugin: 'com.google.gms.google-service
```

Order App vC Layout

Log In Page

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
```

```
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context="com.nadramon.test92.MainActivity">

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPersonName"
    android:hint="@string/insert_username"
    android:ems="10"
    android:layout_below="@+id/LIusernameTV"
    android:layout_alignLeft="@+id/LIusernameTV"
    android:layout_alignStart="@+id/LIusernameTV"
    android:id="@+id/LIusernameET"
    android:textSize="14sp"
    android:layout_alignRight="@+id/LItitle"
    android:layout_alignEnd="@+id/LItitle" />

<TextView
    android:text="@string/username"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/LIusernameTV"
    android:textSize="18sp"
    android:textColor="@color/black"
    android:layout_below="@+id/LItitle"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<TextView
    android:text="@string/password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/LIusernameET"
    android:layout_alignParentLeft="true"
    android:textColor="@color/black"
    android:layout_alignParentStart="true"
    android:id="@+id/LIpasswordTV"
    android:textSize="18sp" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:ems="10"
    android:id="@+id/LIpasswordET"
    android:layout_below="@+id/LIpasswordTV"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:hint="@string/insert_password"
    android:textSize="14sp"
    android:layout_alignRight="@+id/LIusernameET"
    android:layout_alignEnd="@+id/LIusernameET" />

<TextView
    android:text="@string/log_in"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/LItitle"
        android:textSize="24sp"
        android:textColor="@color/black"
        android:layout_marginTop="38dp"
        android:layout_below="@+id/textView2"
        android:layout_centerHorizontal="true" />

    <Button
        android:text="@string/log_in"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/LIlogin"
        android:onClick="LogIn"
        android:layout_centerVertical="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <TextView
        android:text="@string/or"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="39dp"
        android:textColor="@color/black"
        android:id="@+id/LIor"
        android:textAlignment="center"
        android:layout_below="@+id/LIlogin"
        android:layout_centerHorizontal="true" />

    <TextView
        android:text="TextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="invisible"
        android:id="@+id/textView2"
        android:layout_alignParentTop="true"
        android:textColor="@color/black"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <Button
        android:text="@string/sign_up"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/LIsignup"
        android:onClick="SignUp"
        android:layout_marginTop="34dp"
        android:layout_below="@+id/LIor"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true" />

</RelativeLayout>
```

Sign Up Page

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/scrollView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/activity_main"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        tools:context="com.nadramon.test92.Main2Activity">

        <TextView
            android:text="@string/username"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/SUnameET"
            android:layout_alignParentLeft="true"
            android:textColor="@color/black"
            android:layout_alignParentStart="true"
            android:id="@+id/SUusernameTV"
            android:textSize="18sp" />

        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:inputType="textPersonName"
            android:hint="@string/insert_username"
            android:ems="10"
            android:id="@+id/SUusernameET"
            android:textSize="14sp"
            android:layout_below="@+id/SUusernameTV"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true" />

        <TextView
            android:text="@string/password"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/SUusernameET"
            android:layout_alignParentLeft="true"
            android:textColor="@color/black"
            android:layout_alignParentStart="true"
            android:id="@+id/SUpasswordTV"
            android:textSize="18sp" />


        <TextView
            android:text="@string/select_region"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/SUpasswordET"
            android:layout_alignParentLeft="true"
```

```
        android:layout_alignParentStart="true"
        android:textColor="@color/black"
        android:id="@+id/SUregionradio"
        android:textSize="18sp" />

    <RadioGroup
        android:id="@+id/regiongroup"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:orientation="horizontal"
        android:layout_below="@+id/SUregionradio"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true">
        <RadioButton
            android:text="@string/azaiba"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/SUregionradio"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true"
            android:onClick="onRadioButtonClicked"
            android:id="@+id/Azaiba" />

        <RadioButton
            android:text="@string/ghubrah"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/SUregionradio"
            android:layout_toRightOf="@+id/Azaiba"
            android:layout_toEndOf="@+id/Azaiba"
            android:onClick="onRadioButtonClicked"
            android:id="@+id/Ghubrah" />
    </RadioGroup>

    <TextView
        android:text="@string/street_address"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/regiongroup"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:textColor="@color/black"
        android:id="@+id/SUstreetTV"
        android:textSize="18sp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:hint="@string/insert_street_address"
        android:ems="10"
        android:id="@+id/SUstreetET"
        android:textSize="14sp"
        android:layout_below="@+id/SUstreetTV"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <TextView
        android:text="@string/house_address"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/SUstreetET"
        android:textColor="@color/black"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:id="@+id/SUhouseTV"
        android:textSize="18sp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:hint="@string/insert_house_address"
        android:ems="10"
        android:id="@+id/SUhouseET"
        android:textSize="14sp"
        android:layout_below="@+id/SUhouseTV"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:hint="@string/insert_password"
        android:textColor="@color/black"
        android:ems="10"
        android:id="@+id/SUpasswordET"
        android:textSize="14sp"
        android:layout_below="@+id/SUpasswordTV"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:fontFamily="sans-serif" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:hint="@string/insert_name"
        android:ems="10"
        android:id="@+id/SUnameET"
        android:textSize="14sp"
        android:layout_below="@+id/SUnameTV"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@android:drawable/ic_menu_info_details"
        android:id="@+id/SUinfo1"
        android:onClick="PasswordHelp"
        android:layout_below="@+id/SUusernameET"
        android:layout_toRightOf="@+id/SUregionradio"
        android:layout_toEndOf="@+id/SUregionradio"
        android:layout_marginTop="11dp"
        android:clickable="true" />
```

This achieves objective 1cvi

```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@android:drawable/ic_menu_info_details"
    android:id="@+id/SUinfo2"
    android:onClick="RegionHelp"
    android:layout_centerVertical="true"
    android:layout_toRightOf="@+id/SUpasswordET"
    android:layout_toEndOf="@+id/SUpasswordET"
    android:clickable="true" />

<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@android:drawable/ic_menu_info_details"
    android:id="@+id/SUinfo3"
    android:layout_below="@+id/regiongroup"
    android:layout_toRightOf="@+id/SUinfo1"
    android:layout_toEndOf="@+id/SUinfo1"
    android:clickable="true"
    android:onClick="StreetHelp"
    android:layout_marginTop="10dp" />

<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:srcCompat="@android:drawable/ic_menu_info_details"
    android:id="@+id/SUinfo4"
    android:layout_below="@+id/SUstreetET"
    android:layout_alignLeft="@+id/SUinfo3"
    android:layout_alignStart="@+id/SUinfo3"
    android:clickable="true"
    android:onClick="HouseHelp"
    android:layout_marginTop="11dp" />

<TextView
    android:text="@string/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/SUnameTV"
    android:textSize="18sp"
    android:textColor="@color/black"
    android:layout_marginTop="10dp"
    android:layout_below="@+id/SUtitle"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<Button
    android:text="@string/sign_up"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/SUSignUp"
    android:onClick="b"
    android:layout_below="@+id/SUhouseET"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:textSize="14sp"
```

This achieves objective 1cv

```
            android:fontFamily="sans-serif" />

        <TextView
            android:text="TextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBottom="@+id/SUnameET"
            android:layout_alignParentRight="true"
            android:layout_alignParentEnd="true"
            android:visibility="invisible"
            android:id="@+id/debugger" />

        <TextView
            android:text="@string/sign_up"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/SUtitle"
            android:textAlignment="center"
            android:textSize="24sp"
            android:layout_alignParentTop="true"
            android:layout_alignParentStart="true"
            android:layout_marginTop="41dp"
            android:textColor="@color/black"
            android:layout_alignParentEnd="true"
            android:fontFamily="sans-serif" />

        <ImageButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:srcCompat="?attr/actionModeCloseDrawable"
            android:onClick="Back"
            android:id="@+id/SUBack"
            android:textSize="18sp"
            android:layout_alignParentTop="true"
            android:layout_alignParentStart="true" />

    </RelativeLayout>

</ScrollView>
```

Sign Up Complete Page

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_signupcomplete"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.test92.signupcomplete">

    <TextView
        android:text="@string/sign_up_complete"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_marginLeft="130dp"
        android:layout_marginTop="156dp"
        android:textColor="@color/black"
        android:id="@+id/SUPmessage"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <Button
        android:text="@string/return"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="25dp"
        android:onClick="Success"
        android:id="@+id/SUPreturn"
        android:visibility="invisible"
        android:layout_below="@+id/SUPmessage"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

Home Page

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_homepagev2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.test92.homepagev2">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:textColor="@color/black"
        android:layout_alignParentStart="true"
        android:id="@+id/HPWelcome"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:textAlignment="center"
        android:textSize="24sp" />

    <Button
        android:text="@string/make_order"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/HPMakeOrder"
        android:visibility="invisible"
        android:onClick="MakeOrder"
        android:layout_marginTop="72dp"
        android:layout_below="@+id/HPWelcome"
```

```
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <Button
        android:text="@string/view_history"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/HPViewHistory"
        android:visibility="invisible"
        android:onClick="ViewHistory"
        android:layout_below="@+id/HPEditAccount"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginTop="22dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <Button
        android:text="@string/exit_app"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/HPExitApp"
        android:visibility="invisible"
        android:layout_below="@+id/HPViewHistory"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginTop="23dp"
        android:layout_alignParentRight="true"
        android:onClick="ShutDown"
        android:layout_alignParentEnd="true" />

    <Button
        android:text="@string/edit_account"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/HPEditAccount"
        android:visibility="invisible"
        android:onClick="EditAcc"
        android:layout_marginTop="24dp"
        android:layout_below="@+id/HPMakeOrder"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <TextView
        android:text="@string/login_success"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/HPStarter"
        android:textColor="@color/black"
        android:textSize="36sp"
        android:textAlignment="center"
        android:layout_alignTop="@+id/HPEditAccount"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
```

```
        android:layout_alignParentEnd="true" />

</RelativeLayout>
```

Make an Order Page

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:id="@+id/activity_make_an_order"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.test92.MakeAnOrder"
    android:weightSum="1">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:id="@+id/MOBack"
        android:elevation="0dp"
        android:onClick="Back" />

    <TextView
        android:text="@string/make_order"
        android:textColor="@color/black"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/MOtitle"
        android:textAlignment="center"
        android:textSize="24sp" />

    <Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/MOspinner"
        android:layout_weight="0.05" />

    <Button
        android:text="@string/enter"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/MOenter"
        android:onClick="GetCategory"
        android:layout_weight="0.00" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="193dp">

        <LinearLayout
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/forItems"
            android:orientation="vertical" />
    </ScrollView>

    <Button
        android:text="@string/check_order"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/MOcheck"
        android:onClick="CheckOrder" />

</LinearLayout>
```

Check Order Page

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:id="@+id/activity_check_order"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.test92.CheckOrder"
    android:weightSum="1">


    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:id="@+id/COarrow"
        android:elevation="0dp"
        android:onClick="Back" />

    <TextView
        android:text="@string/selected_orders"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/COtitle"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:textSize="24sp" />

    <TableLayout
        android:id="@+id/COTableLayoutID"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >
    </TableLayout>
```

```
</LinearLayout>
```

After Order Complete Page

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_after_order_transition"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.test92.AfterOrderTransition">

    <TextView
        android:text="@string/order_sent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="130dp"
        android:layout_marginTop="156dp"
        android:id="@+id/AOTmessage"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textColor="@color/black" />

    <Button
        android:text="@string/return"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="27dp"
        android:onClick="Success"
        android:id="@+id/AOTenter"
        android:visibility="invisible"
        android:layout_below="@+id/AOTmessage"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Edit Account Page

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_edit_account"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
```

```
tools:context="com.nadramon.test92.EditAccount">


<TextView
    android:text="@string/name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/EATitle"
    android:layout_alignParentStart="true"
    android:id="@+id/EATName"
    android:textColor="@color/black"
    android:textSize="18sp" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPersonName"
    android:ems="10"
    android:layout_below="@+id/EATName"
    android:layout_alignParentStart="true"
    android:id="@+id/EAEName"
    android:textSize="14sp" />

<TextView
    android:text="@string/password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/EAEName"
    android:textColor="@color/black"
    android:layout_alignParentStart="true"
    android:id="@+id/EATPassword"
    android:textSize="18sp" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:ems="10"
    android:layout_below="@+id/EATPassword"
    android:layout_alignParentStart="true"
    android:id="@+id/EAEPassword"
    android:textSize="14sp" />

<TextView
    android:text="@string/region"
    android:layout_width="wrap_content"
    android:textColor="@color/black"
    android:layout_height="wrap_content"
    android:layout_below="@+id/EAEPassword"
    android:layout_alignParentStart="true"
    android:id="@+id/EATRegion"
    android:textSize="18sp" />

<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:id="@+id/EARgroup"
    android:layout_below="@+id/EATRegion"
```

```
        android:layout_alignParentStart="true"
        android:layout_marginTop="10dp">

        <RadioButton
            android:text="@string/azaiba"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="AzaibaB"
            android:id="@+id/EARAzaiba"
            android:layout_weight="1" />

        <RadioButton
            android:text="@string/ghubrah"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="GhubrahB"
            android:id="@+id/EARGhubrah"
            android:layout_weight="1" />
    </RadioGroup>

    <TextView
        android:text="@string/street_address"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:id="@+id/EATStreet"
        android:textSize="18sp"
        android:layout_below="@+id/EARgroup"
        android:layout_alignParentStart="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:layout_below="@+id/EATStreet"
        android:layout_alignParentStart="true"
        android:id="@+id/EAEStreet"
        android:textSize="14sp" />

    <TextView
        android:text="@string/house_address"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/EAEStreet"
        android:textColor="@color/black"
        android:layout_alignParentStart="true"
        android:id="@+id/EATHouse"
        android:textSize="18sp" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:layout_below="@+id/EATHouse"
        android:layout_alignParentStart="true"
        android:id="@+id/EAEHouse"
        android:textSize="14sp" />
```

```
    <Button
        android:text="Update"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/EAEHouse"
        android:layout_alignParentStart="true"
        android:layout_marginTop="18dp"
        android:id="@+id/EAUpdateDetails"
        android:onClick="Change"
        android:layout_alignParentEnd="true" />

    <TextView
        android:text="@string/edit_account"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/EATitle"
        android:textSize="24sp"
        android:textAlignment="center"
        android:layout_alignParentTop="true"
        android:textColor="@color/black"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="39dp" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:onClick="EABack"
        android:id="@+id/EABBack"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

Order History Page

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_order_history"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.test92.OrderHistory">

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:onClick="Back"
        android:id="@+id/OHBack" />
```

```
    <TextView
        android:text="@string/view_history"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:id="@+id/OHtitle"
        android:layout_below="@+id/OHBack"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:textAlignment="center"
        android:textSize="24sp" />

    <Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/OHspinner"
        android:layout_below="@+id/OHtitle"
        android:layout_alignParentStart="true"
        android:layout_toStartOf="@+id/OHsearch" />

    <Spinner
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/OHsearch"
        android:layout_alignParentStart="true"
        android:id="@+id/OHspinner2"
        android:layout_toStartOf="@+id/OHsearch" />

    <Button
        android:text="@string/search"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/OHsearch"
        android:onClick="GetHistory"
        android:layout_below="@+id/OHtitle"
        android:layout_alignParentEnd="true" />

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentStart="true"
        android:weightSum="1"
        android:id="@+id/linearLayout"
        android:layout_below="@+id/OHsearch">

    </LinearLayout>

</RelativeLayout>
```

View History Page

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:orientation="vertical"
    android:id="@+id/activity_check_order"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nadramon.test92.ViewHistory"
    android:weightSum="1">


    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="?attr/actionModeCloseDrawable"
        android:id="@+id/VHBack"
        android:elevation="0dp"
        android:onClick="Back" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/VHsentence"
        android:textColor="@color/black"
        android:textAlignment="center"
        android:textSize="24sp" />

    <TableLayout
        android:id="@+id/TableLayoutID2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >
    </TableLayout>

</LinearLayout>
```

Order App vC Code

Log In Page

```
package com.nadramon.test92;
//LOG IN ACTIVITY
//Import stuff
import android.app.Notification;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.support.v4.app.NotificationManagerCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.app.NotificationCompat;
import android.view.View;
import android.widget.EditText;
```

```
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.security.MessageDigest;


public class MainActivity extends AppCompatActivity {

    DatabaseReference DReference = FirebaseDatabase.getInstance().getReference();
//get the database reference
    DatabaseReference DAccounts = DReference.child("Accounts"); //get the
reference for the accounts field

    //Defining data types for variables
    TextView textView;
    Context Popup;
    CharSequence LogInError;
    int PopupDuration;
    EditText Username;
    EditText Password;
    String username;
    String password;
    int First;
    String PasswordHash;

    //When this activity is launched
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main); //make page with the specified
layout

        //Giving the variables a reference to the xml file
        textView = (TextView) findViewById(R.id.textView2);
        Username = (EditText) findViewById(R.id.LIusernameET);
        Password = (EditText) findViewById(R.id.LIpasswordET);

        //For Popup
        Popup = getApplicationContext();
        LogInError = "The username or password is incorrect.";
        PopupDuration = Toast.LENGTH_SHORT;

        //If logging in
        First = 0;
    }

    //If they click the sign up button
    public void SignUp(View view) {
        Intent intent = new Intent(this, Main2Activity.class); //selecting which
activity to go to
        startActivity(intent); //Starts the sign up activity
        finish(); //close current activity
    }
```

```
//Method for hashing password
    public static String sha256(String base) {
        try{
            MessageDigest digest = MessageDigest.getInstance("SHA-256"); //Get
the SHA-256 hash
            byte[] hash = digest.digest(base.getBytes("UTF-8")); //convert input
string to series of bytes
            StringBuffer hexString = new StringBuffer(); //make the string
modifiable

            for (int i = 0; i < hash.length; i++) { //for each set of bytes
                String hex = Integer.toHexString(0xff & hash[i]); //Convert the
bytes sequence to a hex string
                if (hex.length() == 1) hexString.append('0');
                hexString.append(hex);//add each one to one string

            }
            return hexString.toString(); //convert it back to a regular string
and return the value
        }
        catch (Exception e){
            throw new RuntimeException(e);
        }
    }

    @Override
    public void onBackPressed() { //To prevent users from pressing the back
button on their phone

    }


    //If they click log in button
    public void LogIn(final View view) {
        //Gets the user inputs and converted it to strings
        username = Username.getText().toString();
        final String usernameLowered = username.toLowerCase();
        password = Password.getText().toString();

        PasswordHash = sha256(password); //hashes the input password

        //Reading the Database Values for the Accounts field
        DAccounts.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) { //Getting the
Database Values for the Accounts field
                Long count =
dataSnapshot.child(usernameLowered).getChildrenCount(); //Gets number of values
is under the account field
                if (count > 1){ //Error trapping, an existing user should have
more than 1 thing under them
                    String PasswordCheck = (String)
dataSnapshot.child(usernameLowered).child("Password").getValue(); //Gets a hashed
password from the account data
                    if (PasswordCheck.equals(PasswordHash)) {
                        //Gets the rest of the values
                        String NameCheck = (String)
dataSnapshot.child(usernameLowered).child("Name").getValue();
```

This achieves objective 3dii

```
                        String RegionCheck = (String)
dataSnapshot.child(usernameLowered).child("Region").getValue();
                        String StreetCheck = (String)
dataSnapshot.child(usernameLowered).child("Street Address").getValue();
                        String HouseCheck = (String)
dataSnapshot.child(usernameLowered).child("House Address").getValue();

                        Intent intent = new Intent(view.getContext(),
homepagev2.class); //selecting which activity to go to

                        //Sends the values of user data to the next activity
                        intent.putExtra("Name", NameCheck);
                        intent.putExtra("Username", usernameLowered);
                        intent.putExtra("Password", PasswordCheck);
                        intent.putExtra("Region", RegionCheck);
                        intent.putExtra("Street", StreetCheck);
                        intent.putExtra("House", HouseCheck);
                        intent.putExtra("First", First);
                        startActivity(intent); //Starts the homepage activity
                        finish(); //Closes this activity
                    }
                    else {
                        Toast.makeText(Popup, LogInError, PopupDuration).show();
                    }
                }
                else { //No username
                    Toast.makeText(Popup, LogInError, PopupDuration).show();
                }
            }
            //If an error occurs when trying to read database values
            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
    }

    //Procedure to receive notifications
    private void GetNotification(String message) { //needs a string variable
        NotificationCompat.Builder builder = new
NotificationCompat.Builder(this); //notification builder
        builder.setSmallIcon(R.drawable.ic_notification_alert); //set the icon
        builder.setContentTitle("UPDATE!"); //set the title of the notification
        builder.setContentText(message); //set the message
        builder.setVibrate(new long[] {1000L, 1000L, 1000L,}); //set the length
of the vibration
        builder.setLights(Color.RED, 3000, 3000); //set the flashing lights on
phone
        Notification notification = builder.build(); //build the notification
        NotificationManagerCompat.from(this).notify(0, notification); //play the
notification
    }

    @Override
    protected void onStart() {
        super.onStart();

        DReference.addValueEventListener(new ValueEventListener() {//read the
database just this once
```

```
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                //get the value from the public notification field
                String message = (String) dataSnapshot.child("Public
Notification").getValue();
                if (message != null) {//if the field isn't empty
                    GetNotification(message); //Run the getnotification procedure
                }
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });
    }
}
```

