Android Project: Mealer App

SEG 2105 – Introduction to Software Engineering

Fall 2022

University of Ottawa

Course Coordinator: Dr. Hussein Al Osman

Teaching Assistant: Dharmin Sodwadia

Group 22

Jessica Chen - 300238463

Tyler Blinson - 300169866

Tom Latimer - 300250278

Owen Halvorson -300251644

Annika Whitwam - 300232764

Submission Date: December 9, 2022

Table of Contents

List of Figures

List of Tables

Introduction

Mealer is an app that allows people to order homemade meals or cook them. Users can choose from a variety of different cuisines and meal types. Cooks can earn income on the app by cooking requested meals from their offered list. The Mealer team also moderates the app to ensure that complaints about chefs are continually checked and dealt with. This allows for an excellent user experience as only trusted chefs operate on the app. The app is developed for android and utilizes Firebase to store the backend in the cloud.

UML Diagram

Figure 1: UML Diagram

Contributions of Each Team Member

Team Member	Deliverable 1	Deliverable 2	Deliverable 3	Deliverable 4
Jessica Chen	- UI and implementation for welcome page (that leads to sign in/sign up)	- Created and debugged temporary suspension method - Implement verification for suspension page	- Implement Unit Tests - Update app's UI and interfaces	- Implement Complaints Class in Firebase - Improve Client Home page and set up list of Purchase Requests - Implement client submit complaint and rate cooks functions
Tyler Blinson	-Ui for the sign in page -Created User, Client, Cook, Admin and Meal class along with basic functions (getters and setters)	-Implemented the sign in functions -Made the database retrievals work	-Fixed a lot of bugs -Made the cook class directly work with the database	-Implemented search function - Screenshots to showcase application
Tom Latimer	-UI and implementation of client sign up page -set up account registration; accounts are stored in Firebase	-Implement the admins ability to permanently suspend a cook	-Implement Unit Tests -Implemented code to offer and un-offer a Meal on the cooks menu -Implemented code to display cooks' meals in listview	-Implemented search results recyclerview -Implemented client's ability to view meal information -Implemented code to send a purchase request for a meal
Owen Halvorson	- Home page UI and Functionality - Log off	-Worked on permanent suspension method -Implemented code to check if suspended if logged on -Uml diagram changes	-Implement Unit tests - Updated UI -UML Diagram changes - Helped implement various features associated with the menu	- User profile and ratings UI -Implemented code for user profile page

| Annika Whitwam | - UI and implementation of cook sign up page
- UML diagram | - Set up list view for administrator complaints
- Implement the admins ability to dismiss complaint | - UML diagram
- Update the app's UI
- Helped to implement code to add meals to the "Meals" section in Firebase | |

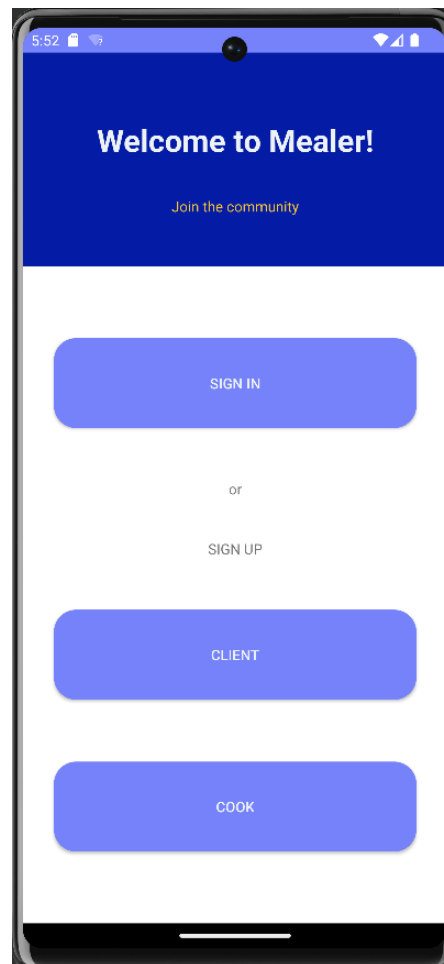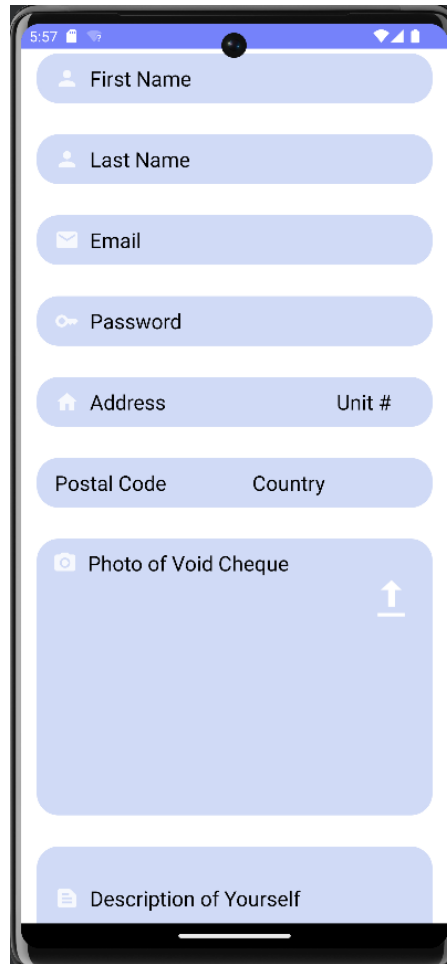Table 1: Contributions of Each Team Member for Each Deliverable

