

Cuisine Royal
Project for SEG2105[B]

Team:

Jad Bader, 300250963

Gurjot Grewal, 300263760

Tal Musienko, 300258476

Alexander Wilson, 300219585

Table of contents:

Introduction	4
UML	5
Deliverable breakdown	5
Lessons learned	11
Conclusion	11

Figures:

Figure 1	UML Class Diagram	5
Figure 2	Main menu	6
Figure 3	Cook logged in screen	6
Figure 4	Admin logged in screen	6
Figure 5	Logged in screen	6
Figure 6	Chef sign up screen	6
Figure 7	Sign up screen.	6
Figure 8	Cook profile screen	7
Figure 9	Meal preview	7
Figure 10	Meal search parameter	7
Figure 11	Meal search results	7
Figure 12	Menu meal list	7
Figure 13	Pending orders format cook	7
Figure 14	Pending purchases cook	8
Figure 15	Cook modify meal list	8
Figure 16	Cook modify menu	8
Figure 17	Previous orders format customer	8
Figure 18	Previous order item preview	8
Figure 19	Purchase in progress dialog	8
Figure 20	Purchase request dialog	9
Figure 21	Purchase list	9
Figure 22	Rate cook screen	9

Figure 23	Remove meal list item dialog	9
Figure 24	Remove menu item dialog	9
Figure 25	Cook profile list formatting	9
Figure 26	Customer previous order	10
Figure 27	Complaint List	10
Figure 28	Complaint pop up	10
Figure 29	Complaints review screen	10
Figure 30	Complaint about cook screen	10
Figure 31	Permanent suspension	10
Figure 32	Add meal list	11
Figure 33	add Meal List Item Dialog	11
Figure 34	Add menu Item	11

Tables:

Table 1 Deliverables table:	5
-----------------------------	---

As part of SEG2105, we developed an app titled Cuisine Royal. The goal of this app was to become familiar with the development process, work with an IDE such as android studios and manage a database such as firebase. To achieve these goals the team was tasked with developing an app that could connect clients with cooks for a meal delivery service. This was done in 4 deliverables or sprints which are detailed later in Table 1.