Sign Up Page

```
package com.nadramon.test92;
//SIGN UP ACTIVITY
//Import stuffs
import android.content.Context;
import android.content.Intent;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.security.MessageDigest;

public class Main2Activity extends AppCompatActivity {

    DatabaseReference DReference =
FirebaseDatabase.getInstance().getReference();//get the database reference
    DatabaseReference DAccounts = DReference.child("Accounts"); //get the
reference for the accounts field


    //Defines variable data type
    EditText Name;
    EditText Username;
    EditText Password;
    EditText Street;
    EditText House;
    Button SignUp;
    RadioButton Region;
```

```
    String region;
    Context Popup;
    CharSequence ErrorMessage;
    CharSequence Taken;
    CharSequence PasswordHelpM;
    CharSequence RegionHelpM;
    CharSequence StreetHelpM;
    CharSequence HouseHelpM;
    int PopupDuration;
    String name;
    String username;
    String password;
    String street;
    String house;
    DatabaseReference DID;
    DatabaseReference DName;
    DatabaseReference DUserName;
    DatabaseReference DPassword;
    DatabaseReference DRegion;
    DatabaseReference DStreet;
    DatabaseReference DHouse;
    TextView debugger;
    Long count;
    String PasswordHash;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2); //Creates this activity's page
with the specified layout


        //Giving the variable a reference to the xml file
        Name = (EditText) findViewById(R.id.SUnameET);
        Username = (EditText) findViewById(R.id.SUusernameET);
        Password = (EditText) findViewById(R.id.SUpasswordET);
        Street = (EditText) findViewById(R.id.SUstreetET);
        House = (EditText) findViewById(R.id.SUhouseET);
        SignUp = (Button) findViewById(R.id.SUSignUp);

        debugger = (TextView) findViewById(R.id.debugger);

        //setting the messages
        Popup = getApplicationContext();
        ErrorMessage = "You have left a field blank!";
        Taken = "That Username is already in use.";
        RegionHelpM = "We cannot deliver outside of Azaiba and Ghubrah.";
        PasswordHelpM = "Password must be at least 6 characters.";
        StreetHelpM = "For Example: Way 4618, 12th Road, Natih Street";
        HouseHelpM = "For Example: Villa 1100A, House 3A, Al Noor Gardens
Complex";
        PopupDuration = Toast.LENGTH_SHORT;


        //intially set region to equal a value
        region = "";
```

```
    }

    //when the user clicks on the radio buttons
    public void onRadioButtonClicked(View view) {
        boolean checked = ((RadioButton) view).isChecked(); //checks that an
option is chosen
        switch(view.getId()) {
            case R.id.Azaiba: //if the selected one has an id called "Azaiba"
                if (checked) {
                    Region = (RadioButton) findViewById(R.id.Azaiba);
                    region = Region.getText().toString();
                }
                break;
            case R.id.Ghubrah:
                if (checked) {
                    Region = (RadioButton) findViewById(R.id.Ghubrah);
                    region = Region.getText().toString();
                }
                break;
        }
    }

    //Takes users back to the previous page, the Log In page
    public void Back(View view) {
        Intent Back = new Intent(this, MainActivity.class);
        startActivity(Back);//opens new page
        finish(); //closes current page
    }

    //If the info buttons are pressed, pop up a message:
    public void RegionHelp(View view) {
        Toast.makeText(Popup, RegionHelpM, PopupDuration).show();
    }
    public void PasswordHelp(View view) {
        Toast.makeText(Popup, PasswordHelpM, PopupDuration).show();
    }
    public void StreetHelp(View view) {
        Toast.makeText(Popup, StreetHelpM, PopupDuration).show();
    }
    public void HouseHelp(View view) {
        Toast.makeText(Popup, HouseHelpM, PopupDuration).show();
    }

//Method for hashing password

    public static String sha256(String base) {
        try{
            MessageDigest digest = MessageDigest.getInstance("SHA-256"); //Get
the SHA-256 hash
            byte[] hash = digest.digest(base.getBytes("UTF-8")); //convert input
string to series of bytes
            StringBuffer hexString = new StringBuffer(); //make the string
modifiable

            for (int i = 0; i < hash.length; i++) { //for each set of bytes
                String hex = Integer.toHexString(0xff & hash[i]); //Convert the
bytes sequence to a hex string
```

This achieves objective 3di

```
                if (hex.length() == 1) hexString.append('0');
                hexString.append(hex);//add each one to one string

            }
            return hexString.toString(); //convert it back to a regular string
and return the value
        }
        catch (Exception e){
            throw new RuntimeException(e);
        }
    }


    @Override
    public void onBackPressed() {//To prevent users from pressing the back button
on their phone
    }



    @Override
    public void onStart() {
        super.onStart();


        //when the sign up button is pressed
        SignUp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(final View v) {
                //get the inputted values
                name = Name.getText().toString();
                username = Username.getText().toString();
                password = Password.getText().toString();
                street = Street.getText().toString();
                house = House.getText().toString();

                String usernameLowered = username.toLowerCase(); //convert the
username to all lowercase

                PasswordHash = sha256(password); //run the hash function on the
password value

                //Get references for the database fields
                DID = DAccounts.child(usernameLowered);
                DName = DID.child("Name");
                DUserName = DID.child("Username");
                DPassword = DID.child("Password");
                DRegion = DID.child("Region");
                DStreet = DID.child("Street Address");
                DHouse = DID.child("House Address");


                DID.addValueEventListener(new ValueEventListener() { //Read the
database just this once
                    @Override
                    public void onDataChange(DataSnapshot dataSnapshot) {
                        count = dataSnapshot.getChildrenCount(); //Get the number
of records that is under the username
                        if (count > 1) { //an account should at least have 1
```

This achieves objective 1ci, 1ciii, 1civ

This achieves objective 1cix

This achieves objective 1cx

```
record, hence the username is already taken
                            debugger.setText("Block");
                        }
                        else { //Othewise, username not taken
                            debugger.setText("Pass");
                        }
                    }

                    @Override
                    public void onCancelled(DatabaseError databaseError) {

                    }
                });

                Handler timer = new Handler(); //new handler
                timer.postDelayed(new Runnable() { //Wait 2 seconds
                    @Override
                    public void run() {
                        String text = debugger.getText().toString(); //get the
text for taken/not taken username
                        int lengthPassword = password.length(); //get the length
of the password


                        //Makes sure no field is left blank
                        if (name.equals("") || username.equals("") ||
password.equals("") || region.equals("") || street.equals("") ||
house.equals("")) {
                            Toast.makeText(Popup, ErrorMessage,
PopupDuration).show();
                        }
                        //Makes sure the account doesn't already exist
                        else if (text.equals("Block")) {
                            Toast.makeText(Popup, Taken, PopupDuration).show();
                        }
                        //Makes sure the password is at a suitable length
                        else if (lengthPassword < 6) {
                            Toast.makeText(Popup, "Your password is too short.",
PopupDuration).show();
                        }
                        else {
                            //Writes to database, the values under the fields

                            DName.setValue(name);
                            DUserName.setValue(username);
                            DPassword.setValue(PasswordHash);
                            DRegion.setValue(region);
                            DStreet.setValue(street);
                            DHouse.setValue(house);

                            Intent intent = new
Intent(v.getContext(),signupcomplete.class);
                            startActivity(intent); //open new page
                            finish(); //close current page
                        }
                    }
                }, 2000);
            }
        });
```

This achieves objective 1cviii

This achieves objective 1cii

This achieves objective 1vii

```
    }
}
```

Sign Up Complete Page

```
package com.nadramon.test92;

import android.content.Intent;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class signupcomplete extends AppCompatActivity {

    //define variable types
    Handler timer;
    Button button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_signupcomplete); //create page with the
specified layout

        timer = new Handler(); // new handler
        timer.postDelayed(new Runnable() { //wait for 1 second
            @Override
            public void run() {
                button = (Button) findViewById(R.id.SUPreturn); //Get XML
reference of the button
                button.setText(R.string.continue_login); //Set the text
                button.setVisibility(View.VISIBLE); //Make it visible
            }
        }, 1000);
    }

    public void Success(View view) { //upon pressing the button
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent); //Open the new page
        finish(); //Close current page
    }

    @Override
    public void onBackPressed() {//To prevent users from pressing the back button
on their phone
    }
}
```

Home Page

```
package com.nadramon.test92;

//import stuff
```

```
import android.app.Notification;
import android.content.Intent;
import android.graphics.Color;
import android.os.Handler;
import android.support.v4.app.NotificationManagerCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.app.NotificationCompat;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

public class homepagev2 extends AppCompatActivity {

    DatabaseReference DReference =
FirebaseDatabase.getInstance().getReference();//Get database reference
    DatabaseReference DAccounts = DReference.child("Accounts"); //get reference
for the accounts field
    DatabaseReference DUserAcc; //define variable type

    //Defining data types for variables
    Handler timer2;
    TextView Starter;
    TextView Welcome;
    Button MakeOrder;
    Button EditAccount;
    Button ViewHistory;
    Button ExitApp;
    String Name;
    String Username;
    String Password;
    String Region;
    String Street;
    String House;
    int First;
    ArrayList<String> SelectedItem;
    ArrayList<String> SelectedPrice;
    ArrayList<Integer> Quantity;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_homepagev2); //Creates the layout

        Intent intent = getIntent();
        //Retrieves values from previous activity
        Name = intent.getExtras().getString("Name");
        Username = intent.getExtras().getString("Username");
```

```
        Password = intent.getExtras().getString("Password");
        Region = intent.getExtras().getString("Region");
        Street = intent.getExtras().getString("Street");
        House = intent.getExtras().getString("House");
        First = intent.getExtras().getInt("First");


        //new lists
        SelectedItem = new ArrayList<>();
        SelectedPrice = new ArrayList<>();
        Quantity = new ArrayList<>();


        //Gives the variables their respective values
        MakeOrder = (Button) findViewById(R.id.HPMakeOrder);
        EditAccount = (Button) findViewById(R.id.HPEditAccount);
        ViewHistory = (Button) findViewById(R.id.HPViewHistory);
        ExitApp = (Button) findViewById(R.id.HPExitApp);
        Starter = (TextView) findViewById(R.id.HPStarter);
        Welcome = (TextView) findViewById(R.id.HPWelcome);

        //If first time landing on the home page (coming from the log in page)
        if (First == 0) {
            timer2 = new Handler();//new handler
            timer2.postDelayed(new Runnable() { //wait for 3 seconds
                @Override
                public void run() {
                    //reveal all the buttons and message and hide the starter
message
                    Starter.setVisibility(View.INVISIBLE);
                    Welcome.setText("Welcome " + Name + "! What would you like to
order?");
                    MakeOrder.setVisibility(View.VISIBLE);
                    EditAccount.setVisibility(View.VISIBLE);
                    ViewHistory.setVisibility(View.VISIBLE);
                    ExitApp.setVisibility(View.VISIBLE);
                }
            }, 3000);
        }
        //otherwise...
        else {
            //do the same without waiting 3 seconds
            Starter.setVisibility(View.INVISIBLE);
            Welcome.setText("Welcome " + Name + "! What would you like to
order?");
            MakeOrder.setVisibility(View.VISIBLE);
            EditAccount.setVisibility(View.VISIBLE);
            ViewHistory.setVisibility(View.VISIBLE);
            ExitApp.setVisibility(View.VISIBLE);
        }

        //Change first to 1 since we have visited the home page
        First = 1;

        //get reference for the username
        DUserAcc = DAccounts.child(Username);

    }
```

```
    //If the "Exit App" button is pressed, completely shuts down the App
    public void ShutDown(View view) {
        finish(); //close this page
        System.exit(0); //shut down
    }
```

This achieves objective 1fi

```
    //Procedure to receive notifications
    private void GetNotification(String message) { //needs a string variable
        NotificationCompat.Builder builder = new
NotificationCompat.Builder(this); //notification builder
        builder.setSmallIcon(R.drawable.ic_notification_alert); //set the icon
        builder.setContentTitle("Order Status Update!"); //set the title of the
notification
        builder.setContentText(message); //set the message
        builder.setVibrate(new long[] {1000L, 1000L, 1000L,}); //set the length
of the vibration
        builder.setLights(Color.RED, 3000, 3000); //set the flashing lights on
phone
        Notification notification = builder.build(); //build the notification
        NotificationManagerCompat.from(this).notify(1, notification); //play the
notification
    }
```

This achieves objective 1gi, 1gii

```
    //procedure to change page
    private void ChangePage(Class page, boolean makeOrder) {//need class and
boolean variable
        Intent change = new Intent(this, page);
        //pass these values to the next page
        change.putExtra("Name", Name);
        change.putExtra("Username", Username);
        change.putExtra("Password", Password);
        change.putExtra("Region", Region);
        change.putExtra("Street", Street);
        change.putExtra("House", House);
        change.putExtra("First", First);
        if (makeOrder) { //if going to the make an order page
            //pass these lists there
            change.putStringArrayListExtra("SelectedItem", SelectedItem);
            change.putStringArrayListExtra("SelectedPrice", SelectedPrice);
            change.putIntegerArrayListExtra("Quantity",Quantity);
        }
        startActivity(change); //open the new page
        finish(); //close current page
    }

    public void EditAcc(View view) { //run change page procedure
        ChangePage(EditAccount.class, false);
    }

    public void MakeOrder(View view) { //run change page procedure
        Date now = new Date(); //get current date
        SimpleDateFormat currentTime = new SimpleDateFormat("HH"); //get just the
hour
        String currentTimeS = currentTime.format(now); // convert it to string
        int currentTimeI = Integer.parseInt(currentTimeS); //convert it to
integer
```

```
        if (currentTimeI >= 21 || currentTimeI < 9) {//If the number is smaller
than 9 or bigger than 21...
            //error popup
            Toast.makeText(getApplicationContext(), "You cannot order for home
delivery at this hour.", Toast.LENGTH_SHORT).show();
        }
        else {
            //otherwise run change page procedure
            ChangePage(MakeAnOrder.class, true);
        }
    }

    public void ViewHistory(View view) {//run change page procedure
        ChangePage(OrderHistory.class, false);
    }

    @Override
    protected void onStart() {
        super.onStart();

        DUserAcc.addValueEventListener(new ValueEventListener() {//Check the
database just this once
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                //Get a value of the order status from the username field
                String message = (String) dataSnapshot.child("Order
Status").getValue();
                //is the record wasn't empy
                if (message != null) {
                    GetNotification(message);//run get notification procedure
                }
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });
    }
    @Override
    public void onBackPressed() {//To prevent users from pressing the back button
on their phone
    }
}
```

This achieves objective 1avi

Make an Order Page

```
package com.nadramon.test92;

//import stuff
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.os.Handler;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.view.ContextThemeWrapper;
import android.text.InputType;
```

```
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

public class MakeAnOrder extends AppCompatActivity {

    //Getting Database references
    DatabaseReference DReference = FirebaseDatabase.getInstance().getReference();
    DatabaseReference DMenu = DReference.child("Menu");

    //Defining variable types globally
    Spinner Categories;
    LinearLayout linearLayout;
    ArrayList<String> SelectedItem;
    ArrayList<String> SelectedPrice;
    ArrayList<Float> SelectedPriceF;
    ArrayList<Integer> Quantity;
    String Name;
    String Username;
    String Password;
    String Region;
    String Street;
    String House;
    int First;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_make_an_order);
        linearLayout = (LinearLayout) findViewById(R.id.forItems); //creates the
initial layout


        //Creating the spinner list of item categories
        Categories = (Spinner) findViewById(R.id.MOspinner);
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.Categories,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        Categories.setAdapter(adapter);
```

```
        //Creating Lists for customers' orders
        SelectedPriceF = new ArrayList<>();

        Intent intent = getIntent();
        //Retrieves values from previous activity
        Name = intent.getExtras().getString("Name");
        Username = intent.getExtras().getString("Username");
        Password = intent.getExtras().getString("Password");
        Region = intent.getExtras().getString("Region");
        Street = intent.getExtras().getString("Street");
        House = intent.getExtras().getString("House");
        First = intent.getExtras().getInt("First");
        SelectedItem = intent.getStringArrayListExtra("SelectedItem");
        SelectedPrice = intent.getStringArrayListExtra("SelectedPrice");
        Quantity = intent.getIntegerArrayListExtra("Quantity");


    }

    @Override
    public void onBackPressed() {//To prevent users from pressing the back button
on their phone

    }

    //Takes users back to the previous page, the Home Page
    public void Back(View view) {
        Intent Back = new Intent(this, homepagev2.class);
        //pass these values to the next page
        Back.putExtra("Name", Name);
        Back.putExtra("Username", Username);
        Back.putExtra("Password", Password);
        Back.putExtra("Region", Region);
        Back.putExtra("Street", Street);
        Back.putExtra("House", House);
        Back.putExtra("First", First);
        startActivity(Back); //open the new page
        finish();//close current page
    }


    //Displays the items within the category
    public void GetCategory(View view) {

        int s = linearLayout.getChildCount(); //Gets number of Views in the
layout
        if (s > 0) { //Because initially there is nothing
        //this is more for when the button is being pressed after the first time
            for (int x = 0; x < s; x++) {
                //It will get rid of the views after the 0 views
                linearLayout.removeViewAt(0);
            }
        }
        final String ChosenCategory = Categories.getSelectedItem().toString();
//Gets the selected item in the spinner
        DMenu.addValueEventListener(new ValueEventListener() { //Connect to the
firebase database
            @Override
```

This achieves objective 1ai

```
            public void onDataChange(final DataSnapshot dataSnapshot) {
                Long count =
dataSnapshot.child("Item").child(ChosenCategory).getChildrenCount(); //Gets
number of items of that category
                for (int x = 0; x < count; x++) {

                    final String y = Integer.toString(x);
                    final String item = (String)
dataSnapshot.child("Item").child(ChosenCategory).child(y).getValue(); //Gets
value of Item
                    final String price = (String)
dataSnapshot.child("Price").child(ChosenCategory).child(y).getValue(); //Gets
value of the corresponding price
                    TextView Items = new TextView(MakeAnOrder.this); //Creating a
new view of TextView
                    Items.setText(item); //Gives the view the text of the Item
                    Items.setTextSize(23); //Sets the font size to 23
                    Items.setTextColor(Color.BLACK); //Set the text colour to
black
                    Items.setClickable(true); //Allows it to be clicked on by
users
                    linearLayout.addView(Items); //Adds the new view to the
layout

                    Items.setOnClickListener(new View.OnClickListener() { //Sets
something to happen when it is clicked
                        @Override
                        public void onClick(View v) {

                            String CompleteSentence = item + ": " + price + "
OMR" + '\n' + "How many portions would you like?";

                            //Creates a Dialog Box
                            final AlertDialog.Builder dialogBox = new
AlertDialog.Builder(new ContextThemeWrapper(MakeAnOrder.this, R.style.myDialog));
                            dialogBox.setMessage(CompleteSentence); //Giving the
Dialog box a message
                            final EditText quantity = new
EditText(MakeAnOrder.this); //Creating a new Entry box in the layout
                            quantity.setInputType(InputType.TYPE_CLASS_NUMBER);
//This only lets user put integers, no decimals or other characters
                            dialogBox.setView(quantity); //Puts the new entry box
inside the dialog box
                            dialogBox.setPositiveButton("OK", new
DialogInterface.OnClickListener() { //Sets the positive button for the dialog box
                                @Override
                                public void onClick(DialogInterface dialog, int
which) {
                                    String number =
quantity.getText().toString(); //Gets the input from the user
                                    if (number.equals("")) { //Error trap, in the
event they just press "OK" and not insert any number
                                        Toast.makeText(getApplicationContext(),
"You did not type any number.", Toast.LENGTH_SHORT).show(); //Popup
                                        dialog.dismiss(); //closes dialog box
                                    }
                                    else {
                                        int number2 = Integer.parseInt(number);
//Converts input to an integer
```

This achieves objective 1aii

```
                                        if (number2 > 0) { //Error trap, in case
they type 0
                                            Float PricePer =
Float.parseFloat(price); //Converts price value to a float
                                            SelectedItem.add(item); //Adds the
item to the Item List
                                            SelectedPrice.add(price); //Adds the
price (String) to the Price List
                                            SelectedPriceF.add(PricePer); //Adds
the price (float) to the Price f list
                                            Quantity.add(number2); //Adds the
quantity to the Quantity List
                                            CharSequence added = number + "
portions of " + item + " has been added to your order.";

Toast.makeText(getApplicationContext(), added, Toast.LENGTH_SHORT).show();
//Popup
                                            dialog.dismiss(); //closes dialog box
                                        }
                                        else {

Toast.makeText(getApplicationContext(), "You can't order zero amount of
something.", Toast.LENGTH_SHORT).show(); //Popup
                                            dialog.dismiss(); //closes dialog box
                                        }
                                    }
                                }
                            });
                            dialogBox.setNegativeButton("CANCEL", new
DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int
which) {
                                    dialog.dismiss(); //closes dialog box

                                }
                            });
                            AlertDialog Dialog = dialogBox.create(); //Creates
the dialog box
                            Dialog.show(); // Making the dialog box appear

                        }
                    });

                    //Creates an image view
                    ImageView divider = new ImageView(MakeAnOrder.this);
                    LinearLayout.LayoutParams div = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, 5);
                    div.setMargins(15, 15, 15, 15); //sets the margins
                    divider.setLayoutParams(div);
                    linearLayout.addView(divider); //add it to the layout
                    //It's an empty image, just to create spacing between the
items
                }
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });
```

```
    }

    public void CheckOrder(View view) {
        if (SelectedItem.size() == 0) { //error trap so they do not move forward
without ordering anything
            Toast.makeText(getApplicationContext(), "You have not selected any
items!", Toast.LENGTH_SHORT).show();
        }
        else {
            Intent intent = new Intent(this, CheckOrder.class);
            //pass these values and lists to the next page
            intent.putExtra("Name", Name);
            intent.putExtra("Username", Username);
            intent.putExtra("Password", Password);
            intent.putExtra("Region", Region);
            intent.putExtra("Street", Street);
            intent.putExtra("House", House);
            intent.putExtra("First", First);
            intent.putStringArrayListExtra("SelectedItem", SelectedItem);
            intent.putStringArrayListExtra("SelectedPrice", SelectedPrice);
            intent.putIntegerArrayListExtra("Quantity",Quantity);
            startActivity(intent); //open the next page
            finish();//close current page
        }
    }
}
```