Screenshots of the App

Figure 2: Sign in/sign up screen
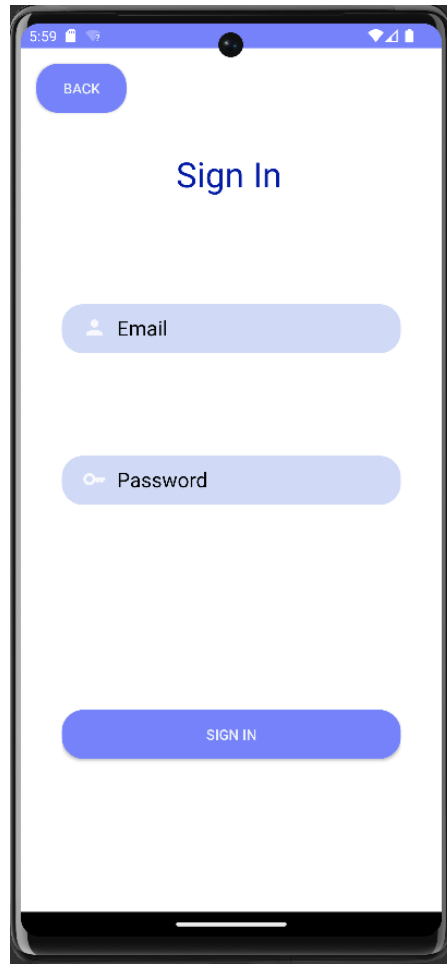
Figure 3: Cook sign up page
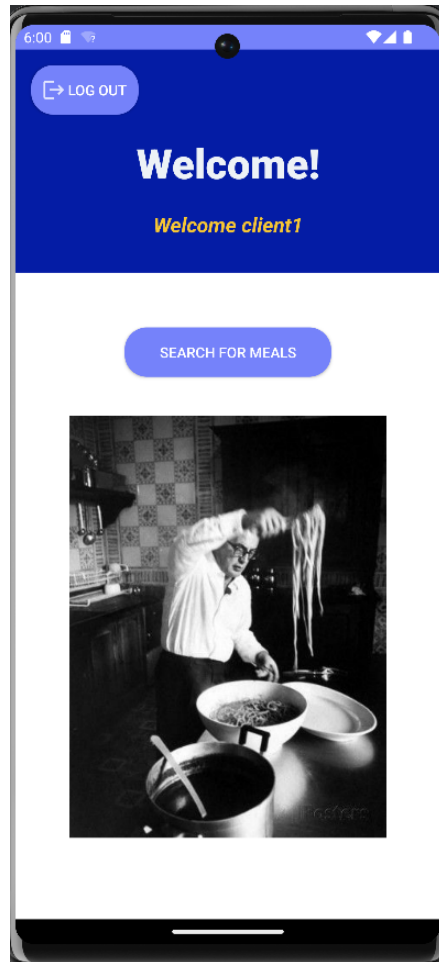
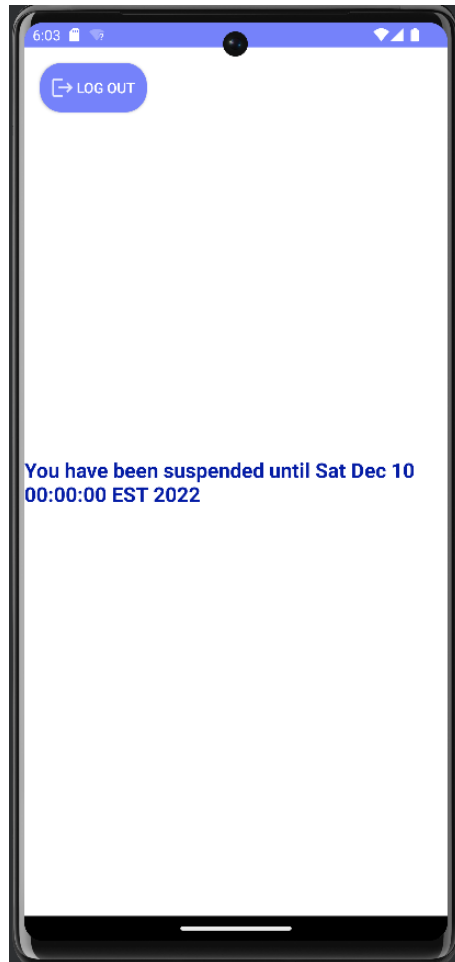Figure 4: Client sign up