In addition to the basic functionality of the app, the system uses additional classes and utilities to make the operations easier. First, the information for cooks and clients is stored on firebase. The next feature is part of the complaints system, it is possible for a cook to be temporarily or permanently suspended based on the severity of the complaint. Last feature is a rating system for the cooks. It is possible for clients to rate their interactions with the cooks so that other clients know which cooks are the preferred cooks.

```

classDiagram
    class Account {
        email : String !
        username : String !
        password : String !
        name : String !
        address : Address !
        type : AccountType !
        + Account(String, String, String, String, String, Address, AccountType) : void !
        + Account() : void !
        + Add More ...
    }
    class AccountType {
        + Add More ...
    }
    class Administrator {
        + Add More ...
        + Administrator(String, String, String, String, Address) : void !
        + Admin() : void !
        + createSuspension(Cook, Complaint) : void !
        + approveCook(Cook) : void !
        + deleteAccount() : void !
        + reviewComplaint(Cook) : void !
        + banCook() : void !
        + Add More ...
    }
    class Client {
        preferences : ArrayList !
        cards : Card !
        favMeals : ArrayList !
        ratings : HashMap !
        + Client(String, String, String, String, Address, Card) : void !
        + Client() : void !
        + addReference(CuisineType) : void !
        + addPayment(Card) : void !
        + getPayment() : Card !
        + createComplaint(Complaint) : void !
        + addFavMeal(Meal) : void !
        + getRating(Meal, Integer) : void !
        + purchaseMeal(Meal, Cook) : void !
        + Add More ...
    }
    class Card {
        nameOnCard : String !
        cardNum : String !
        securityCode : String !
        expiryCode : String !
        + Add More ...
        + Card(String, String, String, String) : void !
        + Card() : void !
        + setCardName(String) : void !
        + Add More ...
    }
    class Cook {
        + Add More ...
        + Cook(String, String, String, String, Address) : void !
        + Cook() : void !
        + addCuisineType(CuisineType) : void !
        + getCuisineTypes() : ArrayList !
        + addComplaint(Complaint) : void !
        + getComplaints() : ArrayList !
        + addMeal(Meal) : void !
        + removeMeal(Meal) : void !
        + getMeal() : ArrayList !
        + endingRating(Meal, Integer) : void !
        + isSuspension() : boolean !
        + endSuspension() : boolean !
        + Add More ...
    }
    class CuisineType {
        + Add More ...
        + Add More ...
    }
    class Complaint {
        + Add More ...
        + Complaint() : void !
        + Complaint(String, Client, Cook) : void !
        + getDate() : Date !
        + getSummary() : String !
        + getSummary(String) : void !
        + getOwner() : Client !
        + getReviewed() : Cook !
        + setCook(Cook) : void !
        + Add More ...
    }
    class Meal {
        types : String !
        name : String !
        price : double !
        ingredients : String !
        allergens : String !
        description : String !
        cookAsuggested() : String !
        id : String !
        + Meal(String, String, String, String, double, String, String, String) : void !
        + Meal() : void !
        + setPrice(double) : void !
        + Add More ...
    }
    class ClientSignUpScreen {
        registerBtn : Button !
        registerCardBtn : Button !
        clientFirstName : EditText !
        clientLastName : EditText !
        clientEmail : EditText !
        clientPassword : EditText !
        email : EditText !
        username : EditText !
        postal : EditText !
        street : EditText !
        city : EditText !
        country : EditText !
        cardNum : EditText !
        securityCode : EditText !
        exp : EditText !
        mAuth : FirebaseAuth !
        + Add More ...
        + onCreate(Bundle) : void !
        + Add More ...
        + onCreate(Bundle) : void !
        + homeNavigator() : void !
        + registerCookBtn() : void !
        + textClear() : void !
        + Add More ...
    }
    class ComplaintPopup {
        + Add More ...
        + onCreate(Bundle) : void !
        + Add More ...
    }
    class CookSignUpScreen {
        cookFirstName : EditText !
        cookLastName : EditText !
        cookUsername : EditText !
        cookEmail : EditText !
        cookPassword : EditText !
        postal : EditText !
        street : EditText !
        city : EditText !
        country : EditText !
        + Add More ...
        + onCreate(Bundle) : void !
        + homeNavigator() : void !
        + registerCookBtn() : void !
        + textClear() : void !
        + Add More ...
    }
    class ComplaintsReviewScreen {
        + Add More ...
        + onCreate(Bundle) : void !
        + Add More ...
        + onCreate(Bundle) : void !
        + Add More ...
    }
    class LoggedInScreen {
        + Add More ...
        + onCreate(Bundle) : void !
        + loginOut() : void !
        + clientRegister() : void !
        + login() : void !
        + Add More ...
    }
    class MainActivity {
        email : EditText !
        pass : EditText !
        cookRegisterBtn : Button !
        clientRegisterBtn : Button !
        loginBtn : Button !
        + Add More ...
        + onCreate(Bundle) : void !
        + launchAdmin() : void !
        + cookRegisterNavigator() : void !
        + clientRegisterNavigator() : void !
        + loginViaGate() : void !
        + loginUser() : void !
        + Add More ...
    }
    class AdminLoggedInScreen {
        reviewComplaintsBtn : Button !
        + Add More ...
        + onCreate(Bundle) : void !
        + complaintNav() : void !
        + Add More ...
    }
    class PermanentSuspensionScreen {
        + Add More ...
        + onCreate(Bundle) : void !
        + Add More ...
    }
    class MenuMealList {
        meals : List !
        + Add More ...
        + MenuMealList(Activity, List) : void !
        + getMeal(Int, View, ViewGroup) : View !
        + Add More ...
    }
    Account "1" -- "*" AccountType
    Account "1" -- "*" Administrator
    Account "1" -- "*" Client
    Account "1" -- "*" Cook
    Account "1" -- "*" CuisineType
    Account "1" -- "*" Complaint
    Account "1" -- "*" Meal
    Account "1" -- "*" ClientSignUpScreen
    Account "1" -- "*" ComplaintPopup
    Account "1" -- "*" CookSignUpScreen
    Account "1" -- "*" ComplaintsReviewScreen
    Account "1" -- "*" LoggedInScreen
    Account "1" -- "*" MainActivity
    Account "1" -- "*" AdminLoggedInScreen
    Account "1" -- "*" PermanentSuspensionScreen
    Account "1" -- "*" MenuMealList
    
```

Figure 1: project UML Diagram

The above UML was built using umple and the source code for the project can be found on git hub under: https://github.com/Jastes33/SEG2105_ProjectGroup4. The upper part of the uml shows all the basic classes and how they interact with each other. The lower part of the UML is the classes that contain the functionality for the screens and dialog boxes.