Check Order Page

```
package com.nadramon.test92;

import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.view.ContextThemeWrapper;
import android.text.InputType;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
import android.widget.TableRow.LayoutParams;
import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.math.RoundingMode;
import java.security.MessageDigest;
```

```
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

public class CheckOrder extends AppCompatActivity {

    DatabaseReference DReference = FirebaseDatabase.getInstance().getReference();
//get the database reference
    DatabaseReference DOrderH = DReference.child("Order History"); // get
reference for the order history field
    DatabaseReference DPendingO = DReference.child("Pending Orders"); //get
reference for the pending orders field

    //Define the variable types
    ArrayList<String> SelectedItem;
    ArrayList<String> SelectedPrice;
    ArrayList<Integer> Quantity;
    String Name;
    String Username;
    String Password;
    String Region;
    String Street;
    String House;
    String TCostS;
    int First;
    Float TCost;
    Long childcount;
    Long childcount2;
    Boolean Discount;
    ArrayList<String> QuantityS;
    ArrayList<String> TotalPrices;
    DatabaseReference DAcc;


//Method for hashing password

    public static String sha256(String base) {
        try{
            MessageDigest digest = MessageDigest.getInstance("SHA-256"); //Get
the SHA-256 hash
            byte[] hash = digest.digest(base.getBytes("UTF-8")); //convert input
string to series of bytes
            StringBuffer hexString = new StringBuffer(); //make the string
modifiable

            for (int i = 0; i < hash.length; i++) { //for each set of bytes
                String hex = Integer.toHexString(0xff & hash[i]); //Convert the
bytes sequence to a hex string
                if (hex.length() == 1) hexString.append('0');
                hexString.append(hex);//add each one to one string

            }
            return hexString.toString(); //convert it back to a regular string
and return the value
        }
        catch (Exception e){
            throw new RuntimeException(e);
        }
```

```
    }

    //function to calculate the new total cost with discount
    private static String discountCalculate(Float OriginalCost) {
        Float a = 100F;
        Float b = 15F;
        Float c = OriginalCost / a;
        Float d = c * b;
        //Calculation to get 15% off
        //Makes sure it only goes to one decimal place
        DecimalFormat decimalFormat = new DecimalFormat("#.#");
        //round the value to one dp
        decimalFormat.setRoundingMode(RoundingMode.CEILING);
        //return the value
        return decimalFormat.format(d);

    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_check_order); //creates the page with
the specified layout

        Intent intent = getIntent();
        //Retrieves values from previous activity
        Name = intent.getExtras().getString("Name");
        Username = intent.getExtras().getString("Username");
        Password = intent.getExtras().getString("Password");
        Region = intent.getExtras().getString("Region");
        Street = intent.getExtras().getString("Street");
        House = intent.getExtras().getString("House");
        First = intent.getExtras().getInt("First");
        SelectedItem = intent.getStringArrayListExtra("SelectedItem");
        SelectedPrice = intent.getStringArrayListExtra("SelectedPrice");
        Quantity = intent.getIntegerArrayListExtra("Quantity");

        final int length = SelectedItem.size(); //get size of the item list
        Discount = false; //set by default the discount to false


        DAcc = DOrderH.child(Username); //get reference for the username's order
history

        DAcc.addListenerForSingleValueEvent(new ValueEventListener() { //Check
database just this once
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                Long numberOfOrders = dataSnapshot.getChildrenCount(); //Get
number of things under the field
                if (numberOfOrders >= 15) { //If it is greater than or equal to
15...
                    Discount = true; //Change discount to true
                }
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });
```

This achieves objective 1bi

```
        //New list for the price in FLOAT
        ArrayList<Float> SelectedPriceF = new ArrayList<>();

        //for each price
        for (int x = 0; x < length; x++) {
            String number = SelectedPrice.get(x); //geth the price
            Float real = Float.parseFloat(number); //Convert string to float
            SelectedPriceF.add(real); //add the float value to the new list
        }

        //new list for total price
        TotalPrices = new ArrayList<>();
        TCost = 0.0F; //initial total cost is 0

        //for each price
        for (int x = 0; x < length; x++) {
            //multiply the price with the quantity
            Float total = SelectedPriceF.get(x) * Quantity.get(x);
            String Total = Float.toString(total); //convert from float to a
string
            TotalPrices.add(Total);//Add the total price to the new list
            TCost += total; //Add the total price to the total cost
        }
        if (Discount) { //If the user gets a discount
            TCostS = discountCalculate(TCost);//run the discount function
        }
        else { //otherwise
            TCostS = Float.toString(TCost); //simply convert the total cost from
float to a string
        }

        //new list for quantity string version
        QuantityS = new ArrayList<>();

        //for each quantity
        for (int x = 0; x < length; x++) {
            int giveme = Quantity.get(x); //get the value
            String stringed = Integer.toString(giveme); //convert from integer to
string
            QuantityS.add(stringed); //add to the new list
        }


        TableLayout tableLayout = (TableLayout)
findViewById(R.id.COTableLayoutID); //get table layout reference
        LayoutParams textLayout = new LayoutParams(LayoutParams.FILL_PARENT,
LayoutParams.WRAP_CONTENT, 1);// set the layout parameters

        //make a new table row (first row)
        TableRow Frow = new TableRow(this);
        //set layout parameter
        Frow.setLayoutParams(new LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.WRAP_CONTENT));
        Frow.setWeightSum(4); //have 4 columns
        Frow.setOrientation(TableRow.VERTICAL); //make it go across

        //make new textviews
        TextView ItemT = new TextView(this);
```

This achieves objective 1aiii

This achieves objective 1aiv

This achieves objective 1bii

```
        TextView QtyT = new TextView(this);
        TextView PriceT = new TextView(this);
        TextView TotalT = new TextView(this);

        ItemT.setText(R.string.item);//set the text
        ItemT.setTextSize(18); //set the text size
        ItemT.setTextColor(Color.BLACK); //set the font colour
        ItemT.setLayoutParams(textLayout); //set the layout parameters
        ItemT.setTextAlignment(View.TEXT_ALIGNMENT_CENTER); //set the alignment
        ItemT.setRight(1);

        QtyT.setText(R.string.qty);//set the text
        QtyT.setTextSize(18);//set the text size
        QtyT.setTextColor(Color.BLACK);//set the font colour
        QtyT.setLayoutParams(textLayout);//set the layout parameters
        QtyT.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);//set the alignment
        QtyT.setRight(1);

        PriceT.setText(R.string.price);//set the text
        PriceT.setTextSize(18);//set the text size
        PriceT.setTextColor(Color.BLACK);//set the font colour
        PriceT.setLayoutParams(textLayout);//set the layout parameters
        PriceT.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);//set the alignment
        PriceT.setRight(1);

        TotalT.setText(R.string.total);//set the text
        TotalT.setTextSize(18);//set the text size
        TotalT.setTextColor(Color.BLACK);//set the font colour
        TotalT.setLayoutParams(textLayout);//set the layout parameters
        TotalT.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);//set the alignment
        TotalT.setRight(1);

        //add the textviews into each column
        Frow.addView(ItemT);
        Frow.addView(QtyT);
        Frow.addView(PriceT);
        Frow.addView(TotalT);
        tableLayout.addView(Frow); //add this row to the table

        //for each item in the list
        for (int x = 0; x < length; x++) {

            //same like above, just for the middle rows
            TableRow row = new TableRow(this);
            row.setLayoutParams(new LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.WRAP_CONTENT));
            row.setWeightSum(4);
            row.setOrientation(TableRow.VERTICAL);

            TextView item = new TextView(this);
            TextView qty = new TextView(this);
            TextView price = new TextView(this);
            TextView totalprice = new TextView(this);


            item.setText(SelectedItem.get(x));
            item.setTextSize(14);
            item.setTextColor(Color.BLACK);
            item.setTextAlignment(View.TEXT_ALIGNMENT_TEXT_START);
```

```
            item.setLayoutParams(textLayout);
            item.setRight(1);

            qty.setText(QuantityS.get(x));
            qty.setTextSize(14);
            qty.setTextColor(Color.BLACK);
            qty.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
            qty.setLayoutParams(textLayout);
            qty.setRight(1);

            price.setText(SelectedPrice.get(x));
            price.setTextSize(14);
            price.setTextColor(Color.BLACK);
            price.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
            price.setLayoutParams(textLayout);
            price.setRight(1);

            totalprice.setText(TotalPrices.get(x));
            totalprice.setTextSize(14);
            totalprice.setTextColor(Color.BLACK);
            totalprice.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
            totalprice.setLayoutParams(textLayout);
            totalprice.setRight(1);


            row.addView(item);
            row.addView(qty);
            row.addView(price);
            row.addView(totalprice);

            tableLayout.addView(row);


        }

        //same as above, just for last row
        TableRow Lrow = new TableRow(this);
        Lrow.setLayoutParams(new LayoutParams(LayoutParams.MATCH_PARENT,
LayoutParams.WRAP_CONTENT));
        Lrow.setWeightSum(1); //only 1 column
        Lrow.setOrientation(TableRow.VERTICAL);

        TextView costT = new TextView(this);

        final String TotalCost = "Total Cost: " + TCostS + " Rials";

        costT.setText(TotalCost);
        costT.setTextSize(14);
        costT.setTextColor(Color.BLACK);
        costT.setTextAlignment(View.TEXT_ALIGNMENT_TEXT_END);
        costT.setLayoutParams(textLayout);
        costT.setRight(1);

        Lrow.addView(costT);

        tableLayout.addView(Lrow);


        LinearLayout linearLayout = (LinearLayout)
```

```
findViewById(R.id.activity_check_order);


        //make a new button
        Button Restart = new Button(CheckOrder.this);
        Restart.setText(R.string.restart_order);//set text
        Restart.setClickable(true);//make it clickable
        linearLayout.addView(Restart);//add it to the layout
        Restart.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {//upon clicking
                //clear the lists
                SelectedPrice.clear();
                SelectedItem.clear();
                Quantity.clear();
                Intent intent = new Intent(v.getContext(), MakeAnOrder.class);
                //pass these values and lists to the next page
                intent.putExtra("Name", Name);
                intent.putExtra("Username", Username);
                intent.putExtra("Password", Password);
                intent.putExtra("Region", Region);
                intent.putExtra("Street", Street);
                intent.putExtra("House", House);
                intent.putExtra("First", First);
                intent.putStringArrayListExtra("SelectedItem", SelectedItem);
                intent.putStringArrayListExtra("SelectedPrice", SelectedPrice);
                intent.putIntegerArrayListExtra("Quantity",Quantity);
                startActivity(intent);//open next page
                finish();//close current page
            }
        });


        //make a new button
        Button Confirm = new Button(CheckOrder.this);
        Confirm.setText(R.string.confirm_order);
        Confirm.setClickable(true); //same as above
        linearLayout.addView(Confirm);
        Confirm.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { //upon clicking...

                //make a dialog box
                AlertDialog.Builder dialogBox = new AlertDialog.Builder(new
ContextThemeWrapper(CheckOrder.this, R.style.myDialog));
                dialogBox.setMessage("Please enter your account password to
confirm your order:"); //set the message
                final EditText Confirmation = new EditText(CheckOrder.this);
//make a new edit text

Confirmation.setInputType(InputType.TYPE_TEXT_VARIATION_PASSWORD); //set the
input type so that it is hidden
                dialogBox.setView(Confirmation); // add the edit text to the
dialog box
                dialogBox.setPositiveButton("SEND ORDER", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
//Set the SEND ORDER button
```

This achieves objective 1av

```
                        String ConfirmPassword =
Confirmation.getText().toString();//Get the value from the edit text
                        if (sha256(ConfirmPassword).equals(Password)) { //Compare
password, if success

                            DAcc.addListenerForSingleValueEvent(new
ValueEventListener() { //read database just this once
                                @Override
                                public void onDataChange(DataSnapshot
dataSnapshot) {
                                    childcount = dataSnapshot.getChildrenCount();
//Get how many records there are under the account
                                    Long newVal = childcount + 1; //Get a the new
ID
                                    String NewVal = Long.toString(newVal); //
convert from long to string
                                    String newOrder = "Order " + NewVal; //set
text for new ID

                                    //get reference of database for the new
records
                                    DatabaseReference NewOrder =
DAcc.child(newOrder);
                                    DatabaseReference DItem =
NewOrder.child("Item");
                                    DatabaseReference DPrice =
NewOrder.child("Price");
                                    DatabaseReference DQuantity =
NewOrder.child("Quantity");
                                    DatabaseReference DTotal =
NewOrder.child("Total Price");
                                    DatabaseReference DCost =
NewOrder.child("Total Cost");

                                    //for each item
                                    for (int x = 0; x < length; x++) {
                                        String y = Integer.toString(x); //convert
counter to string
                                        //add the order details to the database

DItem.child(y).setValue(SelectedItem.get(x));

DPrice.child(y).setValue(SelectedPrice.get(x));

DQuantity.child(y).setValue(QuantityS.get(x));

DTotal.child(y).setValue(TotalPrices.get(x));
                                    }

                                    //get current date
                                    Date now = new Date();
                                    SimpleDateFormat CurrentDate = new
SimpleDateFormat("dd/MM/yyyy"); //get date
                                    String CurrentDateS =
CurrentDate.format(now); //convert it to string
                                    //add details to database

NewOrder.child("Date").setValue(CurrentDateS);
                                    NewOrder.child("OrderNo").setValue(newOrder);
```

This achieves objective 1avii

```
                                    DCost.setValue(TCostS);
                                }

                                @Override
                                public void onCancelled(DatabaseError
databaseError) {
                                }
                            });

                            DPendingO.addListenerForSingleValueEvent(new
ValueEventListener() {//Read database just this once
                                @Override
                                public void onDataChange(DataSnapshot
dataSnapshot) {

                                    childcount2 = 5L; //set value of the count
initially as 5

                                    Date now = new Date(); //get current date
                                    SimpleDateFormat CurrentTime = new
SimpleDateFormat("HHmm"); //get hour and minutes together
                                    String CurrentTimeS =
CurrentTime.format(now); //change it to string

                                    //Error trap, in the event two or more people
make the order at the same time
                                    while (childcount2 == 5) {//while the count
is still 5
                                        childcount2 =
dataSnapshot.child(CurrentTimeS).getChildrenCount(); //Get count of the pending
order ID
                                        if (childcount2 == 5) {//if that time id
has 5 things under it
                                            int CurrentTimeI =
Integer.parseInt(CurrentTimeS); //Change the count to integer
                                            CurrentTimeI = CurrentTimeI + 1;
//add one
                                            CurrentTimeS =
Integer.toString(CurrentTimeI); // change it back to string and try again
                                        }

                                    }


                                    //Get reference of database for the new
pending orders
                                    DatabaseReference DiD =
DPendingO.child(CurrentTimeS);
                                    DatabaseReference DTime = DiD.child("Time");
                                    DatabaseReference DItem2 = DiD.child("Item");
                                    DatabaseReference DQuantity2 =
DiD.child("Quantity");
                                    DatabaseReference DUser =
DiD.child("Username");
                                    DatabaseReference DCost = DiD.child("Total
Cost");

                                    //for each item and quantity add to the
database
```

This achieves objective 1aviii

```
                                    for (int x = 0; x < length; x++) {
                                        String y = Integer.toString(x);

DItem2.child(y).setValue(SelectedItem.get(x));

DQuantity2.child(y).setValue(QuantityS.get(x));
                                    }
                                    //Add the other values to the database
                                    DTime.setValue(CurrentTimeS);
                                    DUser.setValue(Username);
                                    DCost.setValue(TCostS);
                                }
                                @Override
                                public void onCancelled(DatabaseError
databaseError) {
                                }
                            });
                            //new activity to homepage (like signupcomplete
thing)
                            Intent intent = new Intent(CheckOrder.this,
AfterOrderTransition.class);
                            //pass these values to the next page
                            intent.putExtra("Name", Name);
                            intent.putExtra("Username", Username);
                            intent.putExtra("Password", Password);
                            intent.putExtra("Region", Region);
                            intent.putExtra("Street", Street);
                            intent.putExtra("House", House);
                            intent.putExtra("First", First);
                            dialog.dismiss(); //close dialog box
                            startActivity(intent); //open the new page
                            finish(); //close the current page
                        }
                        else {
                            Toast.makeText(getApplicationContext(),"Password is
incorrect", Toast.LENGTH_SHORT).show(); //error pop up
                            dialog.dismiss(); //close the dialog
                        }
                    }
                });
                dialogBox.setNegativeButton("CANCEL", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.dismiss(); //close the dialog box
                    }
                });
                dialogBox.create().show(); //make the dialog box and make it
appear on screen

            }
        });

    }

    public void Back(View view) { //upon pressing the arrow button
        Intent intent = new Intent(this, MakeAnOrder.class);
        //pass these values and lists to the next page
        intent.putExtra("Name", Name);
```

```
        intent.putExtra("Username", Username);
        intent.putExtra("Password", Password);
        intent.putExtra("Region", Region);
        intent.putExtra("Street", Street);
        intent.putExtra("House", House);
        intent.putExtra("First", First);
        intent.putStringArrayListExtra("SelectedItem", SelectedItem);
        intent.putStringArrayListExtra("SelectedPrice", SelectedPrice);
        intent.putIntegerArrayListExtra("Quantity",Quantity);
        startActivity(intent); //open new page
        finish();//close current page
    }

    @Override
    public void onBackPressed() {//To prevent users from pressing the back button
on their phone
    }
}
```

After Order Complete Page

```
package com.nadramon.test92;

import android.content.Intent;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class AfterOrderTransition extends AppCompatActivity {

    //Define variable types
    Handler timer;
    Button button;
    String Name;
    String Username;
    String Password;
    String Region;
    String Street;
    String House;
    int First;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_after_order_transition); //create page
with specified layout
        timer = new Handler(); //new handler
        timer.postDelayed(new Runnable() { //wait for 1 second
            @Override
            public void run() {
                button = (Button) findViewById(R.id.AOTenter); //get the XML
reference for the button
                button.setText(R.string.continue_home); //Set the text
                button.setVisibility(View.VISIBLE); //Make it visible
            }
```

```
        }, 1000);


        Intent intent = getIntent();
        //Retrieves values from previous activity
        Name = intent.getExtras().getString("Name");
        Username = intent.getExtras().getString("Username");
        Password = intent.getExtras().getString("Password");
        Region = intent.getExtras().getString("Region");
        Street = intent.getExtras().getString("Street");
        House = intent.getExtras().getString("House");
        First = intent.getExtras().getInt("First");

    }

    public void Success(View view) {//Upon clicking the button
        Intent intent = new Intent(this, homepagev2.class);
        //pass these values to the next page
        intent.putExtra("Name", Name);
        intent.putExtra("Username", Username);
        intent.putExtra("Password", Password);
        intent.putExtra("Region", Region);
        intent.putExtra("Street", Street);
        intent.putExtra("House", House);
        intent.putExtra("First", First);
        startActivity(intent); //open new page
        finish(); //close current page
    }
    @Override
    public void onBackPressed() {//To prevent users from pressing the back button
on their phone
    }
}
```

Edit Account Page

```
package com.nadramon.test92;

import android.content.DialogInterface;
import android.content.Intent;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.view.ContextThemeWrapper;
import android.text.InputType;
import android.view.View;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.Toast;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.security.MessageDigest;

public class EditAccount extends AppCompatActivity {
```

```
    //Define variable types
    String Name;
    String Username;
    String Password;
    String Region;
    String Street;
    String House;
    int First;
    EditText ETname;
    EditText ETpassword;
    EditText ETstreet;
    EditText EThouse;
    RadioButton Azaiba;
    RadioButton Ghubrah;
    String NName;
    String NPassword;
    String NRegion;
    String NStreet;
    String NHouse;

    DatabaseReference DReference = FirebaseDatabase.getInstance().getReference();
//get database reference
    DatabaseReference DAccounts = DReference.child("Accounts"); //get reference
for the accounts field
    DatabaseReference DUserAcc; //define variable type

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_account); //create page with the
specified layout

        Intent intent = getIntent();
        //Retrieves values from previous activity
        Name = intent.getExtras().getString("Name");
        Username = intent.getExtras().getString("Username");
        Password = intent.getExtras().getString("Password");
        Region = intent.getExtras().getString("Region");
        Street = intent.getExtras().getString("Street");
        House = intent.getExtras().getString("House");
        First = intent.getExtras().getInt("First");

        DUserAcc = DAccounts.child(Username); //get reference for the username
field

        //Gives the variables its XML reference
        ETname = (EditText) findViewById(R.id.EAEName);
        ETpassword = (EditText) findViewById(R.id.EAEPassword);
        ETstreet = (EditText) findViewById(R.id.EAEStreet);
        EThouse = (EditText) findViewById(R.id.EAEHouse);

        //Set the text of the EditTexts
        ETname.setText(Name);
        ETstreet.setText(Street);
        EThouse.setText(House);
        //Gives the variables its XML reference
        Azaiba = (RadioButton) findViewById(R.id.EARAzaiba);
        Ghubrah = (RadioButton) findViewById(R.id.EARGhubrah);
```

```
        //Set the radio option in accordance to the user's detail
        if (Region.equals("Azaiba")) {
            Azaiba.setChecked(true);
        }
        else {
            Ghubrah.setChecked(true);
        }

    }

    //upon clicking the azaiba button, azaiba becomes true and ghubrah becomes
false
    public void AzaibaB(View view) {
        Azaiba.setChecked(true);
        Ghubrah.setChecked(false);
    }

    //vice versa
    public void GhubrahB(View view) {
        Azaiba.setChecked(false);
        Ghubrah.setChecked(true);
    }

//Method for hashing password

    public static String sha256(String base) {
        try{
            MessageDigest digest = MessageDigest.getInstance("SHA-256"); //Get
the SHA-256 hash
            byte[] hash = digest.digest(base.getBytes("UTF-8")); //convert input
string to series of bytes
            StringBuffer hexString = new StringBuffer(); //make the string
modifiable

            for (int i = 0; i < hash.length; i++) { //for each set of bytes
                String hex = Integer.toHexString(0xff & hash[i]); //Convert the
bytes sequence to a hex string
                if (hex.length() == 1) hexString.append('0');
                hexString.append(hex);//add each one to one string

            }
            return hexString.toString(); //convert it back to a regular string
and return the value
        }
        catch (Exception e){
            throw new RuntimeException(e);
        }
    }

    public void Change(View view) { //upon clicking the change button

        //make a new dialog box
        AlertDialog.Builder dialogBox = new AlertDialog.Builder(new
ContextThemeWrapper(this, R.style.myDialog));
        dialogBox.setMessage("Please enter your account password (old password if
you just changed it) to confirm your change:"); //set the message
        final EditText Confirmation = new EditText(this); //make a new edit text
        Confirmation.setInputType(InputType.TYPE_TEXT_VARIATION_PASSWORD); //set
input type so it is hidden
```

This achieves objective 1dii

```
        dialogBox.setView(Confirmation); //add the edit text in thedialog box
        dialogBox.setPositiveButton("UPDATE", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String ConfirmPassword = Confirmation.getText().toString(); //Get
the value from the edit text
                NPassword = ETpassword.getText().toString(); //Get the value from
the password edit text on the page
                int NPasswordLength = NPassword.length(); //Get the length of the
new password
                if (!sha256(ConfirmPassword).equals(Password)) {//Compare the
confirmation password with actual password, if not match

                    Toast.makeText(getApplicationContext(),"Password is
incorrect", Toast.LENGTH_SHORT).show(); //error popup
                    dialog.dismiss(); //close dialog box

                }

                else if (!NPassword.equals("") && NPasswordLength < 6) { //If the
new password is not blank or less than 6 characters
                    Toast.makeText(getApplicationContext(),"Password has to be at
least 6 characters.", Toast.LENGTH_SHORT).show(); //error popup
                    dialog.dismiss();//close dialog box
                }

                else { //otherwise

                    //get the new values (or the same ones if unchanged)
                    NName = ETname.getText().toString();
                    NStreet = ETstreet.getText().toString();
                    NHouse = EThouse.getText().toString();

                    if (Azaiba.isChecked()) {
                        NRegion = "Azaiba";
                    }
                    else {
                        NRegion = "Ghubrah";
                    }

                    //If password is not blank, then change of password
                    if (!NPassword.equals("")) {
                        Password = sha256(NPassword); //hash the new password
                        DUserAcc.child("Password").setValue(Password); //set the
new password in database
                    }//otherwise if password is blank, there is no change

                    //update new values to the database
                    DUserAcc.child("Name").setValue(NName);
                    DUserAcc.child("Region").setValue(NRegion);
                    DUserAcc.child("Street Address").setValue(NStreet);
                    DUserAcc.child("House Address").setValue(NHouse);

                    //success pop up
                    Toast.makeText(getApplicationContext(), "Updating Complete!",
Toast.LENGTH_SHORT).show();

                    dialog.dismiss(); //close dialog box
```

This achieves objective 1di

This achieves objective 1diii

```
                    Intent home = new Intent(EditAccount.this, homepagev2.class);
                    //pass these values to the next page
                    home.putExtra("Name", NName);
                    home.putExtra("Username", Username);
                    home.putExtra("Password", Password);
                    home.putExtra("Region", NRegion);
                    home.putExtra("Street", NStreet);
                    home.putExtra("House", NHouse);
                    home.putExtra("First", First);
                    startActivity(home); //open the new page
                    finish(); //close current page

                }
            }
        });
        dialogBox.setNegativeButton("CANCEL", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss(); //close dialog box
            }
        });
        dialogBox.create().show(); //make the dialog box and make it appear on
screen
    }

    public void EABack(View view) {
        Intent home = new Intent(this, homepagev2.class);
        //pass these values to the next page
        home.putExtra("Name", Name);
        home.putExtra("Username", Username);
        home.putExtra("Password", Password);
        home.putExtra("Region", Region);
        home.putExtra("Street", Street);
        home.putExtra("House", House);
        home.putExtra("First", First);
        startActivity(home); //open the new page
        finish(); //close current page
    }

    @Override
    public void onBackPressed() {//To prevent users from pressing the back button
on their phone
    }
}
```

View History Page

```
package com.nadramon.test92;

import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
```

```
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TextView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

public class OrderHistory extends AppCompatActivity {

    DatabaseReference DReference = FirebaseDatabase.getInstance().getReference();
//Get the database reference
    DatabaseReference DOrderHistory = DReference.child("Order History"); //Get
the reference for the order history field
    DatabaseReference DAcc; //define variable type

    //Define variable types
    Spinner Month;
    Spinner Year;
    String Name;
    String Username;
    String Password;
    String Region;
    String Street;
    String House;
    String MonthN;
    int First;
    LinearLayout linearLayout;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_order_history); //create page with
specified layout

        //get its XML reference
        linearLayout = (LinearLayout) findViewById(R.id.linearLayout);

        Intent intent = getIntent();
        //Retrieves values from previous activity
        Name = intent.getExtras().getString("Name");
        Username = intent.getExtras().getString("Username");
        Password = intent.getExtras().getString("Password");
        Region = intent.getExtras().getString("Region");
        Street = intent.getExtras().getString("Street");
        House = intent.getExtras().getString("House");
        First = intent.getExtras().getInt("First");

        DAcc = DOrderHistory.child(Username); //get reference for the username in
```

```
order history

        Date now = new Date(); //get current date

        SimpleDateFormat JustYear = new SimpleDateFormat("yyyy"); //get just the
year
        String CurrentYear = JustYear.format(now); //change to string
        int CurrentYearINT = Integer.parseInt(CurrentYear); //change to integer

        int StartingYear = 2013; //set a starting year (Changeable)

        //new list for years
        ArrayList<String> Years = new ArrayList<>();
        Years.add("2013"); //add the starting year

        //for each year between the current year and the starting year
        for (int a = 1; a < ((CurrentYearINT-StartingYear) + 1); a++ ) {
            int CircleYear = StartingYear + a;
            String CircularYear = Integer.toString(CircleYear);
            Years.add(CircularYear); //add it to the list
        }

        //Create spinner for the months
        Month = (Spinner) findViewById(R.id.OHspinner);
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.months,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        Month.setAdapter(adapter);

        //create spinner for the years
        Year = (Spinner) findViewById(R.id.OHspinner2);
        ArrayAdapter<String> adapter2 = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, Years);

adapter2.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        Year.setAdapter(adapter2);
    }

    public void Back(View view) {
        Intent Back = new Intent(this, homepagev2.class);
        //pass these values to the next page
        Back.putExtra("Name", Name);
        Back.putExtra("Username", Username);
        Back.putExtra("Password", Password);
        Back.putExtra("Region", Region);
        Back.putExtra("Street", Street);
        Back.putExtra("House", House);
        Back.putExtra("First", First);
        startActivity(Back); //open the new page
        finish(); //close current page
    }

    public void GetHistory(View view) {

        if (linearLayout != null) {
            //this is more for when the button is being pressed after the first
time
```

```
            int s = linearLayout.getChildCount(); //Gets number of Views in the
layout
            for (int x = 0; x < s; x++) {
                //It will get rid of all the views
                linearLayout.removeViewAt(0);
            }
        }

        //Get the selected month and year
        String ChosenMonth = Month.getSelectedItem().toString();
        String ChosenYear = Year.getSelectedItem().toString();

        //Get the numerical month from the textual month
        if (ChosenMonth.equals("January")) {
            MonthN = "01";
        }
        else if (ChosenMonth.equals("February")) {
            MonthN = "02";
        }
        else if (ChosenMonth.equals("March")) {
            MonthN = "03";
        }
        else if (ChosenMonth.equals("April")) {
            MonthN = "04";
        }
        else if (ChosenMonth.equals("May")) {
            MonthN = "05";
        }
        else if (ChosenMonth.equals("June")) {
            MonthN = "06";
        }
        else if (ChosenMonth.equals("July")) {
            MonthN = "07";
        }
        else if (ChosenMonth.equals("August")) {
            MonthN = "08";
        }
        else if (ChosenMonth.equals("September")) {
            MonthN = "09";
        }
        else if (ChosenMonth.equals("October")) {
            MonthN = "10";
        }
        else if (ChosenMonth.equals("November")) {
            MonthN = "11";
        }
        else {
            MonthN = "12";

        }

        //join them together
        final String SeeDate = MonthN + "/" + ChosenYear;

        DAcc.addListenerForSingleValueEvent(new ValueEventListener() { //read
database just this once
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
```

This achieves objective 1eii

```
                //for each order history
                for (DataSnapshot snapshot: dataSnapshot.getChildren()) {
                    final String ExistingDate = (String)
snapshot.child("Date").getValue(); //get the date
                    String ExistingDateWD = ExistingDate.substring(3,10); //cut
it so it is just the month and year

                    //Make a new list
                    final ArrayList<String> DItems = new ArrayList<>();
                    final ArrayList<String> DPrices = new ArrayList<>();
                    final ArrayList<String> DTPrices = new ArrayList<>();
                    final ArrayList<String> DQuantities = new ArrayList<>();

                    //if the month and year matches with the selected ones
                    if (SeeDate.equals(ExistingDateWD)) {
                        final String OrderNumber = (String)
snapshot.child("OrderNo").getValue();//Get the order number
                        TextView Items = new TextView(OrderHistory.this);
//Creating a new view of TextView
                        Items.setText(OrderNumber); //Gives the view the text of
the Item
                        Items.setTextSize(23); //Sets the font size to 23
                        Items.setTextColor(Color.BLACK); //Set the text colour to
black
                        Items.setClickable(true); //Allows it to be clicked on by
users
                        linearLayout.addView(Items); //Adds the new view to the
layout

                        final String Title = "This is: " + OrderNumber + ", the
order was made at " + ExistingDate + ".";

                        final String DTCost = (String) snapshot.child("Total
Cost").getValue();
                        Long counter = snapshot.child("Item").getChildrenCount();
                        //get number of records of items

                        //for each item
                        for (int x = 0; x < counter; x++) {

                            //get the values
                            String y = Integer.toString(x); //convert counter to
string
                            String DItem = (String)
snapshot.child("Item").child(y).getValue();
                            String DPrice = (String)
snapshot.child("Price").child(y).getValue();
                            String DTPrice = (String) snapshot.child("Total
Price").child(y).getValue();
                            String DQuantity = (String)
snapshot.child("Quantity").child(y).getValue();

                            //add it to the lists
                            DItems.add(DItem);
                            DPrices.add(DPrice);
                            DTPrices.add(DTPrice);
                            DQuantities.add(DQuantity);
                        }
```

This achieves objective 1eiii

```
                        //upon clicking the text
                        Items.setOnClickListener(new View.OnClickListener() {
                            @Override
                            public void onClick(View v) {

                                Intent History = new Intent(OrderHistory.this,
ViewHistory.class);
                                //Pass these values and lists to the next page
                                History.putExtra("Name", Name);
                                History.putExtra("Username", Username);
                                History.putExtra("Password", Password);
                                History.putExtra("Region", Region);
                                History.putExtra("Street", Street);
                                History.putExtra("House", House);
                                History.putExtra("First", First);
                                History.putStringArrayListExtra("DItems",DItems);

History.putStringArrayListExtra("DPrices",DPrices);

History.putStringArrayListExtra("DTPrices",DTPrices);

History.putStringArrayListExtra("DQuantities",DQuantities);
                                History.putExtra("DTCost",DTCost);
                                History.putExtra("Title",Title);
                                startActivity(History);//open new page
                                finish();//close current page

                            }
                        });

                        //Creates an image view
                        ImageView divider = new ImageView(OrderHistory.this);
                        LinearLayout.LayoutParams div = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, 5);
                        div.setMargins(15, 15, 15, 15); //sets the margins
                        divider.setLayoutParams(div);
                        linearLayout.addView(divider); //add it to the layout
                        //It's an empty image, just to create spacing between the
items
                    }
                }
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {
            }
        });

    }
    @Override
    public void onBackPressed() {//To prevent users from pressing the back button
on their phone
    }
}
```

This achieves objective 1eiv

View History 2 Page

This achieves objective 1eiv

```
package com.nadramon.test92;

import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;

import java.util.ArrayList;

public class ViewHistory extends AppCompatActivity {

    //Define variable types
    ArrayList<String> DItems;
    ArrayList<String> DPrices;
    ArrayList<String> DTPrices;
    ArrayList<String> DQuantities;
    String DTCost;
    String Name;
    String Username;
    String Password;
    String Region;
    String Street;
    String House;
    String Title;
    int First;
    TextView Heading;
    int length;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_history); //Create page with
specified layout

        Intent intent = getIntent();
        //Retrieves values from previous activity
        Name = intent.getExtras().getString("Name");
        Username = intent.getExtras().getString("Username");
        Password = intent.getExtras().getString("Password");
        Region = intent.getExtras().getString("Region");
        Street = intent.getExtras().getString("Street");
        House = intent.getExtras().getString("House");
        First = intent.getExtras().getInt("First");
        DItems = intent.getStringArrayListExtra("DItems");
        DPrices = intent.getStringArrayListExtra("DPrices");
        DTPrices = intent.getStringArrayListExtra("DTPrices");
        DQuantities = intent.getStringArrayListExtra("DQuantities");
        DTCost = intent.getExtras().getString("DTCost");
        Title = intent.getExtras().getString("Title");

        //set the message title of the page
        Heading = (TextView) findViewById(R.id.VHsentence);
        Heading.setText(Title);

        //Get the size of items in the lists
```

```
        length = DItems.size();

        TableLayout tableLayout = (TableLayout)
findViewById(R.id.TableLayoutID2);//Get XML reference for table layout
        //set layout parameters
        TableRow.LayoutParams textLayout = new
TableRow.LayoutParams(TableRow.LayoutParams.FILL_PARENT,
TableRow.LayoutParams.WRAP_CONTENT, 1);

        //make a new table row (first row)
        TableRow Frow = new TableRow(this);
        //set layout parameter
        Frow.setLayoutParams(new
TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT,
TableRow.LayoutParams.WRAP_CONTENT));
        Frow.setWeightSum(4);//have 4 columns
        Frow.setOrientation(TableRow.VERTICAL); //make it go across

        //make new textviews
        TextView ItemT = new TextView(this);
        TextView QtyT = new TextView(this);
        TextView PriceT = new TextView(this);
        TextView TotalT = new TextView(this);

        ItemT.setText(R.string.item);//set the text
        ItemT.setTextSize(18);//set the text size
        ItemT.setTextColor(Color.BLACK);//set the font colour
        ItemT.setLayoutParams(textLayout);//set the layout parameters
        ItemT.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);//set the alignment
        ItemT.setRight(1);

        QtyT.setText(R.string.qty);//set the text
        QtyT.setTextSize(18);//set the text size
        QtyT.setTextColor(Color.BLACK);//set the font colour
        QtyT.setLayoutParams(textLayout);//set the layout parameters
        QtyT.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);//set the alignment
        QtyT.setRight(1);

        PriceT.setText(R.string.price);//set the text
        PriceT.setTextSize(18);//set the text size
        PriceT.setTextColor(Color.BLACK);//set the font colour
        PriceT.setLayoutParams(textLayout);//set the layout parameters
        PriceT.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);//set the alignment
        PriceT.setRight(1);

        TotalT.setText(R.string.total);//set the text
        TotalT.setTextSize(18);//set the text size
        TotalT.setTextColor(Color.BLACK);//set the font colour
        TotalT.setLayoutParams(textLayout);//set the layout parameters
        TotalT.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);//set the alignment
        TotalT.setRight(1);

        //add the textviews into each column
        Frow.addView(ItemT);
        Frow.addView(QtyT);
        Frow.addView(PriceT);
        Frow.addView(TotalT);
        tableLayout.addView(Frow);//add this row to the table
```

```
        //for each item in the list
        for (int x = 0; x < length; x++) {

            //same like above, just for the middle rows
            TableRow row = new TableRow(this);
            row.setLayoutParams(new
TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT,
TableRow.LayoutParams.WRAP_CONTENT));
            row.setWeightSum(4);
            row.setOrientation(TableRow.VERTICAL);

            TextView item = new TextView(this);
            TextView qty = new TextView(this);
            TextView price = new TextView(this);
            TextView totalprice = new TextView(this);


            item.setText(DItems.get(x));
            item.setTextSize(14);
            item.setTextColor(Color.BLACK);
            item.setTextAlignment(View.TEXT_ALIGNMENT_TEXT_START);
            item.setLayoutParams(textLayout);
            item.setRight(1);

            qty.setText(DQuantities.get(x));
            qty.setTextSize(14);
            qty.setTextColor(Color.BLACK);
            qty.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
            qty.setLayoutParams(textLayout);
            qty.setRight(1);

            price.setText(DPrices.get(x));
            price.setTextSize(14);
            price.setTextColor(Color.BLACK);
            price.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
            price.setLayoutParams(textLayout);
            price.setRight(1);

            totalprice.setText(DTPrices.get(x));
            totalprice.setTextSize(14);
            totalprice.setTextColor(Color.BLACK);
            totalprice.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
            totalprice.setLayoutParams(textLayout);
            totalprice.setRight(1);


            row.addView(item);
            row.addView(qty);
            row.addView(price);
            row.addView(totalprice);

            tableLayout.addView(row);


        }

        //same as above, just for last row
        TableRow Lrow = new TableRow(this);
        Lrow.setLayoutParams(new
```

```
TableRow.LayoutParams(TableRow.LayoutParams.MATCH_PARENT,
TableRow.LayoutParams.WRAP_CONTENT));
        Lrow.setWeightSum(1); //only 1 column
        Lrow.setOrientation(TableRow.VERTICAL);

        TextView costT = new TextView(this);

        final String TotalCost = "Total Cost: " + DTCost + " Rials";

        costT.setText(TotalCost);
        costT.setTextSize(14);
        costT.setTextColor(Color.BLACK);
        costT.setTextAlignment(View.TEXT_ALIGNMENT_TEXT_END);
        costT.setLayoutParams(textLayout);
        costT.setRight(1);

        Lrow.addView(costT);

        tableLayout.addView(Lrow);

    }

    public void Back(View view) {
        Intent Back = new Intent(this, OrderHistory.class);
        //Pass these values to the next page
        Back.putExtra("Name", Name);
        Back.putExtra("Username", Username);
        Back.putExtra("Password", Password);
        Back.putExtra("Region", Region);
        Back.putExtra("Street", Street);
        Back.putExtra("House", House);
        Back.putExtra("First", First);
        startActivity(Back); //open new page
        finish();//close current page
    }

    @Override
    public void onBackPressed() {//To prevent users from pressing the back button
on their phone
    }
}
```

Testing

Test Plan

I will describe what the outcome of the test is. There are also videos and screenshots found to support my claims.

Customer App

Log In Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
1	To test the entry fields function properly (upon pressing the log in button)	Expected Data	Correct username (regardless of upper/lower case) and correct password input	Logging in success, taken to home page
		Erroneous Data	Correct username and incorrect password input	Error Message, saying either username or password is incorrect
		Erroneous Data	Incorrect username and correct password input	Error Message, saying either username or password is incorrect
		Erroneous Data	Incorrect username and incorrect password input	Error Message, saying either username or password is incorrect
		Erroneous Data	One or more fields is left blank	Error Message, saying that a field was left blank
2	To test the navigation buttons works	Expected Data	Short/Long press	Log In: Takes user to the Home Page Sign Up: Takes user to the Sign Up Page

Sign Up Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
3	To test the entry fields function properly (upon pressing the sign	Expected Data	Every field (including radio button) is filled, password field is at least 6 characters	Account created and stored in database, taken to sign up complete page

	up button)	Erroneous Data	1 or more fields is left blank	Error Message, saying that a field was left blank
		Erroneous Data	Password field is filled with 5 characters or less	Error Message, saying the password must at least be 6 characters
		Erroneous Data	Type a username that already exists on the database	Error Message, saying that the username is already in use
4	To test the radio buttons function properly	Expected Data	Option A is selected	Option A will be selected
		Expected Data	Option B is selected after Option A was selected	Option B will be selected and Option A will be unselected
5	To test the info buttons function properly	Expected Data	Info button is pressed	Password info should say “minimum of 6 characters required”. House and Street info should give examples. Region info should say why only the two options is availabl.
6	To test the navigation buttons functions	Expected Data	Short/Long press	Arrow: Takes user back to the Log In page Sign Up: Takes user to the Sign Up Complete Page

Sign Up Complete Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
7	To test the navigation button functions	Expected Data	Short/Long press	Continue to Log In: Takes user to the Log In Page

Home Page

Test Plan	Purpose	Test Type	Test Data	Expected Outcome
8	To test exiting the app completely	Expected Data	Short/Long press on button	Closes app

	works			
9	To test all navigation buttons function	Expected Data	Short/Long press on button	Make an Order: Takes users to the Make an Order Page Edit Account: Takes users to the Edit Account Page View Order History: Takes users to the View Order History Page
10	To test that you can only press the Make an Order button between 9AM to 9PM	Expected Data	Short/Long press on button, between 9AM and 9PM	Takes users to the Make an Order Page
		Erroneous Data	Short/Long press on button, Before 9AM and after 9PM	Error Message, saying that you can't order at this hour
11	To test the welcome message uses the user's name in it	Expected Data	Log in with a few accounts	For each account their name that was stored in the database should appear
12	To test that the "Logging In Success" message doesn't appear when coming from pages that is not the Log In Page	Expected Data	Change page to the Home Page from the 3 avaliable pages	The message will not appear.