Throughout the project each member was required to complete specific tasks in order to achieve the goals of each deliverable. This has been broken down for ease of reading and is detailed below.

Table 1. Deliverables table:

	Deliverable 1	Deliverable 2	Deliverable 3	Deliverable 4
Jad Bader	-Fire base initial set up -admin login initial setups	-linking the UI with the classes	- Linking the new features of the UI with the classes	- Linking the new features of the UI with the background
Gurjot Grewal	-Implementations of user accounts	-linked class to the UI for the complaints - finished firebase set up	-Implemented the meal system for the cook class	-Created a search for a meal -set up purchasing system.
Tal Musienko	-UI design -implementation of some functionallity	- further developed the UI -set up test cases	- further developed the UI -setup test cases	- further developed the UI -setup test cases
Alexander Wilson	- Created a git hub repository for the app - Created the UML class diagram for the domain model	- updated the UML class diagram - created a complaint class.	- updated the UML class diagram - updated the meal class adding new featured needed to work with the meal class	- updated the UML class diagram. - drafted the final report

With the app there are 33 different screens. Some of the screens are viewable by all users, such as the main log in, and some screens are only visible to a specific account type. Screen shots for the app are detailed below.




<p>Welcome To Cuisine Royale™</p>  <p>Enter Email</p> <p>Enter Password</p> <p>LOGIN</p> <p>REGISTER AS A CLIENT</p> <p>REGISTER AS A COOK</p>	<p>Welcome placeholder! What would you like to do today?</p>  <p>CHECK PENDING PURCHASES</p> <p>MODIFY MEAL LIST</p> <p>CHECK PROFILE</p> <p>MODIFY MENU</p> <p>LOG OUT</p>	<p>Welcome to the admin page! What would you like to do today?</p>  <p>REVIEW COMPLAINTS</p> <p>LOG OUT</p>
---	--	--

Figure 2

Main menu

Figure 3

Cook logged in screen

Figure 4

Admin logged in screen


<p>Welcome placeholder! What would you like to do today?</p> <p>More features coming soon!</p>  <p>BROWSE MEALS</p> <p>VIEW ORDERS</p> <p>LOG OUT</p>	<p>Please enter all the required information to register</p> <p>Enter First Name</p> <p>Enter Last Name</p> <p>Enter Username</p> <p>Enter Password</p> <p>Enter Email</p> <p>Enter Street Address</p> <p>Enter City Enter Country Enter ZIP</p> <p>Enter a Short Description of Yourself</p> <p>UPLOAD VOID CHEQUE</p> <p>COMPLETE REGISTRATION</p> <p>HOME</p>	<p>Please enter all the required information to register</p> <p>Enter First Name</p> <p>Enter Last Name</p> <p>Enter Username</p> <p>Enter Password</p> <p>Enter Email</p> <p>Enter Street Address</p> <p>Enter City Enter Country Enter ZIP</p> <p>Enter Card Number MM/YY CVV</p> <p>COMPLETE REGISTRATION</p> <p>HOME</p>
--	--	--

Figure 5

Logged in screen

Figure 6

Chef sign up screen

Figure 7

Sign up screen.