Edit Account Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
13	To test that clicking the UPDATE button will summon a	Expected Data	Short/Long press	A dialog box appearing asking for password confirmation

	Dialog Box			
14	To test that the CANCEL and UPDATE button in the notify dialog box works	Expected Data	Short/Long press	CANCEL – Closes dialog box
		Expected Data	Short/Long press, password typed correctly	UPDATE – Popup saying the account has been updated, takes user back to the home page
		Erroneous Data	Short/Long press, password typed incorrectly	UPDATE – Popup saying the password was incorrect
15	To test the entry fields function properly (upon pressing the UPDATE button)	Expected Data	Every field (including radio button) is filled, password field is at least 6 characters	Database values are updated
		Erroneous Data	1 or more fields is left blank	Error Message, saying that a field was left blank
		Erroneous Data	Password field is filled with 5 characters or less	Error Message, saying the password must at least be 6 characters
16	To test the radio buttons function properly	Expected Data	Option A is selected	Option A will be selected
		Expected Data	Option B is selected after Option A was selected	Option B will be selected and Option A will be unselected
17	To test the entry fields has the user's current data written in them (except for password, including radio button option)	Expected Data	Log In with a few accounts	Each field should have the user's data written in them
18	To test all navigation buttons function	Expected Data	Short/Long press on button	Arrow/Update: Takes users to the Home Page

Make an Order Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
19	To test the navigation button functions	Expected Data	Short/Long press	Arrow: Takes users to the Home Page Check Order: Takes users to the Confirm Order Page
20	To test the spinner selection is correct (upon pressing the enter button)	Expected Data	Select a couple of categories, cross-reference to the database	Each category chosen should get a list of every food item in that category
21	To test if users can proceed to the Check Orders Page	Expected Data	Made 1 or more selection of orders and then press button	Taken to the Confirm Order Page with selected Orders
		Erroneous Data	Made no selections at all and then press button	Error Message, saying that the user had not made any selections
22	To test if pressing an item name will create a dialog box	Expected Data	Press a few item names	A dialog box appearing
23	To test that the food item has the correct price alongside it	Expected Data	Open up with a few food items, cross check to the database	The correct price is retrieved for each one
24	To test that users can only type integers when asked for how much portion is wanted (upon pressing OK)	Expected Data	Typing a number such as 7	7 is typed in the entry field and the item is added to the list
		Erroneous Data	Typing a character that's not a number	Not Applicable as the keyboard will be default to numbers only
		Extreme Data	Typing a float value such as 5.6	Not Applicable as the keyboard will not have a full

				stop
		Extreme Data	Typing a negative number such as -5	Not Applicable as the keyboard will not have a minus sign
		Extreme Data	Typing 0	Error Message, saying you can't order zero portions of something
25	To test that the CANCEL and OK button in the dialog box works	Expected Data	Short/Long press	OK – Closes dialog box, saves selected item to list, popup saying it has been added CANCEL – Closes dialog box

Confirm Order Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
26	To test the navigation button functions	Expected Data	Short/Long press	Arrow: Takes users to the Make an Order Page Restart Order: Takes users to the Make an Order Page
27	To test the arrow button keeps the user's selected orders	Expected Data	Press arrow button and then press check order	List of selected orders still present
28	To test the Restart Order button gets rid of the user's selected orders	Expected Data	Press the restart order button and then press check order	Error Message, saying that the user had not made any selections
29	To test that the calculations are done correctly	Expected Data	View page	Quantity * Price = Total. Sum of each Total = Total Cost.
30	To test that the	Expected	Short/Long press	A dialog box asking for

	Confirm Order button will create a dialog box	Data	button	password confirmation appearing
31	To test that typing the password for confirmation functions properly	Expected Data	Correct password input	Order will be uploaded to database
		Erroneous Data	Incorrect password input	Error Message, saying password is incorrect
32	To test that the CANCEL and SEND ORDER button in the dialog box works	Expected Data	Short/Long press	SEND ORDER – Uploads order to database, takes user to the Order Complete Page CANCEL – Closes dialog box

Order Complete Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
33	To test the navigation button functions properly	Expected Data	Short/Long press	Continue to Home: Takes user to the Home Page

View Order History Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
34	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page
35	To test the spinner selection is correct (upon pressing the search button)	Expected Data	Select a month and year which the user have ordered in	Orders from that time period should be listed
		Erroneous	Select a month and year which the user	Error Message, saying the user has not ordered

		Data	have not ordered in	anything in the month and year chosen
36	To test that an order can be clicked	Expected Data	Short/Long press	Takes the user to the View Order History 2 Page
37	To test that when it is a new year it is automatically updated in the spinner	Expected Data	View the spinner selections	There should be options from 2013 to the current year.

View Order History 2 Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
38	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the View Order History Page
39	To test that the data downloaded is all correct	Expected Data	Do this with several orders, cross-check in database	All data correct

Worker App

Home Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
40	To test exiting the app completely works	Expected Data	Short/Long press on button	Closes app
41	To test all navigation buttons	Expected	Short/Long press on	Pending Orders: Takes users to the pending orders page

	function	Data	button	View Menu: Takes users to the view menu page
42	Clicking the edit menu, notify all and delete accounts button should prompt a password confirmation	Expected Data	Short/Long press on button	A dialog box appearing asking to input password
43	To test that the CANCEL and CONFIRM button in the dialog box works	Expected Data	Short/Long press	CANCEL – Closes dialog box
		Expected Data	Short/Long press, password typed correctly	CONFIRM – Takes users to the respective page, dialog box closes
		Erroneous Data	Short/Long press, password typed incorrectly	CONFIRM – Popup saying the password was incorrect
44	Text stating how many orders is pending should update in real time	Expected Data	Have home page screen open at the same time as someone making an order on the customer app	The number should increase by one
		Expected Data	Have home page screen open at the same time as someone completing a pending order on the worker app	The number should decrease by one

Pending Orders Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
45	To test the navigation button	Expected Data	Short/Long press	Arrow: Takes user to the Home Page User's order: Takes user to the

	functions properly			View Order Page
46	Text stating how many orders is pending should update in real time	Expected Data	Have pending order screen open at the same time as someone making an order on the customer app	The number should increase by one
		Expected Data	Have pending order screen open at the same time as someone completing a pending order on the worker app	The number should decrease by one
47	Clickable text of a user's orders should update in real time	Expected Data	Have pending order screen open at the same time as someone making an order on the customer app	A new text should be added at the bottom of the existing list
		Expected Data	Have pending order screen open at the same time as someone completing a pending order on the worker app	That order should disappear from the list

View Order Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
48	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page
49	To test that the item, quantity and cost retrieved is correct	Expected Data	Cross-check to the database	Should be correct

50	The user's details button should prompt a dialog box stating the user's name and address only	Expected Data	Short/Long press	Name, Region, Street Address and House Address should be displayed
51	To test that the GOTCHA button in the user's details dialog box works	Expected Data	Short/Long press	GOTCHA – Closes dialog box
52	The notify button should prompt a dialog box for sending notification	Expected Data	Short/Long press	Default messages and a custom message with an entry field should be present
53	To test that the CANCEL button in the notify dialog box works	Expected Data	Short/Long press	CANCEL – Closes dialog box
54	To test that pressing the message options will send a notification to the user	Expected Data	Short/Long press, watch the phone of the user who made the order	A notification appearing in the user's phone
		Erroneous Data	When pressing the custom message but the entry field is empty	Error message saying there is no message written
55	The order completed button should prompt a dialog box asking for password confirmation	Expected Data	Short/Long press	A dialog box appearing with an entry field for the password of the user who made the order
56	To test that the CANCEL and CONFIRM button in	Expected Data	Short/Long press	CANCEL – Closes dialog box
		Expected	Short/Long press,	CONFIRM – Popup saying

	the notify dialog box works	Data	password typed correctly	order has been completed, takes user back to the pending order page, the order that was completed should have disappeared from the list
		Erroneous Data	Short/Long press, password typed incorrectly	CONFIRM – Popup saying the password was incorrect

View Menu Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
57	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page
58	To test that selecting an item category from the spinner will produce the correct list of items	Expected Data	Select category and press the ENTER button	List of items from the correct category produced along with its price

Edit Menu Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
59	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page Add Item: Takes user to the Add Item Page Edit Item: Takes user to the Edit Item Page Delete Item: Takes user to the Delete Item Page

Add Item Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
60	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Edit Menu Page
61	To test the entry fields function properly (upon pressing the confirm button)	Expected Data	All entry fields are filled in (inluding the spinner option)	Popup saying the new item is added, added to the database, the page is then refreshed
		Erroneous Data	1 or more entry field is not filled in	Popup saying that 1 or more field is missing

Edit Item Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
62	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Edit Menu Page
63	To test that selecting an item category from the spinner will produce the correct list of items	Expected Data	Select category and press the ENTER button	List of items from the correct category produced
64	To test that the produced list of items is clickable	Expected Data	Short/Long press	User will be taken to the Edit Item 2 page with the correct details of the item selected

Edit Item 2 Page

Test	Purpose	Test Type	Test Data	Expected Outcome

Number				
65	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Edit Item Page
66	To test that the entry field has the current data for the item as text	Expected Data	Repeat a few times with different items and cross-check in database	The correct item name and price should be text values in the entry field
67	To test that the confirm button works	Expected Data	Short/Long press	The database is updated with the new values, the user will be taken to the Edit Item Page and a popup saying it was a success should appear
		Erroneous Data	Short/Long press but one or more field is empty	Error message saying a field was left blank

Delete Item Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
68	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Edit Menu Page
69	To test that selecting an item category from the spinner will produce the correct list of items	Expected Data	Select category and press the ENTER button	List of items from the correct category produced
70	To test that the produced list of items is clickable	Expected Data	Short/Long press	A dialog box will prompt asking for confirmation of the deletion

71	To test that the CANCEL and CONFIRM button in the dialog box works	Expected Data	Short/Long press	CONFIRM – a popup saying the item was deleted will appear, and the dialog box will close CANCEL – Closes dialog box

Notify All Page

Test Number	Purpose	Test Type	Test Data	Expected Outcome
72	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page
73	To test that entering nothing will prompt an error message	Expected Data	Leave the entry field blank and press SEND	An error message appearing saying there was nothing written
74	To test that all the customers gets the notification	Expected Data	Type a message and press the SEND button	All users should get the notification

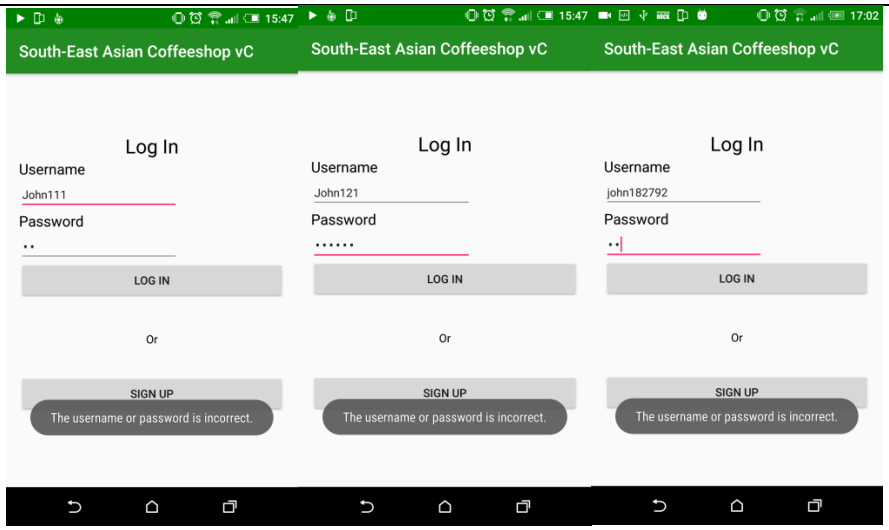
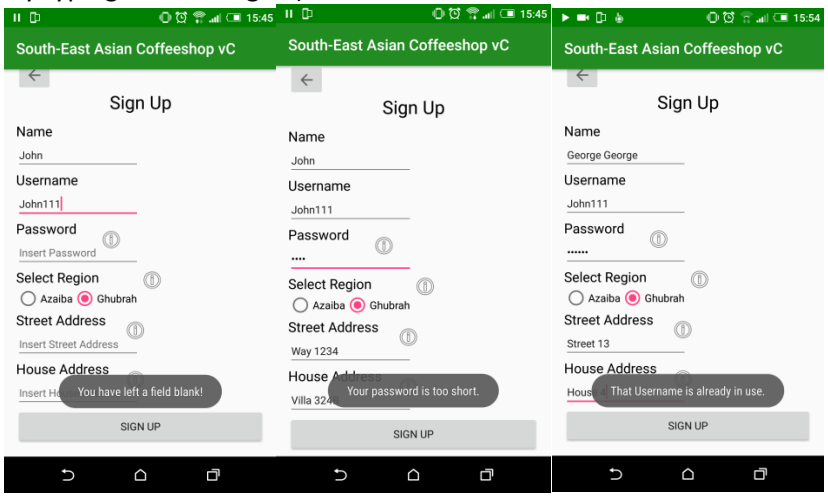
Delete Account Page

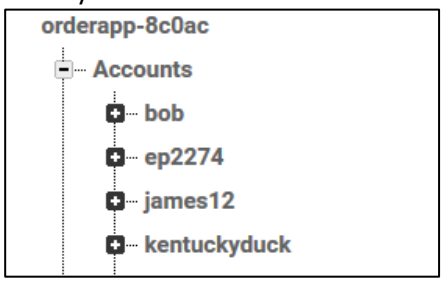
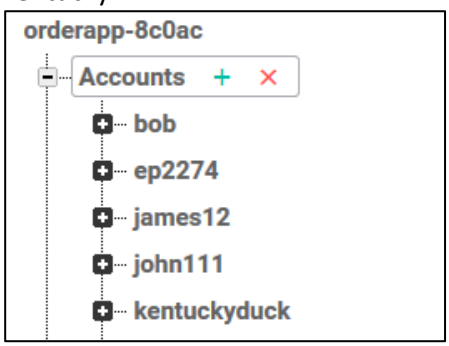
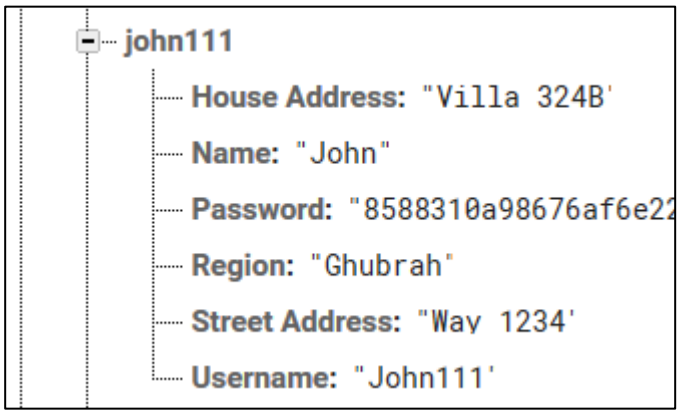
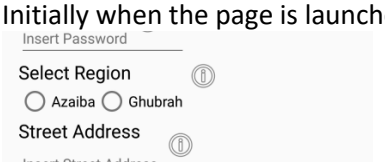
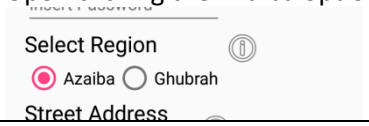
Test Number	Purpose	Test Type	Test Data	Expected Outcome
75	To test the navigation button functions properly	Expected Data	Short/Long press	Arrow: Takes user to the Home Page
76	To test the page functions properly	Expected Data	If an account has not ordered in more than one year	That account should appear on the page and is clickable for deletion
		Expected Data	If there is no account that has not ordered more than one year	A text should say that there is no inactive accounts

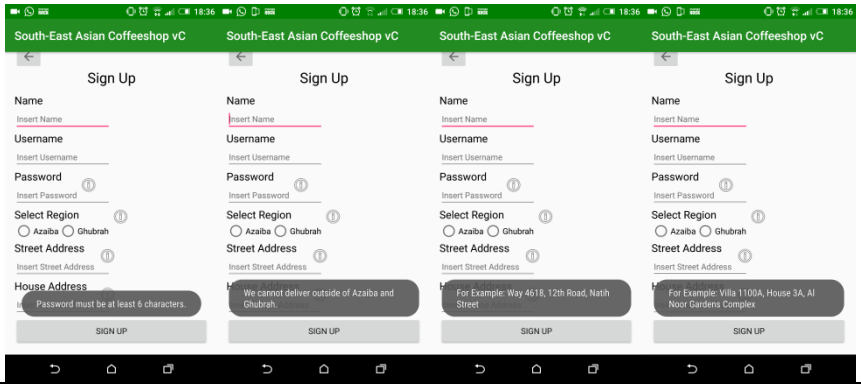
77	To test that clicking on a name will prompt a dialog box	Expected Data	Short/Long press	A dialog box appearing
78	To test that the CONFIRM and CANCEL buttons in the dialog box works	Expected Data	Short/Long press	CONFIRM – Deletes the account from the database, account details and order history, popup saying the deletion is a success and the page is refreshed CANCEL – closes the dialog box

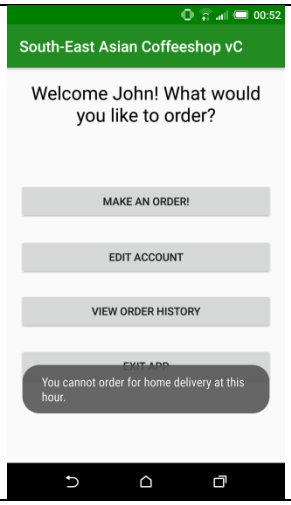
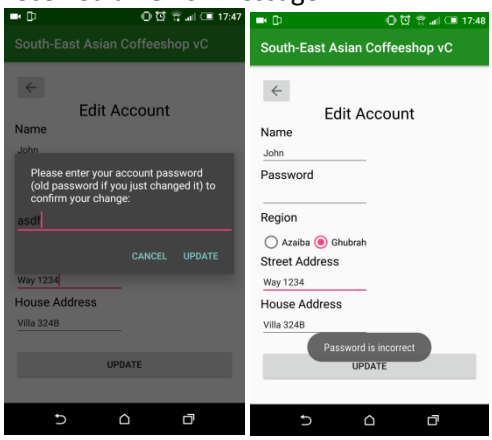
Test Results

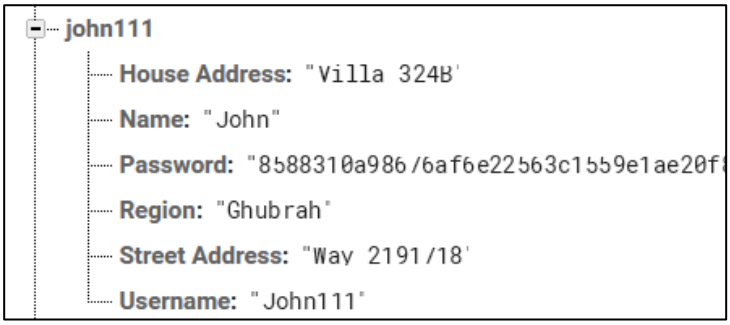
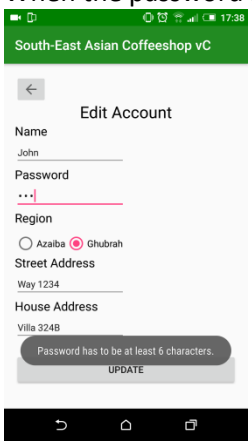
Test Number	Success?	Results and Evidence
1	Yes	When logging in with the correct username and wrong password, I received an error message. (The actual username is John111) When logging in with the wrong username and correct password, I received an error message. (The password is not even 6 letters) When logging in with both username and password incorrect, I received an error message. (Both Wrong)

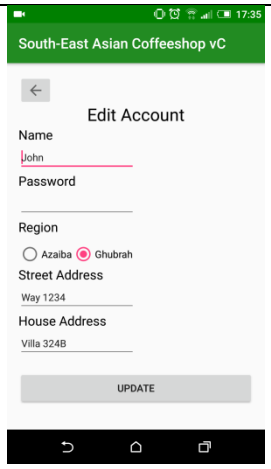
		 When logging in with the correct username and password (regardless of upper/lower case for the username) I was successfully taken to the homepage. The evidence for this can be found in the video “Video 1.mp4”
2	Yes	Pressing the Log in button takes me to the home page. Pressing the Sign Up button takes me to the Sign Up Page The evidence for this can be found in the video “Video 2.mp4”
3	Yes	When 1 or more field was left blank I received an error message. When the password field is less than 6 characters I received an error message. When I type a username that already exists I received an error message. (I try typing John111 again)  When all the fields are filled in and the password is at least 6 characters, I was taken to the sign up complete page (The evidence for this can be found in the video “Video 3.mp4”), and the account data was added to the database:

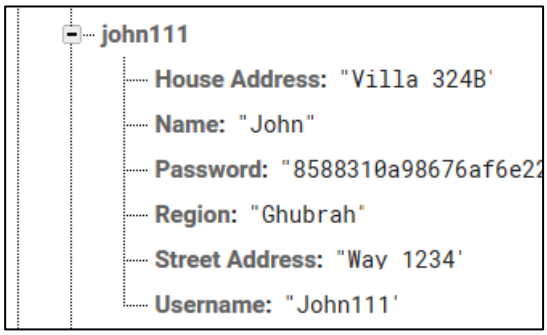
		Initially the database had these accounts:  Then after I made the account john111 appeared between james and kentucky:  Here you can see the data that was saved in the database, they are the same as what I typed in the app, except that the password is stored as the hashed value. 
4	Yes	Initially when the page is launched neither region is selected  Upon clicking the Azaiba option it became selected. 

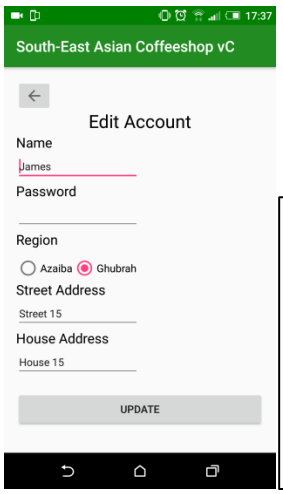
		Upon clicking the Ghubrah option it became selected AND the Azaiba option became unselected. 
5	Yes	Pressing the info buttons pops up a help message for guidance First one for the password info, second one for the region info, third one for the street info and the last one for the house info 
6	Yes	Pressing the arrow button did take me to the Log In Page, The evidence for this can be found in the video “Video 4.mp4”. Pressing the Sign Up button did take me to the Sign Up Complete Page, The evidence for this can be found in the video “Video 3.mp4”.
7	Yes	Pressing the button did take me to the Log In Page, The evidence for this can be found in the video “Video 5.mp4”.
8	Yes	Upon clicking the Exit App button I was taken to my home screen on my phone, The evidence for this can be found in the video “Video 6.mp4”.
9	Yes	Each button took me to the correct pages, The evidence for this can be found in the video “Video 7.mp4”.
10	Yes	I was able to go to the page in the allowed hours, the evidence for this can be found in the video “Video 7.mp4”. I was not able to go to the page in the disallowed hours, as you can see I tried accessing the page at 00:52 and I received a popup instead:

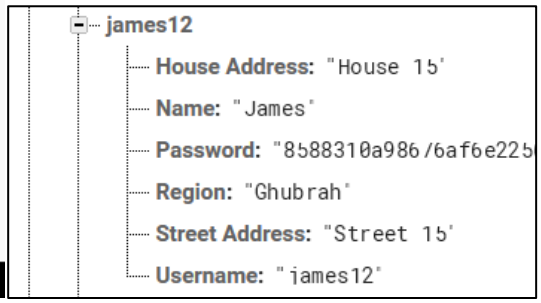
		
11	Yes	Every time I logged on with a different account, they got the correct name in the welcome message. The evidence for this can be found in the video "Video 8.mp4".
12	Yes	When I go to the Homepage from the Make an order page, edit account page, and view order history page, the message did not appear. The evidence for this can be found in the video "Video 9.mp4".
13	Yes	When I click the UPDATE button a dialog box does indeed appear asking for a password confirmation. The evidence for this can be found in the video "Video 10.mp4".
14	Yes	When I click the CANCEL button the dialog box simply closes. The evidence for this can be found in the video "Video 10.mp4". When I click the UPDATE button but the password I typed in is incorrect, I received an error message:  When I click the UPDATE button and the password I typed is correct, I get taken to the home page and a popup appears saying the account has been updated. The evidence for this can be found in the video "Video 10.mp4".

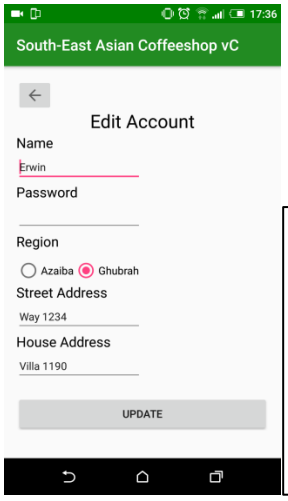
15	No	When every field is filled, the account gets updated and the user gets taken to the home page. The evidence for this can be found in the video “Video 10.mp4”. As you can see the data in the database has updated, John’s street address has changed:  When the password is less than 6 characters, an error message pops up.  However when a field (excluding the password field) is left blank, the account still gets updated and the user taken to the home page. This is simply because I have forgotten to error trap this. The evidence for this can be found in the video “Video 11.mp4”.
16	Yes	This functions exactly like Test 4, please see the example given in Test 4, since it has been tested in test 4, it must also work here.
17	Yes	Each time I open the Edit Accounts page the correct data is set in the EditTexts. For John:

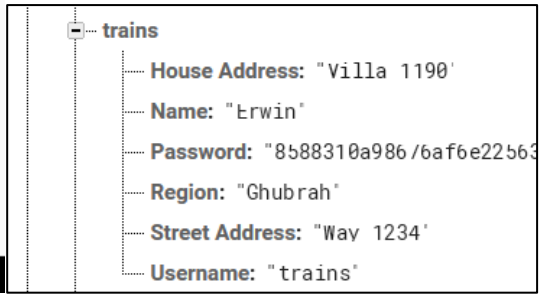
<table>
<tr>
<td></td>
<td></td>
<td>




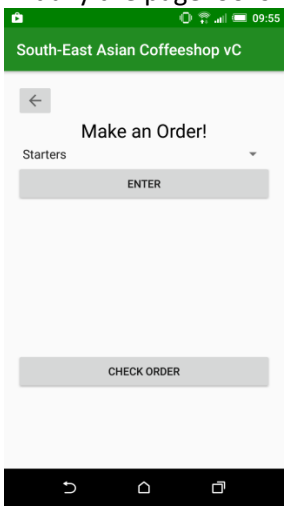
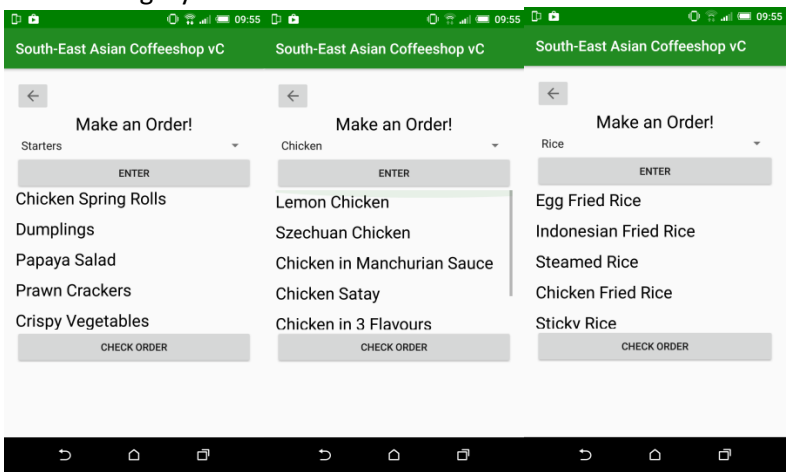
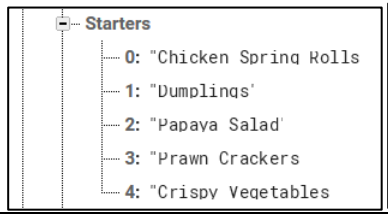
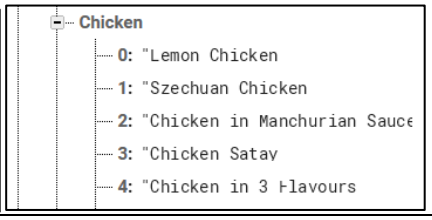
For James:


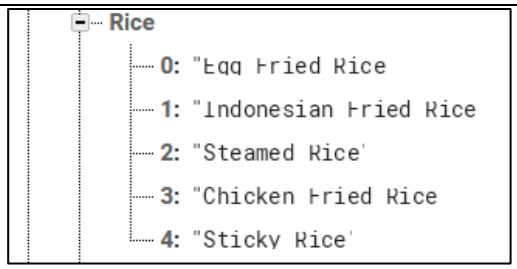
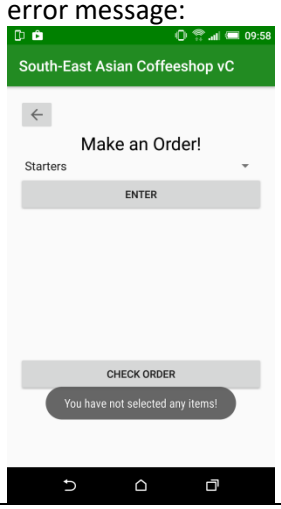
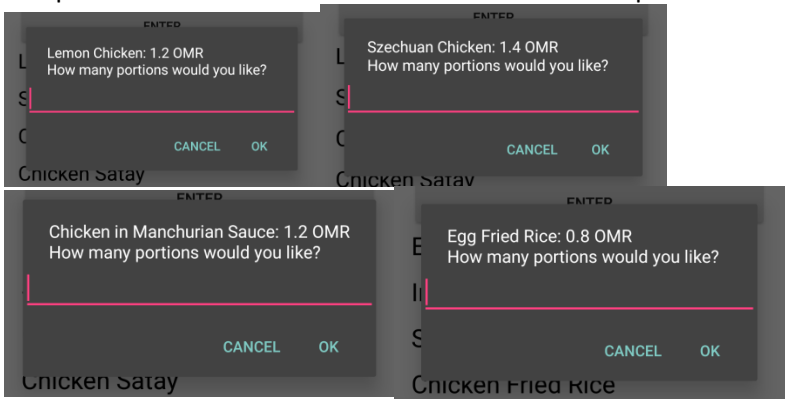


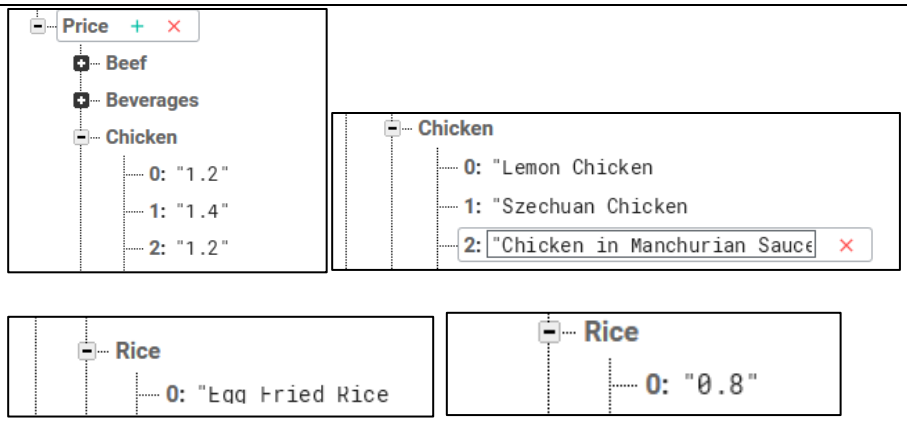
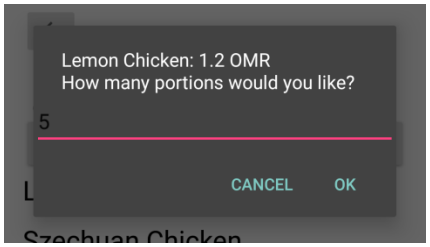
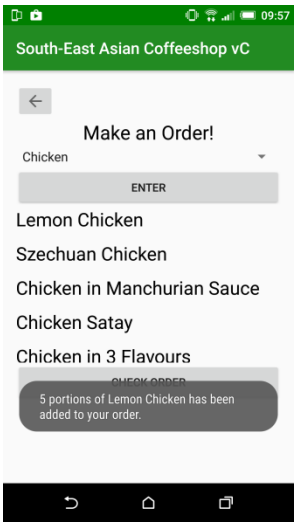
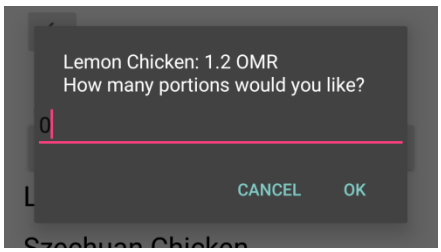
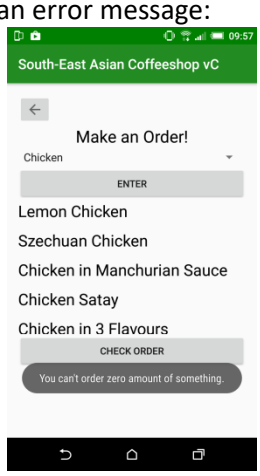
For Erwin:


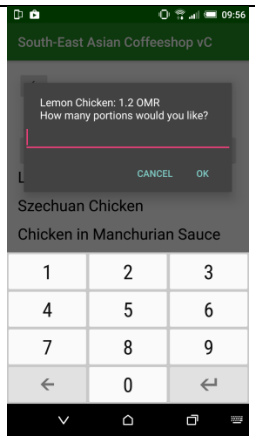


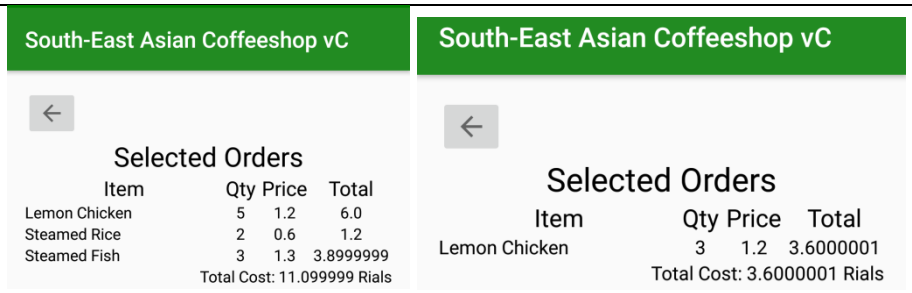
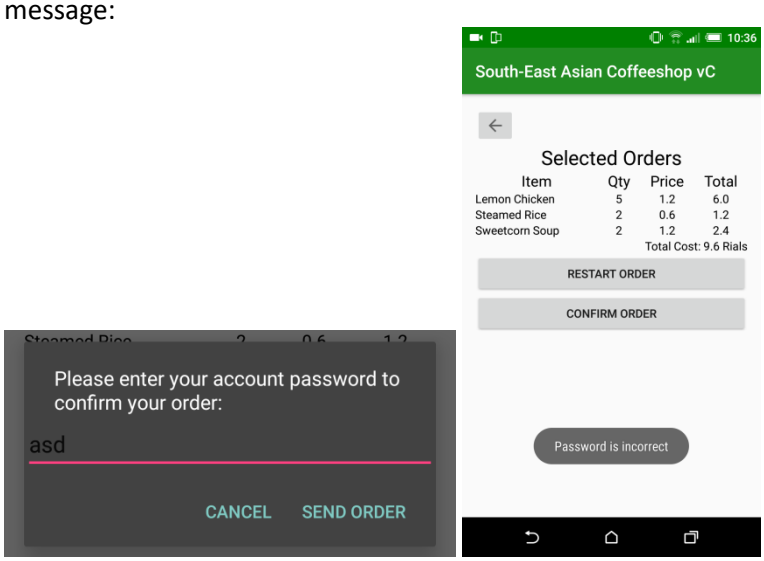
As you can see the data is exactly the same in the edit accounts page as it is on the database.
</td>
</tr>
<tr>
<td>18</td>
<td>Yes</td>
<td>Navigation buttons act exactly the same for every single page, the arrow</td>
</tr>
</table>

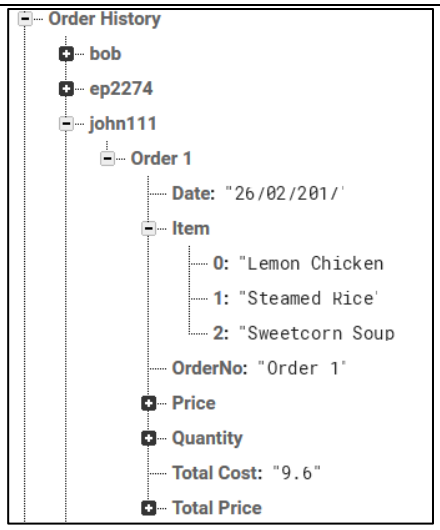
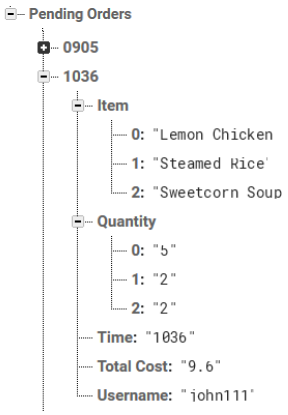
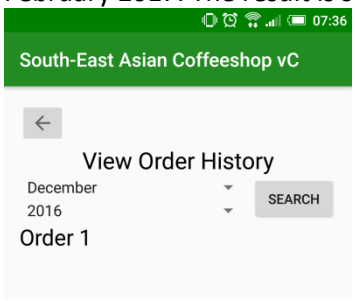
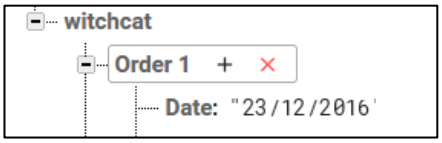
		button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
19	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
20	Yes	Initially the page looks like this:  Upon choosing the category and pressing the button, I get a list of the items in that category:  As you can see, the list produced is correct as seen from the database:  

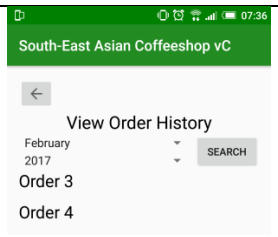
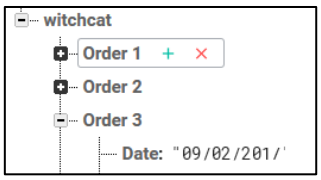
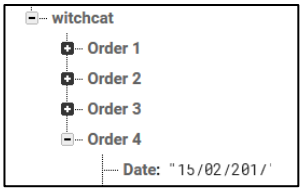
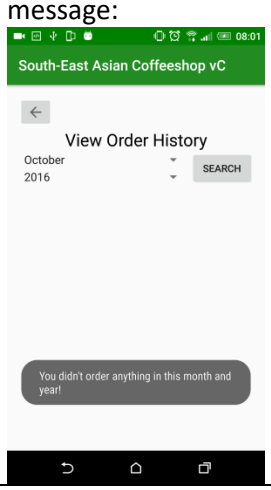
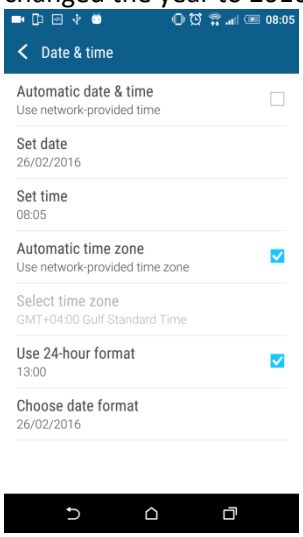
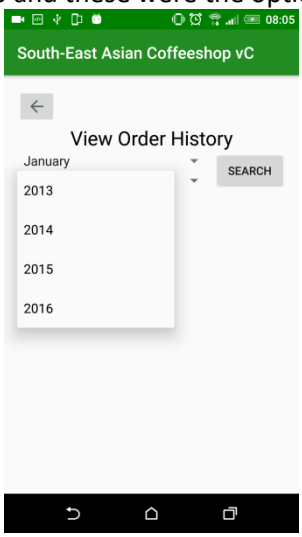
		
21	Yes	When I pressed the button with 1 or more selections, I was taken to the Confirm Order Page. The evidence for this can be found in the video "Video 13.mp4". When I pressed the button without making any selections, I received an error message: 
22	Yes	Upon clicking an item name a dialog box did appear, asking me to input the desired quantity of the item. The evidence for this can be found in the video "Video 12.mp4".
23	Yes	The price of the item is correct for each item I have opened: 

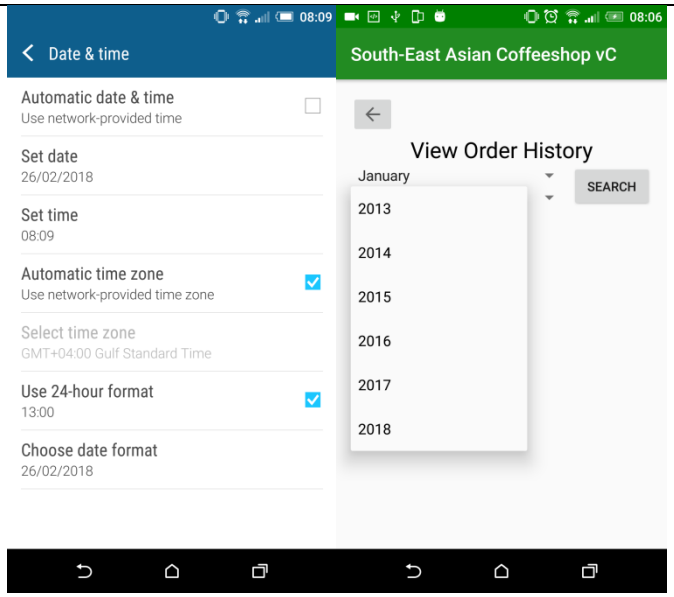
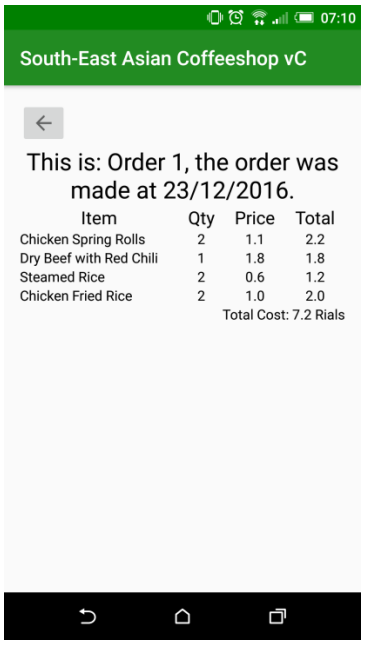
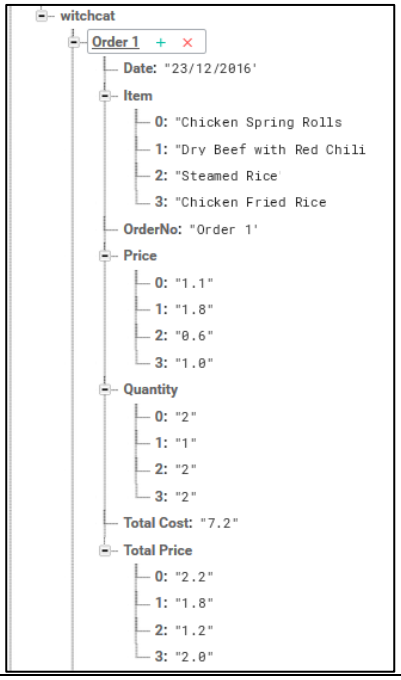
		
24	Yes	When I type a number greater than or equal to 1, I get a popup saying the item was added to the list:   When I type 0 for my quantity, I get an error message:   I am not able to type other characters, a decimal point or a negative number as the defaulted keyboard is numbers only.

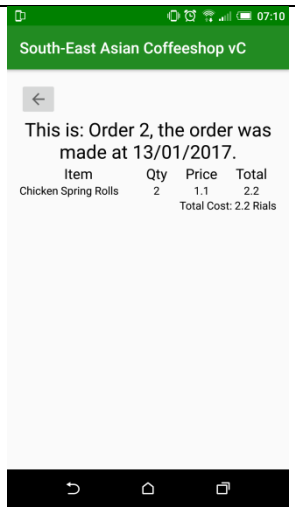
		
25	Yes	Pressing the cancel button simply closes the dialog box. The evidence for this can be found in the video “Video 12.mp4”. Pressing the OK button closes the dialog box and saves my selection to a list. The evidence for this can be found in the video “Video 13.mp4”.
26	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
27	Yes	When I clicked the back button I get taken to the make an order page, and following that when I click the check order button I get taken back to the check order page with my order still present. The evidence for this can be found in the video “Video 14.mp4”.
28	Yes	When I clicked the restart button I get taken to the make an order page, and following that when I click the check order button I get an error message saying I haven’t made any selections. The evidence for this can be found in the video “Video 15.mp4”.
29	Yes and No?	The calculations were done correctly. As you can see 5 * 1.2 does equal 6, 2 * 0.6 does equal 1.2 and finally 2 * 1.2 does equal 2.4. 6 + 1.2 + 2.4 does equal 9.6. Hence the calculations were done perfectly.  However, there seems to be a problem when the program is calculating with a quantity of 3:

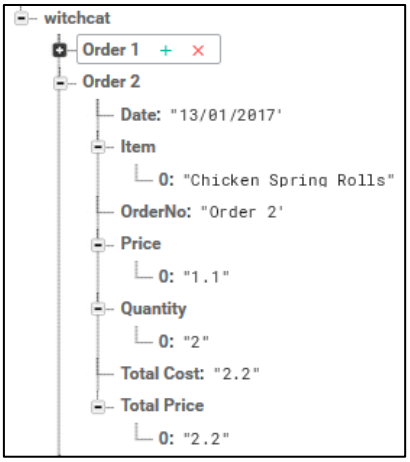
		 It seems to be some sort of float error.
30	Yes	Upon clicking the button a dialog box did appear, asking for my password confirmation. The evidence for this can be found in the video “Video 16.mp4”.
31	Yes	When I type my password wrong and try to send the order, I get an error message:  When I type my password correctly and send the order, I get taken to the order complete page and the data was uploaded in the database. The evidence for this can be found in the video “Video 17.mp4”.