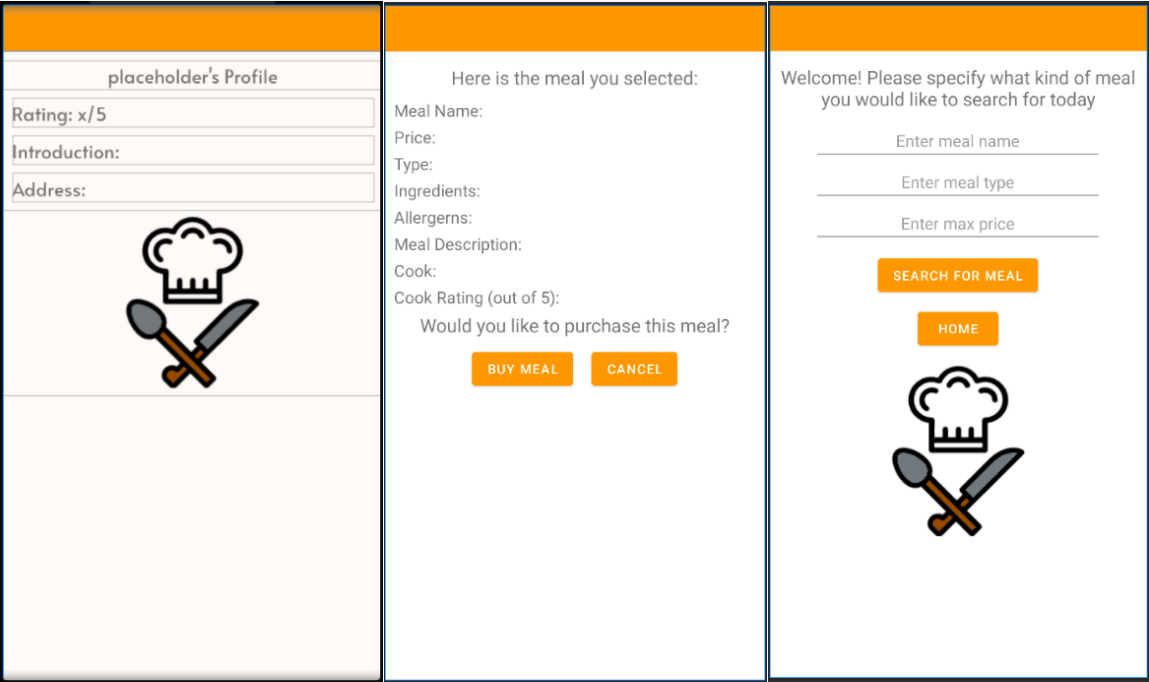


Figure 8

Cook profile screen

Figure 9

Meal preview

Figure 10

Meal search parameter

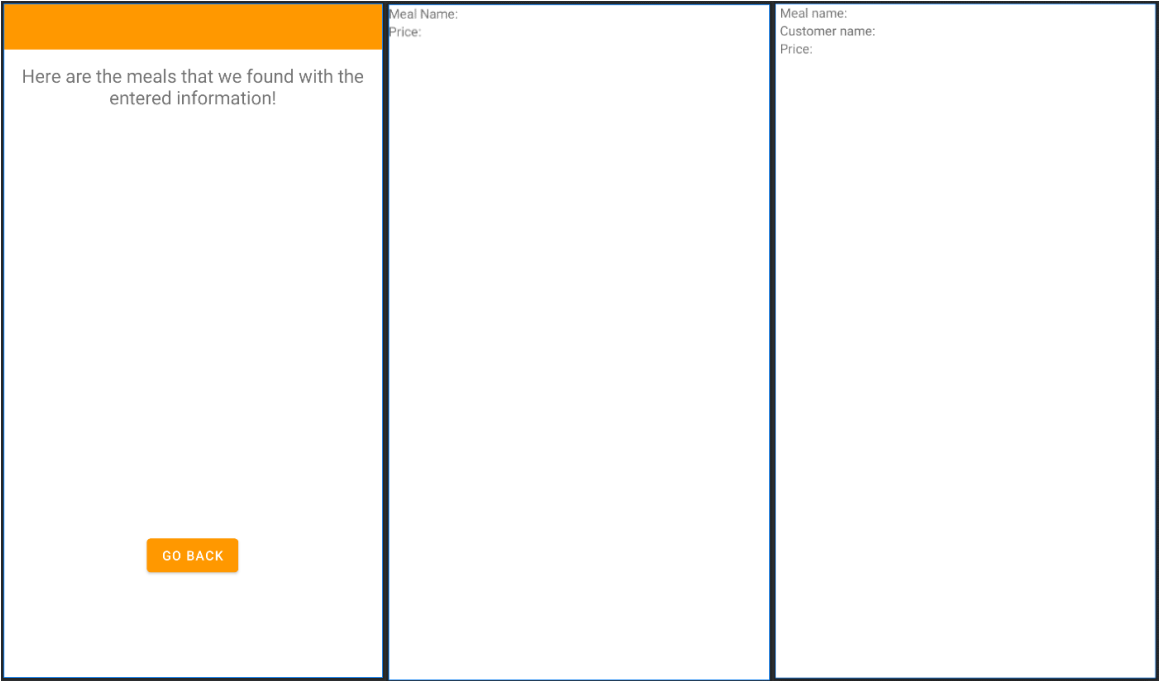


Figure 11

Meal search results

Figure 12

Menu meal list

Figure 13

Pending orders format cook

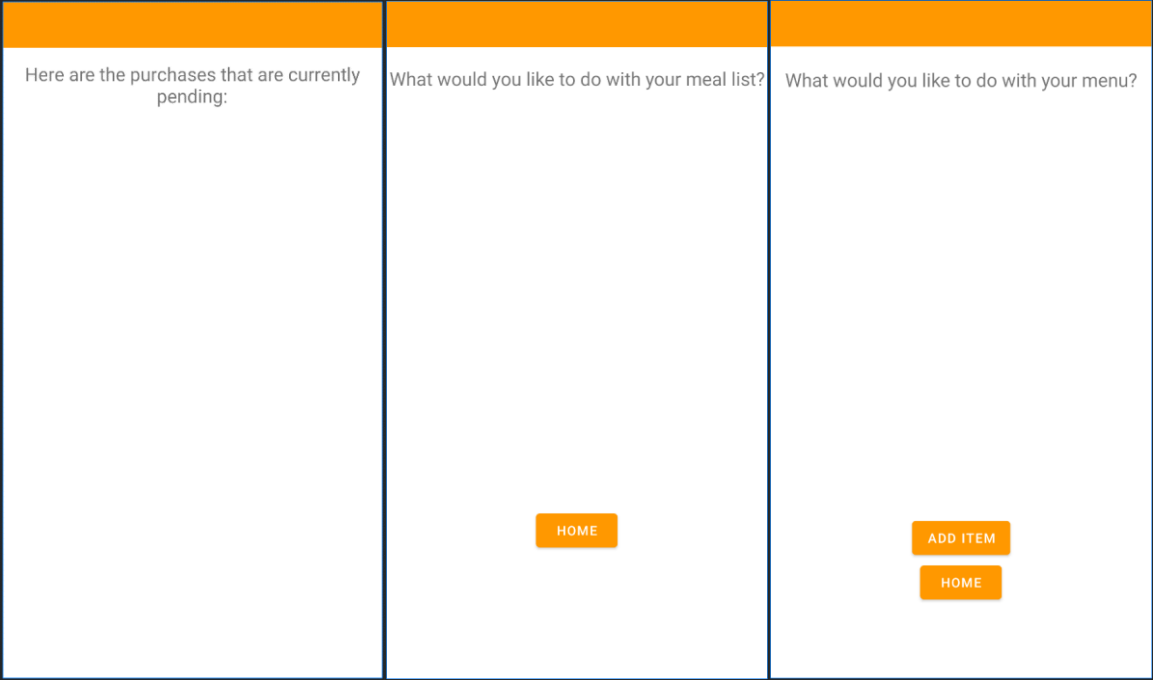


Figure 14

Figure 15

Figure 16

Pending purchases cook

Cook modify meal list

Cook modify menu

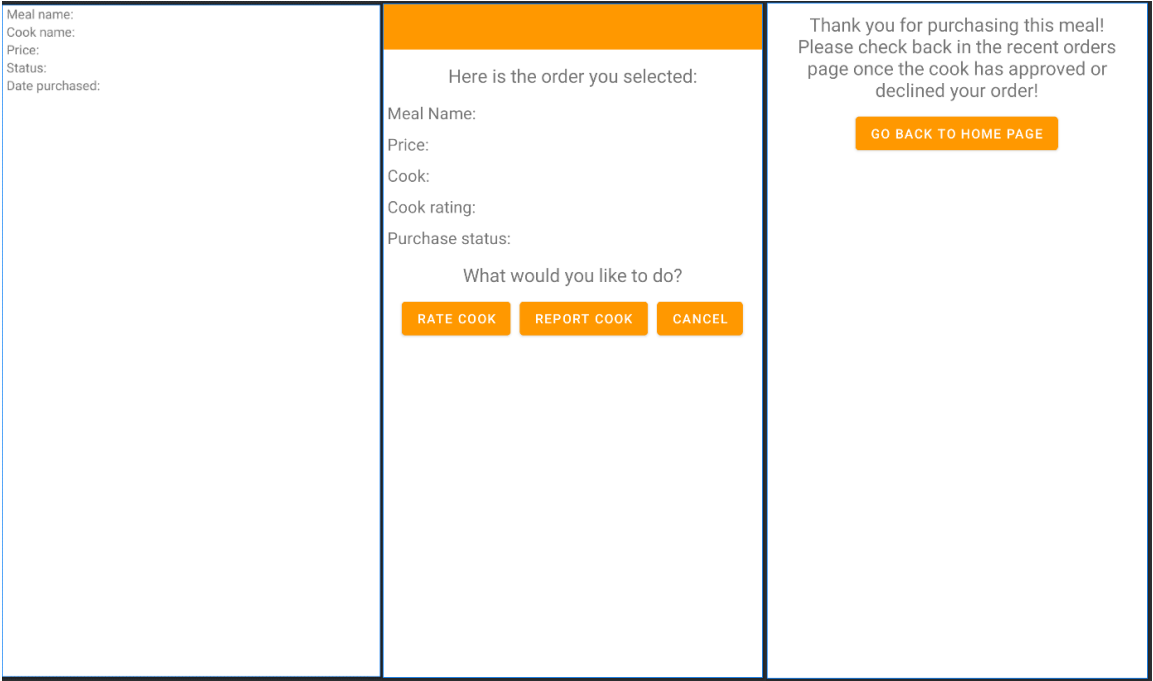


Figure 17

Figure 18

Figure 19

Previous orders format customer

Previous order item preview

Purchase in progress dialog

<p>Here are the details of the purchase request:</p> <p>Client:</p> <p>Meal Name:</p> <p>Price:</p> <p>Order ID:</p> <p>Would you like to accept this purchase?</p> <p><input type="button" value="ACCEPT"/> <input type="button" value="REFUSE"/></p>	<p>Meal Name:</p> <p>Order ID:</p> <p>Status:</p>	<div></div> <p>What would you like to rate the cook out of 5?</p> <p>Enter a number from 0-5</p> <p><input type="button" value="SUBMIT RATING"/> <input type="button" value="CANCEL"/></p>
--	---	--

Figure 20

Purchase request dialog

Figure 21

Purchase list

Figure 22

Rate cook screen

<p>Would you like to remove this item?</p> <p><input type="button" value="YES"/> <input type="button" value="NO"/></p>	<p>What would you like to do with this meal?</p> <p><input type="button" value="ADD MEAL TO OFFERED MEAL LIST"/></p> <p><input type="button" value="REMOVE MEAL"/></p> <p><input type="button" value="CANCEL"/></p>	<p>Cook name:</p> <p>Rating: x/5</p> <p>Description:</p>
--	---	--

Figure 23

Remove meal list item dialog

Figure 24

Remove menu item dialog

Figure 25

Cook profile list formatting

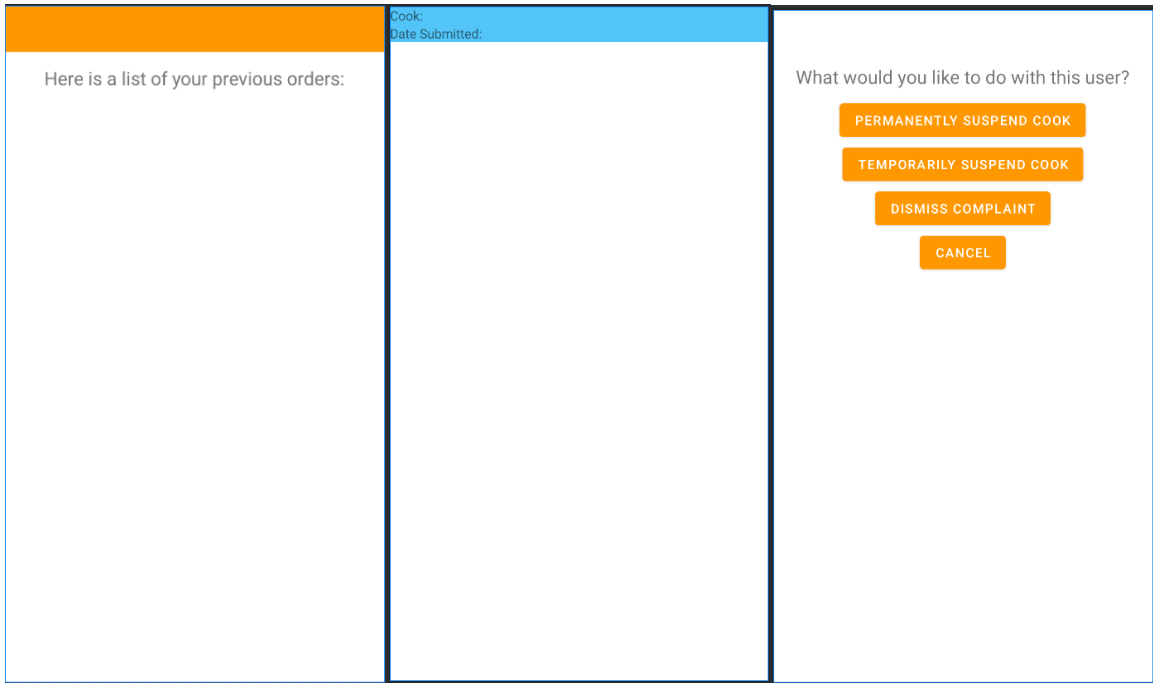


Figure 26

Figure 27

Figure 28

Customer previous order

Complaint List

Complaint pop up

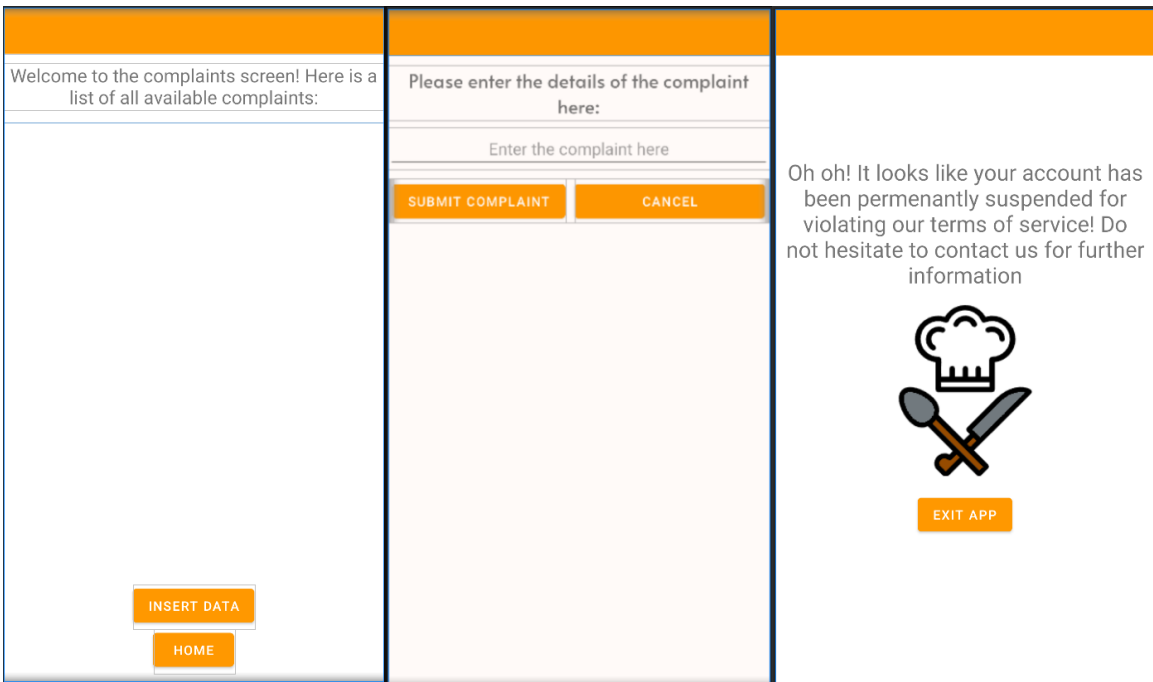


Figure 29

Figure 30

Figure 31

Complaints review screen

Complaint about cook screen

Permanent suspension

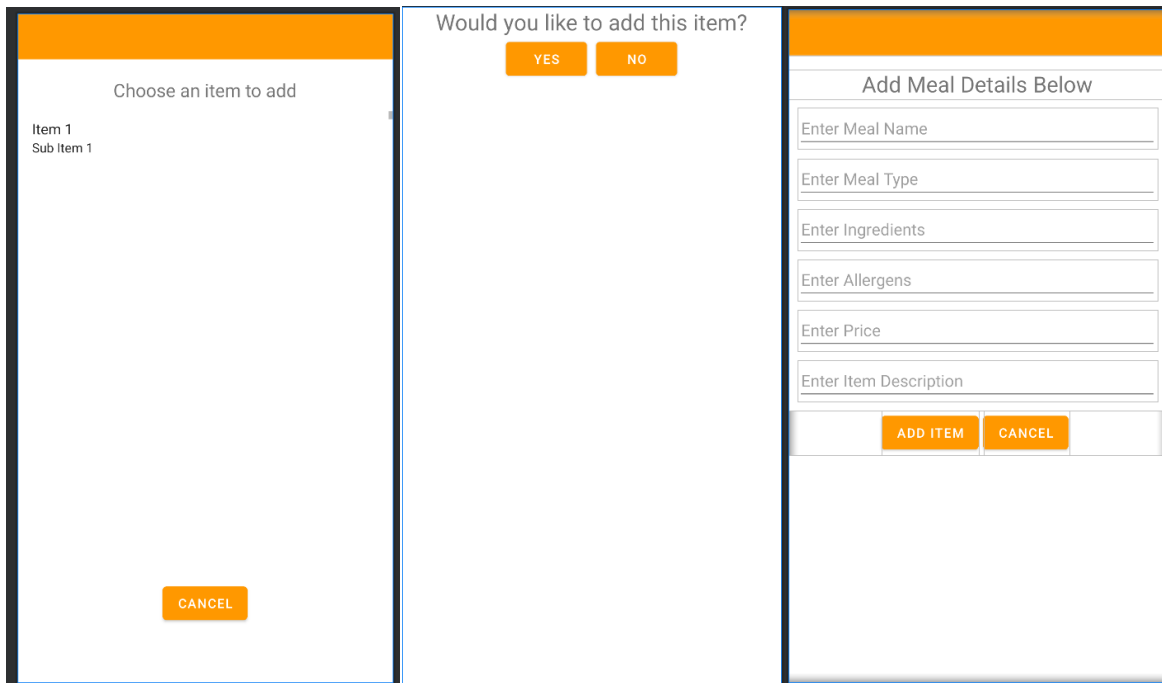


Figure 32

Add meal list

Figure 33

add Meal List Item Dialog

Figure 34

Add menu Item

Lessons learned:

Through out this project several problems we encountered. During the first deliverable we had an issue with the app scaling depending on what emulator model was being used. This was solved when the page layouts were being reviewed. One of the other issues that we had was when setting up the complaint system. We looked at different options on how to implement it to make it easily accessible across the different account types as well as making it easy to store on firebase. A positive lesson from the project what the use of git hub and the use of discord. Both of these systems allowed for great communication throughout the team.

Conclusion:

During this project many concepts learned in the labs, tutorials and lectures were applied. The labs demonstrated how to use a database effectively and how to use the basic functionality of android studios. As part of the tutorial, GitHub was introduced as an effective concept for working remotely as a team. Lastly in the lectures concepts such as design principles and workflow methods helped to keep our group organized and on track. Overall, this project is a great way to get involved and better understand concepts that our team will use later in life.