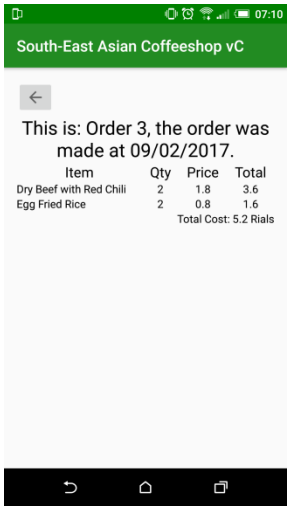
		  As you can see the data of the order was successfully uploaded to the database.
32	Yes	When I press the CANCEL button the dialog box simply closes. The evidence for this can be found in the video “Video 16.mp4”. When I press the SEND ORDER button and typed the correct password, I was taken to the order complete page and the data was uploaded to the database. The evidence for this can be found in test number 31.
33	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well. But there is evidence in the video “Video 18.mp4”.
34	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
35	Yes	Using the ‘witchcat’ account, I selected the date of December 2016 and February 2017. The result is successful as follows:   As you can see order 1 was ordered on the 12/2016 so it did appear when I searched for it.

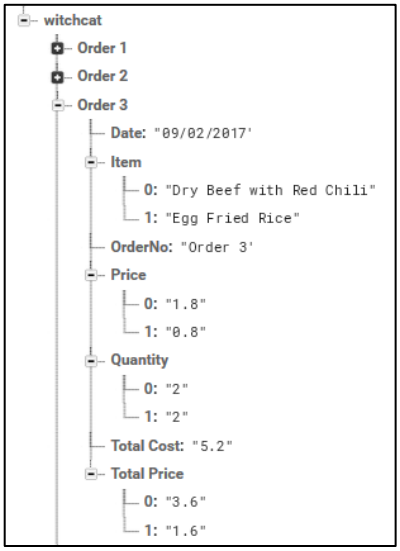
		   As you can see, order 3 and 4 were ordered on the 02/2017 (Believe me it is 2017 and not 201/ in the database) so it did appear when I searched for it. When I select a month and year that I didn't order in, I get an error message: 
36	Yes	Upon pressing an order number I was taken to the View Order History 2 page. The evidence for this can be found in the video "Video 19.mp4".
37	Yes	I have tested this by manually changing the year on my phone. Here I changed the year to 2016 and these were the options available:   Here I changed it to 2018 and these were the options available:

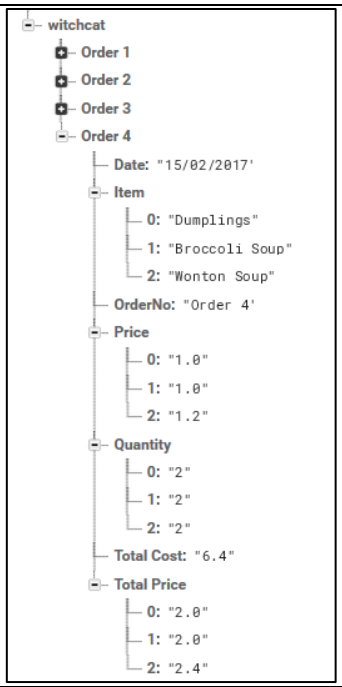
		 As you can see the spinner selections for updating each new year is a success!
38	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
39	Yes	I checked the order history for the account 'witchcat'. I made the orders in December 2016, January 2017 and February 2017. By checking the database I can see that the downloaded data are correct:  

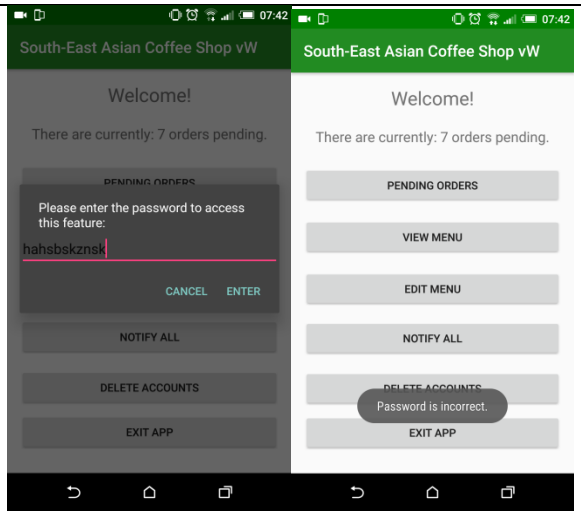

07:10
South-East Asian Coffeeshop vC
This is: Order 2, the order was made at 13/01/2017.
Item Qty Price Total
Chicken Spring Rolls 2 1.1 2.2
Total Cost: 2.2 Rials

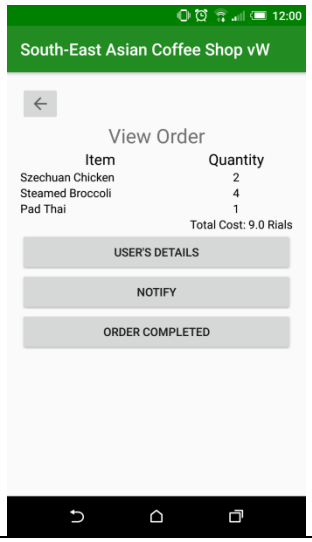
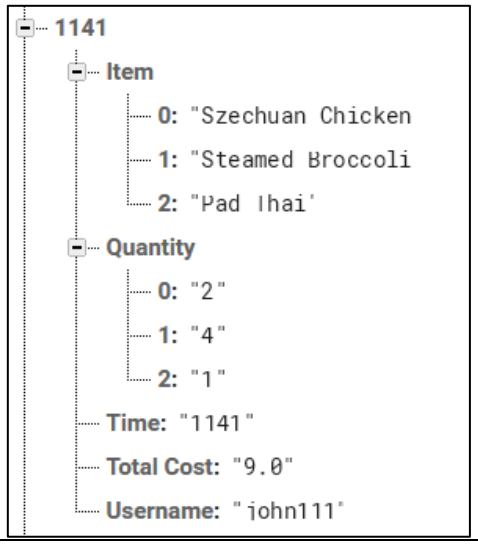
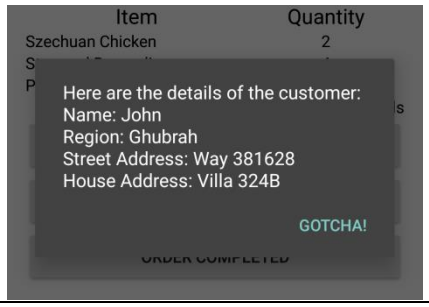
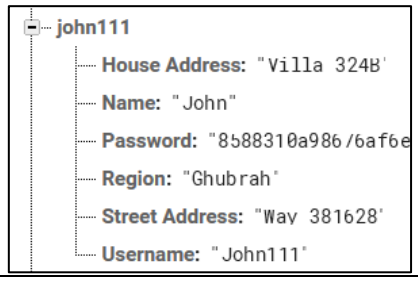

witchcat
Order 1
Order 2
Date: "13/01/2017"
Item
0: "Chicken Spring Rolls"
OrderNo: "Order 2"
Price
0: "1.1"
Quantity
0: "2"
Total Cost: "2.2"
Total Price
0: "2.2"

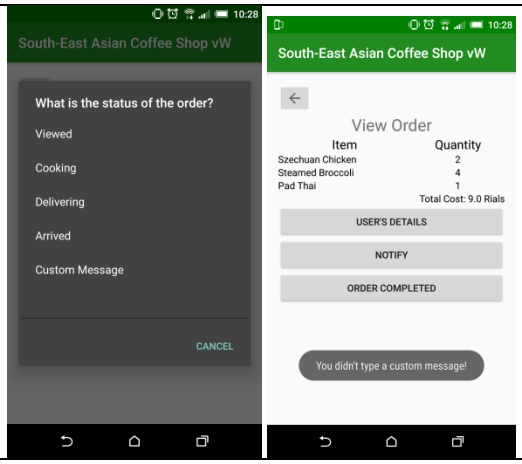
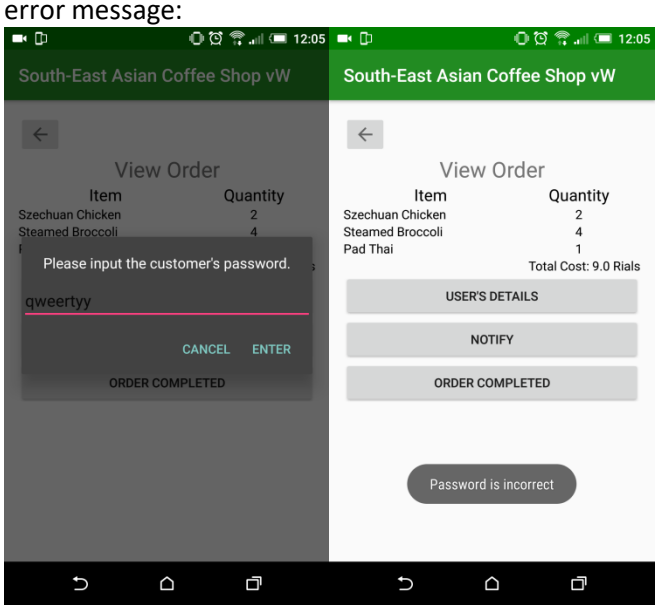

07:10
South-East Asian Coffeeshop vC
This is: Order 3, the order was made at 09/02/2017.
Item Qty Price Total
Dry Beef with Red Chili 2 1.8 3.6
Egg Fried Rice 2 0.8 1.6
Total Cost: 5.2 Rials

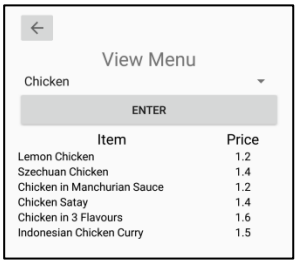
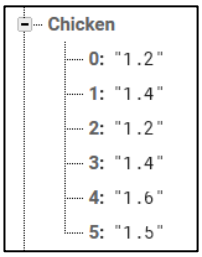
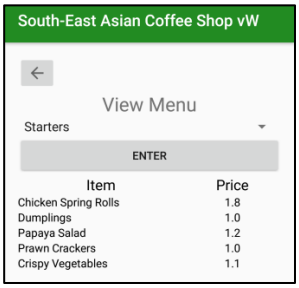
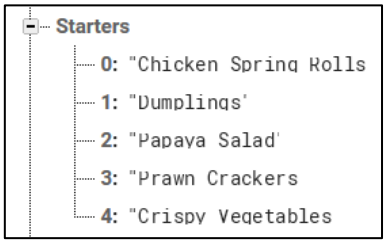
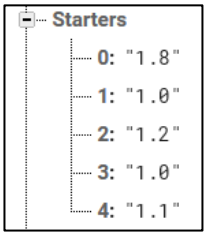
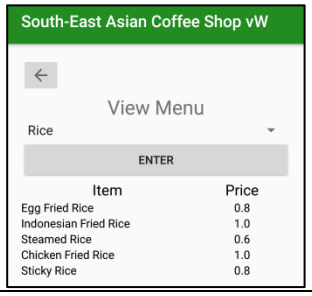
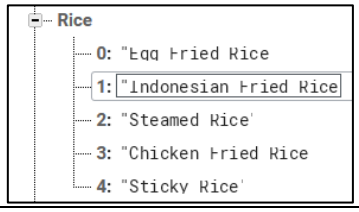
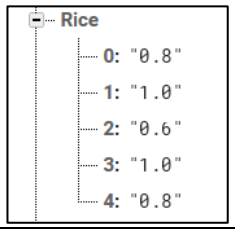

witchcat
Order 1
Order 2
Order 3
Date: "09/02/2017"
Item
0: "Dry Beef with Red Chili"
1: "Egg Fried Rice"
OrderNo: "Order 3"
Price
0: "1.8"
1: "0.8"
Quantity
0: "2"
1: "2"
Total Cost: "5.2"
Total Price
0: "3.6"
1: "1.6"

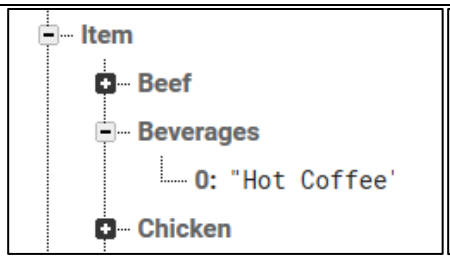
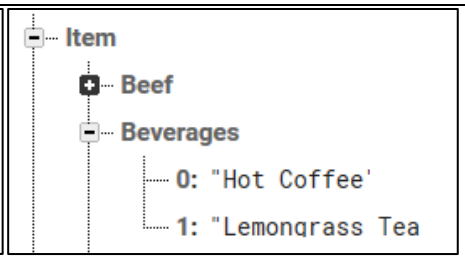
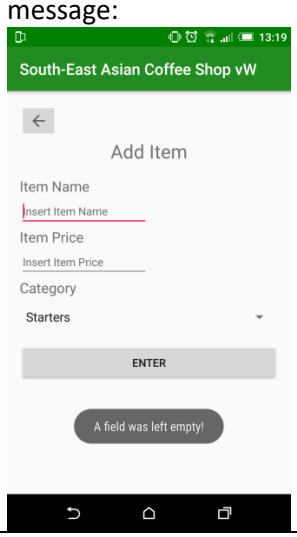
		 
40	Yes	Exiting the app functions exactly the same way it does in the customer app. It has been tested in Test number 8; hence it must also work since it is the same.
41	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
42	Yes	Upon clicking the specified buttons, a dialog box did appear asking for a password confirmation. The evidence for this can be found in the video “Video 20.mp4”.
43	Yes	Clicking the CANCEL button simply closes the dialog box. The evidence for this can be found in the video “Video 20.mp4”. When I type the wrong password and click CONFIRM I received an error message:

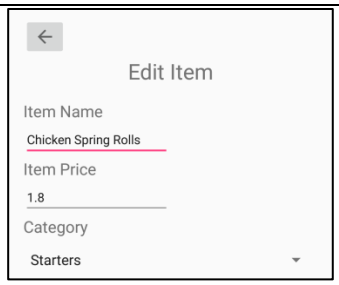
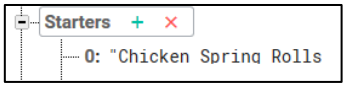
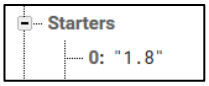
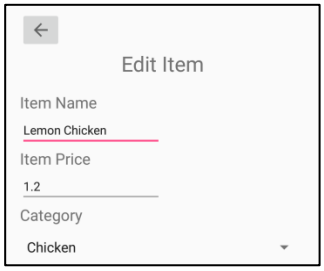
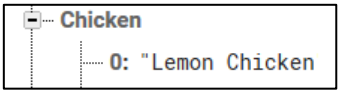
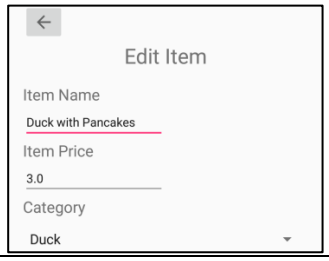
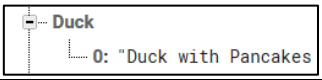
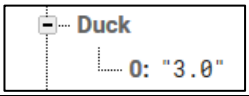
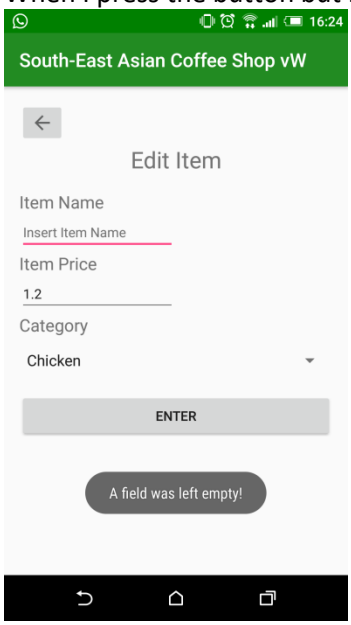
		 When I type the correct password and click CONFIRM I was taken to the specific page. The evidence for this can be found in the video “Video 21.mp4”.
44	Yes	I used two phones, one for the worker app and one for the customer app. As I make an order in the customer app, the number of pending orders updated straight away in the worker app, it increases by one. The evidence for this can be found in the video “Video 22.mp4”. I used both phones for the worker app. As I finish an order on one phone, the other phone gets updated straight away and the pending orders number goes down by one. The evidence for this can be found in the video “Video 23.mp4”.
45	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
46	Yes	The test for this is the same as Test 44, where I use two phones and check one for the number to update and the other to make an order or to finish an order. The evidence for this can be found in the video “Video 24.mp4” for the number decreasing by one. The evidence for this can be found in the video “Video 25.mp4” for increasing by one.
47	Yes	When I finish an order, the clickable text of that order disappears straight away. The evidence for this can be found in the video “Video 24.mp4”. When I make an order, the clickable text of that order appears straight away. The evidence for this can be found in the video “Video 25.mp4”.
48	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
49	Yes	From this, I can see that the data on the page is the same as the data in the

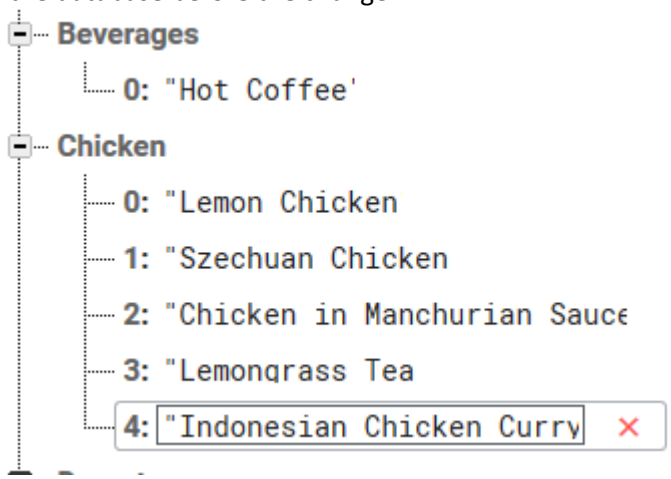
		database, hence it worked:  
50	Yes	When I click the user's detail button, a dialog box with the user's address details does appear. The evidence for this can be found in the video "Video 26.mp4". From cross checking the database I can see that the data downloaded is correct:  
51	Yes	When I click the GOTCHA button the dialog box closes. The evidence for this can be found in the video "Video 26.mp4".
52	Yes	When I click the notify button a dialog box appears with options to choose from. The evidence for this can be found in the video "Video 26.mp4".
53	Yes	When I click the CANCEL button the dialog box simply closes. The evidence for this can be found in the video "Video 26.mp4".
54	Yes	When I clicked on a notification option, a notification was sent to the customer's phone. This will work as long as they have the app running in the background and are signed in to their account. The evidence for this can be found in the video "Video 27.mp4" and "Video 28.mp4". When I click the custom notification option but didn't write any message, I get an error message:

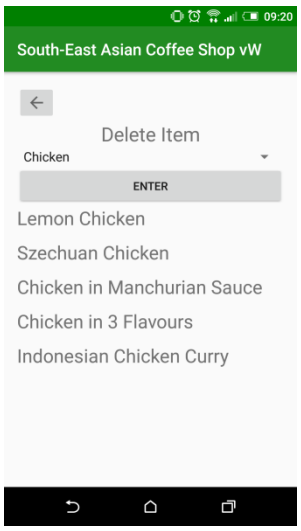
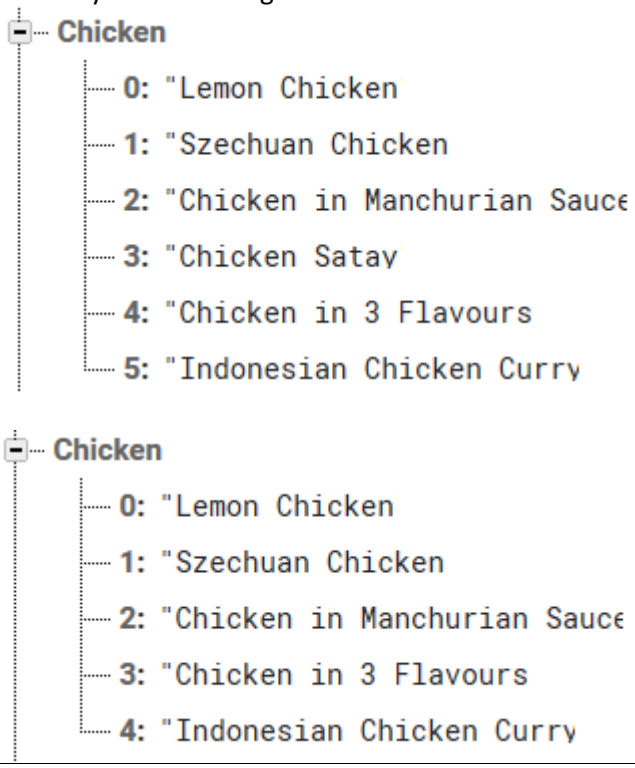
		
55	Yes	When I click the order completed button a dialog box appears asking for password confirmation. The evidence for this can be found in the video "Video 26.mp4".
56	Yes	When I click the CANCEL button the dialog box closes. The evidence for this can be found in the video "Video 26.mp4". When I click the CONFIRM button and type the wrong password, I get an error message:  When I click the CONFIRM button and type the correct password, the order was removed from the list and in the database under the pending orders branch. The evidence for this can be found in the video "Video 29.mp4". As you can see I chose the last john111 order, and when I complete the order there is only 2 john111 orders left, and the last one was something different.
57	Yes	Navigation buttons act exactly the same for every single page, the arrow

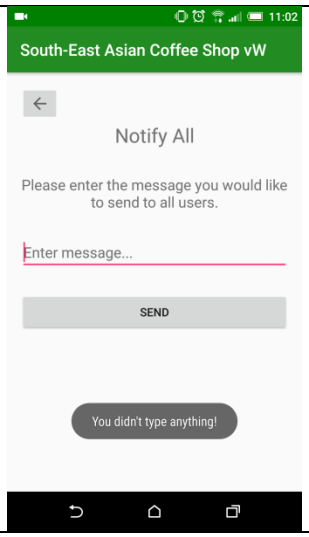
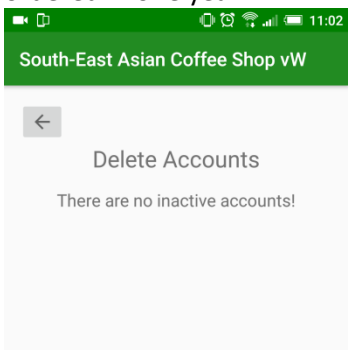
		button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
58	Yes	I have selected three categories to check, and from cross-referencing the database I can see that they are downloaded correctly:         
59	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
60	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
61	Yes	When I add a new item and press confirm, the page was refreshed and a popup appeared saying that the item has been added to the database. The evidence for this can be found in the video “Video 30.mp4”.

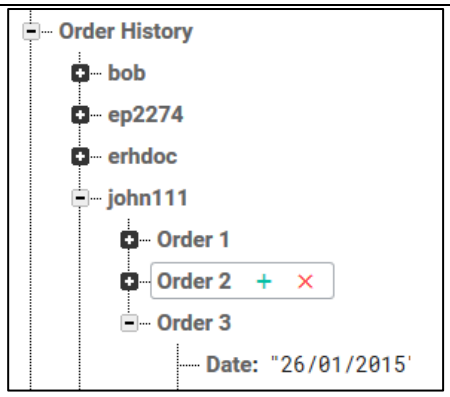
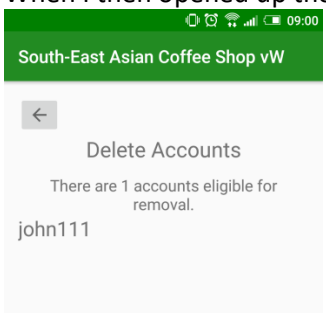
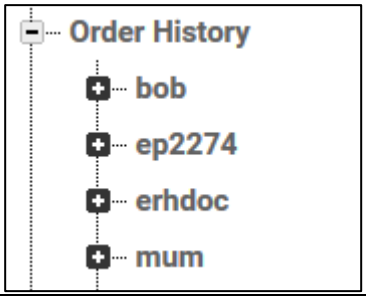
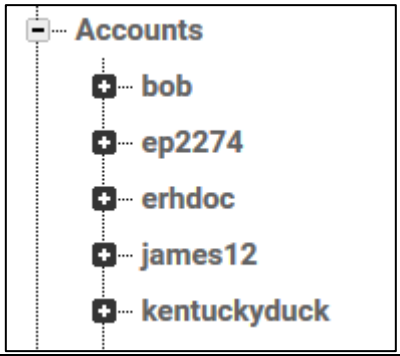
		  When I press the confirm button with 1 or more fields missing, I get an error message: 
62	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
63	Yes	This functions exactly the same way it does in Test 20 and Test 35. Since it functions the exact same way, it will then work as well.
64	Yes	This functions exactly the same way it does in Test 36, that the produced text is clickable and will take me to the next page. Since it functions the same way, it will then work as well.
65	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
66	Yes	I selected three items to check and as you can see the data that was set is correctly downloaded from the database: For Chicken Spring Rolls:

		   For Lemon Chicken:    For Duck with Pancakes:   
67	Yes	When I press the button but had a field left blank I get an error message:  When I filled in the fields and pressed ENTER I was taken to the previous page and a popup saying the update was done. The evidence for this can be

		found in the video “Video 31.mp4”. From the video you can see I mistakenly had Lemongrass Tea under the Chicken category and decided to edit it. This is the database before the change:  This is after the change, as you can see lemongrass tea was moved and the ID for chicken curry was shifted down: 
68	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
69	Yes	This functions exactly the same way it does in Test 20 and Test 35. Since it functions the exact same way, it will then work as well.
70	Yes	This functions exactly the same way it does in Test 22, that the produced text is clickable and will create a dialog box (for this case a confirmation). Since it functions the same way, it will then work as well.
71	Yes	When I click the CANCEL button it simply closes the dialog box. When I click the CONFIRM button the page becomes refreshed and the item was deleted and the items below that had their ID’s shifted by one. The evidence for this can be found in the video “Video 32.mp4”. Upon clicking the chicken category again, Chicken Satay is gone from the

		list:  And here you can see it gets deleted and the ID's shifting: 
72	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
73	Yes	When I didn't type anything and press the SEND button I got an error message:

		
74	Yes	Upon clicking the SEND button, any phone that has the customer app opened or running in the background, regardless of signing in, will get the notification. The evidence for this can be found in the video "Video 33.mp4".
75	Yes	Navigation buttons act exactly the same for every single page, the arrow button taking the user to the previous page, and the normal buttons taking the user to what is written on it. This has been tested in Test 6, Test 9 and Test 12; hence since it is the same it will definitely work for this Test as well.
76	Yes	Upon going to the page I get this when there are no accounts that have not ordered in one year:  I manually changed the date on john111's last order:

		 When I then opened up the page this appeared: 
77	Yes	When I click john111 a dialog box did appear asking for confirmation of the deletion. The evidence for this can be found in the video “Video 34.mp4”.
78	Yes	When I click CANCEL the dialog box simply closes. When I click CONFIRM the john111 account was successfully deleted and the page was refreshed. The evidence for this can be found in the video “Video 34.mp4”. As you can see john111 is not in the database anymore:  

Evaluation

Were the objectives met?

Functional Objectives

Here:

Objective	Met?	Comment
Customer App		
Make an Order		
1ai - Customers selects variety of menu items through searching by category	Yes	The users were able to choose a category item which would produce a list of the menu items under that category. The list is produced in order of how they will appear in a typical menu, starting from Starters and ending in Beverages.
1aii - Inputting the desired quantity of the item chosen	Yes	The users were able to enter their desired quantity when clicking the item that they chose. Errors have been minimized by only allowing the user to input a number greater than 0 for the quantity, and choosing auto-generated items for the chose item instead of typing it themselves.
1aiii - Calculate total price of each item (tax already included with each item) by multiplying the item price with the respective quantity	Yes	The program calculates this by multiplying the price with the quantity for each item.
1aiv - Calculate the total cost of the order by adding all the total prices	Yes	The program calculates this by adding the total prices of each item to get the total cost
1av – Require password to confirm selections	Yes	Upon confirming the order, a dialog box for password confirmation will appear. This prevents people from making 'troll' orders, like if someone uses their friend's phone to order a 50 OMR meal that they didn't want, they won't be able to since they won't (probably) know the password.
1avi – Customers should only be able to order from 9AM until 9PM	Yes	Users cannot access the 'Make an Order' page between 9PM and 9AM.

		The coffee shop isn't running between those times so it makes sense the users would not be able to order in that time.
1avii - Upload order to database, listing item id, quantity and total costs	Yes	Upon the user completing the order, the item name, price, quantity and total cost are uploaded into the database in the "Order History" field and the "Pending Order" field. Uploading the total costs prevents the same calculation occurring again within the app.
1aviii - Get the time of order and make sure it is unique, for an ID	Yes	The time of order is gotten to use as an ID. Should there be an existing ID for this time, it is simply increased by 1 until it becomes a unique ID. This will automatically generate the list of orders in chronological order of time ordered.
Discount Benefits		
1bi - If the customer has ordered at least 15 times, there will be a 15% Discount for every purchase onwards	Yes	The program checks how many orders the user has made from the database, if it is 15 or more it will then calculate a new total cost which is 85% of the original
Make user accounts		
1ci – Customers must input username and password	Yes	The user will not be able to make an account without filling in those details
1cii - Password must be at least 6 characters long (No need variety, e.g. Capital letters, numbers, symbols are not needed)	Yes	Should the user input a value with 5 characters or less an error message will pop up. The user can make the password with any combinations of characters as long as they are 6 characters at least. This creates a somewhat slightly more secure password.
1ciii - Customers must input region, region can only be in Ghubrah or Azaiba	Yes	The user will not be able to make an account without filling in those details. Having only those two options for regions shows that people from other regions cannot make an account
1civ - Customers must input Street address, House Address	Yes	The user will not be able to make an account without filling in those details

1cv - Give customers examples of what to type for street and house address	Yes	An information button is present for the user to press that will provide them with examples. Three examples are given since there are varieties of ways someone's house/street address can be written.
1cvi - Tell customers why only the two regions is selectable	Yes	An information button is present for the users to press that will provide them with why there are only two options
1cvii - Store data in database	Yes	If all values are correct, the account will be stored in the database under the "Accounts" field. This data will be used by the worker so that they know where to deliver the food.
1cviii - If user didn't input one or more fields, give a message and don't upload to database	Yes	An error message will pop up should this happen, and nothing will be uploaded to the database. This prevents lack of information from the users, if the user didn't type their street address then the worker will not know where to go, since house addresses can be the same in different streets and regions.
1cix - The user's ID will be the user's username in lowercase letters so no other person can have the same username	Yes	The program will make all characters lowercased for the user's username which will be the ID of the user's account details. This prevents other users from having the exact same username, despite upper/lower case, because it will be confusing if you have 5 users ordering at the same time with the usernames: WitchCat, witchcat, WITchCAT, witchCAT and witchcaT. Also it makes logging in easier as you won't need to capitalize any letters you may have capitalized when making it.
1cx – Do not allow if the username is already in use by someone else	Yes	The program checks the database to see if the username already has records under it or not.
Edit user accounts		
1di - Customers can change name, password, region, house address and	Yes	The user has the ability to change any of these details. The user cannot change

street address		their username as it is used as their unique identifier, and if you can frequently change it then it's not really unique. Also if the user had made more than 50 orders all those orders would have to be downloaded and uploaded to the new username ID which could take a big toll on the phone.
1dii - Require customer password to confirm change (If the customer is changing the password then require the old password)	Yes	Upon updating their details, a password confirmation prompt will appear to confirm the change of their details. This is just to make sure they want to change it, and also in case someone is changing their friend's password as a joke they would not be able to since they don't (probably) know the password, hence another method to get rid of 'trolls'.
1diii - Update data to database	Yes	If all values are correct, the account details will be updated in the database
1div – Same rules from Make user accounts is applied here	Yes	The password has to be at least 6 characters long, and there must not be any blank fields. To keep it consistent with what the user had to do initially.
View order history		
1ei – Download data from database	Yes	Get the data needed from the database
1eii - Customers can select a month and year to view a list of orders	Yes	The users get a drop down list selection for the month and year to see. This prevents the users typing the month and year themselves and typing it wrong or differently, e.g. you can type January or 1 for January and my code might only accept '1', hence a drop down list being used to limit their options. The year that is selectable is automatically updated with each new year.
1eiii - List all orders in the selected month and year as buttons called "Order 1", "Order 2" etc	Yes	If the order exists in that time period then they will appear as clickable texts on screen, otherwise a popup will appear saying that the user didn't order anything in that time period.

1eiv - Click on an order (e.g. click on "Order 1"), and it will display all the menu items ordered, price, quantity, total price, total cost and the date ordered on a seperate page.	Yes	Upon clicking the text, it will take the users to a separate page listing the details. Having it on a separate page is neater to display.
Exit App		
1fi – Closes app completely	Yes	Pressing the button to exit the app, closes the app. When the app is launched it will take the user to the starting page, the log in page.
Receive Notifications		
1gi - Get notifications from workers, status of their order	Yes	The customer app can receive notifications about their orders or a general notification. This allows interaction from the worker without requiring a mobile phone number.
1gii - Should appear on phone even if the app is not directly opened and is running in the background	Yes	Changing tabs to a different app, the phone still receives notifications. This is good as the user can do other things on their phone while waiting to get a notification from the worker about their order.
Worker App		
Edit Menu Item		
1hi - Require the edit menu password to access this feature (So workers can't randomly delete, add or change menu items for no reason)	Yes	Upon clicking the button to access the Edit Menu page, a password confirmation dialog box will prompt in order to access the page. This prevents from new workers from changing the menu at their leisure, so only workers with the password will be able to access the feature.
1hii - Can change item name or price	Yes	Users can change the name and price and categories of items. This is in case the worker added an item to the wrong category, so they can simply change the categories instead of deleting it and adding a new one
1hiii - Can delete an item completely	Yes	Users can get rid of an item completely, which will shift the ID's of the items below it

1hiv - Can add new items, typing the item name, price and category	Yes	Users can add a new item, which will be added at the bottom of the category
View Menu		
1ii - Workers can select a category to view the items under that category	Yes	Users can select a category from a drop down list which will produce the items from that category below it
1iii - Download from database	Yes	The data of the item name and price is downloaded from the database
1iiii - Item name and price will be listed	Yes	Item name and price is listed upon selecting a category and is produced in a table. This is a nice way to check the menu on the app, to cross check with the physical menu in case anything is missing or something needs to be deleted.
View Order		
1ji - Download order from database	Yes	The data of the orders are downloaded from the database
1jii - Display under awaiting orders	Yes	All orders are displayed on screen
1jiii - List the orders in chronological order of time ordered	Yes	The orders displayed are in chronological order of time. This will tell the workers that the orders at the top of the list are the highest priority since they ordered first.
1jiv - Display items, quantity and total cost (no need each individual cost)	Yes	The user can see the item names, quantity and total cost upon clicking an order. The individual cost of each item is not needed as the worker simply needs to know how much the customer owes them.
1jv - Option to view the customer's details (name and addresses)	Yes	The user can press a button to see the customer's name and addresses, so that they know where to deliver the food to.
1jvi - Option to send notifications	Yes	The user can press a button to send notifications about the customer's order to the customer, this allows interaction with the customer without requiring the mobile phone number of the customer.
1jvii - Option to remove the order when it is completed, however a password input is required from the customer	Yes	The user can press a button which will prompt a password confirmation dialog box that the customer has to fill in to

who made the order		delete the order. This prevents the workers from simply getting rid of orders for no reason, hence they need to actually go to the user that made the order to get rid of the order.
Send Notifications		
1ki - Select options (e.g. “Food is being made”, or “On the way”) and send	Yes	Users have the option to select ready-made messages on screen when notifying the customer about their order. This prevents the workers from typing the same message over and over again for each individual customer since that is time consuming, hence ready-made messages are provided where they simply have to click an option and it is sent.
1kii - Can also type send custom notification (e.g. “Arrived at the front of your apartment” or “Arrived at the back of your apartment”)	Yes	Users also have the option to type a custom message to notify the customer about their order. This option is provided in case there are multiple areas for parking in the customer’s address and to tell the customer which exact one the worker is at, and also other similar situations of uncertainty.
1kiii - Can also send notifications to everyone (not just customer who ordered) for things like announcements (e.g. “NEW MENU ITEM! Fish Soup for 1 OMR!”)	Yes	Users can send notifications that will go to every single customer, by typing a custom message. This is to broadcast a message to advertise a new menu item for example to every single customer.
1kiv - Require the send notifications password to do so (to avoid workers from sending random notifications to users)	Yes	Upon pressing a button to access the Notify All page, a password prompt will appear in order to access the page. This prevents workers from sending random notifications to customers.
View Customer Details		
1li - Can check name, address of those that ordered	Yes	The user can check the names and addresses of those that made an order
Delete Customer Details		
1mi - Can delete the account completely	Yes	The user can delete accounts that appear on the page, if the account is more than 1 year inactive then it

		appears on the page. This option is available for workers to get rid of old accounts that are not used to save up space
1mii - Ask for confirmation prior to deletion	Yes	Upon clicking the account username for deletion a confirmation dialog box will appear just to confirm the deletion
1miii - Require Delete customer Password to do so (to avoid workers from deleting accounts for no reason)	Yes	Upon clicking a button to access the Delete Accounts page a password prompt will appear in order to access the page
Exit App		
1ni - Closes app completely	Yes	Upon pressing the button to exit the app the app indeed closes

HCI Objectives

Here:

Objective	Met?	Comment
User Friendly		
2ai – Easy to understand	Yes	The texts in the apps are pretty straight forward
2aii – Clear titles	Yes	The titles of the pages are clear, ensuring the users that they are on a certain page. It has the biggest font size to show what page it is as it will be the most eye catching thing on the page.
2aiii – Clear instructions	Yes	Info buttons are provided for helpful information, buttons labeled in a way users will know what to do
Buttons		
2bi – Buttons are clearly titled as per their functions	Yes	Each button is titled as what they do, buttons that take you to a different page are titled the name of that page, and buttons that will produce a list have names such as “Enter”
Option Limitations		
2ci – Radio options to limit user input to certain things	Yes	Radio Buttons have been used for selections of the Region, since only two can be selected it limits the choice

		greatly.
2cii – Drop down lists options to limit user input to certain things with a greater range of options	Yes	Spinners have been used for selections of Month, Year and Item Category. Users cannot type their own values and have to choose one from the list. Since there are a range of months and years to choose from, a spinner is most suitable.
Page Navigation		
2di – Arrow buttons for going back a page	Yes	All navigations for going back a page require the user to press the arrow button. An arrow pointing to the left represents going back hence users will definitely know what its functions are.
2dii – Clear titles for advancing pages	Yes	Each button that will take the user to a new page is clearly labeled with the title of the next page. This ensures clearness of where the button will take the user

Data Storage & Security Objectives

Here:

Objective	Met?	Comment
Cloud Storage		
3ai – Using Firebase for the cloud storage	Yes	Firebase has been used for the cloud storage. It stores the account details of the customers, the menu details and the order history of the customers. It has plenty of storage space which is great since the app only stores text
Database		
3bi – Firebase Database is a JSON Tree Database	Yes	N/A
3bii – Have 4 main branches for Menu Items, Accounts, Orders Made and Awaiting Orders	Yes	Accounts, Menu, Order History and Pending Orders are the 4 main branches in the database
Require an Account with Username and Password to log in		
3ci - Account details only known to the account creator	Yes	Apart from the name and addresses, which can be seen by the workers. Ensures that the account is safe and for

		personal use. Name and address needed by the workers so that they know who to address and where to deliver
3cii - Creating an account avoids having to type the users' address over and over again	Yes	N/A
Password Protection		
3di - Password will be hashed one-way for maximum security and the hashed value will be stored in the database	Yes	The user's passwords have been hashed one way with SHA-256 and the hashed value is the one stored in the database. SHA-256 was chosen because it is one of the strongest hashes as of now, hence it will have the best secureness
3dii - Inputted password (logging in, confirming order) will be hashed and compared to the one stored in the database for confirmation	Yes	Inputted passwords are hashed one way with SHA-256 and compared with the password stored in the database for logging in or confirmations. This prevents the download of the user's real password in case of someone downloading the data for malicious purposes.

Environmental Objectives

Here:

Objective	Met?	Comment
Customer Location		
4ai - Only in Azaiba and Ghubrah (too far for the coffee shop to travel for other regions)	Yes	When registering an account there will only be two options of Azaiba or Ghubrah for the region data. This will show that customers living in neither region will not be able to make an account as there are no other options
4aii – Others cannot order for home delivery	Yes	No other possible regions to select from
4aiii - During sign-up there will be a Region detail to verify if they can create an account or not	Yes	An info button is provided telling the users that the coffee shop can only deliver to Azaiba and Ghubrah

I have achieved 100% of my objectives.

Overall, I believe my solution has achieved a level that is usable and comfortable for the users. The solution is definitely a better system than what the original system was. Data is stored which is accesible to both parties of workers and customers, and can also be used in the future. People can order at the same time, which gets rid of the trouble of the coffee shop phone being busy.

However, there are obviously improvements that coul be made to my system, which I will discuss along with feedbacks I received from my peers.

Feedback

My peer Tomas gave me feedback on his use of the customerapp.
"I think it's well done. The setup actually give you the info you need when signing up and not just after you hit the registration button. I liked how the menu items stemmed off of the main dish. For instance, Fish and the Sweet and Sour Fish; that's some good design right there. Another great perk is the efficiency of the app."
Tomas has a smaller phone than mine, and it appears that the layout wasn't produced too well. For example, in the log in page, the password input field is blocked by the log in button.

My peer Mairi says "Overall it is pretty good. However, there should be limits when ordering the food. For example, I am able to order 9000 Lemon Chicken, and I am fairly sure that the coffee shop will not have enough time and resources to make 9000 lemon chickens. There is a time period limit for ordering so that was good. Also, if say I wanted to change the quantity for an item, I would have to restart the whole list. Selecting the same item twice will produce two of that item in the list, which doesn't look as neat."

My peer Erhovwo says "It was a good idea to require passwords for big features in the worker app, however, what if a worker that doesn't work in the coffee shop anymore still has access to the app. There should be a feature where you can change the password for accessing those features. I like how the numbers for the pending orders are updated automatically, it's a quick thing to look at to see how many orders is left, instead of counting the list manually."

My peer Yanal says "I think there are ways of making the system more smooth. Perhaps when choosing something from a spinner it should automatically produce the list, with no need of pressing a button. Another one would be that the selection in the spinner should stay as the selected one when going back to the page.

Analysis of Feeback

Overall, there are definitely a lot of improvements that I can make to my system and my peers have suggested a lot of ideas. From what Tomas has said, I can safely say that my app is easy to understand and very clear. It is also well designed with the fluidity of selecting the menu items. I

can see that the idea of having limits is very important, as said by Mairi, as it does make the app more realistic for both design and for the coffee shop. Security is very important, and not just for user accounts, as Erhovwo suggested, I should also make the access to features in the worker app more secure. Fluidity and smoothness is preferable when using apps, as said by Yanal.

Possible Improvements of System

I should make the layout so that it is suited to all sizes of phone.

In the analysis section I mentioned about features needing to be usable to both customer and worker. As for this solution, the order history feature is only useful for the customers. The system could be further improved so that this feature is also useful for workers. The workers could have access to each customer's order histories. The data from this could then be analysed and processed in order to get information, for example, which menu item is the most popular in the coffee shop, or what's the most frequent item that is bought by a certain customer, or whether there's a correlation between an item and its price for popularity. This could lead to knowing which menu items to market to certain users and also to come up with potential new dishes.

In the worker app, perhaps I can add a feature where if you tap the title of the home page 10 times a dialog box will appear that allows you to change the access passwords to the features in the app. Of course, this will only be known to the owner. This will add to the secureness that Erhovwo had suggested.

During the making an order section, I should make it so that if you choose the same item for the second time, it should merge with the first one and the quantity be added to the first one. I believe a limit of 10 for the quantity is also good. Furthermore, I should also make it so that users can only order something every one hour, otherwise they can spam orders and keep ordering 10 lemon chickens every minute!