Figure 5: Sign in page
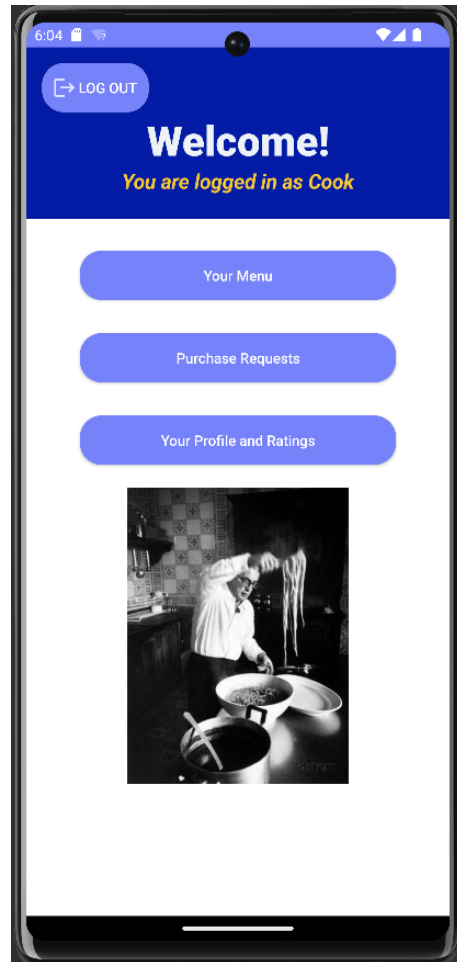
Figure 6: Client home page
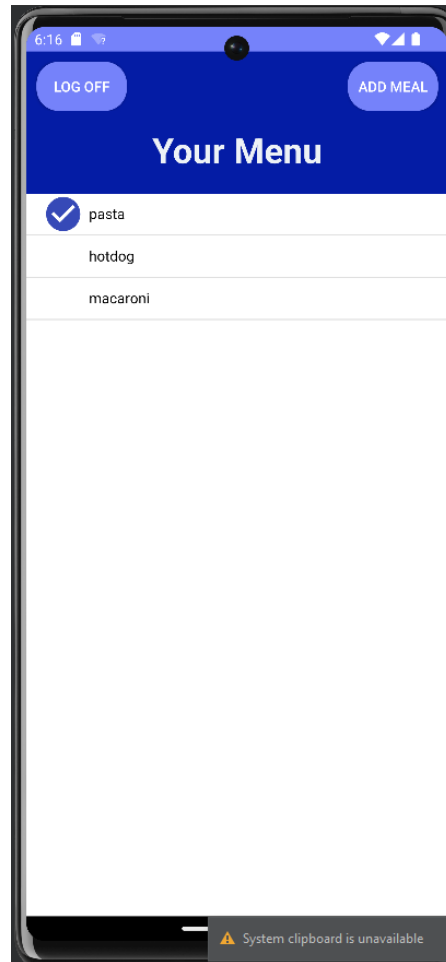
Figure 7: Suspended cook sign in

Figure 8: Cook home page

Figure 9: Cook menu page

Figure 10: Cook add meal page
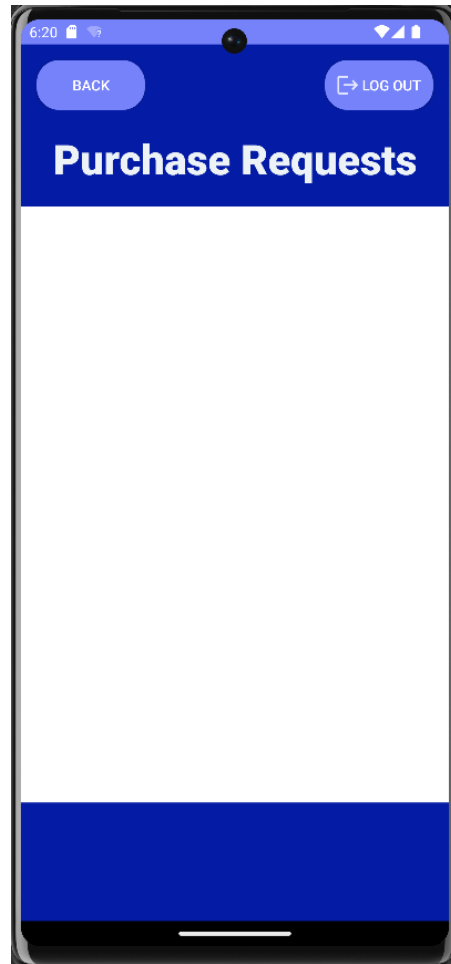
Figure 11: Cook purchase requests
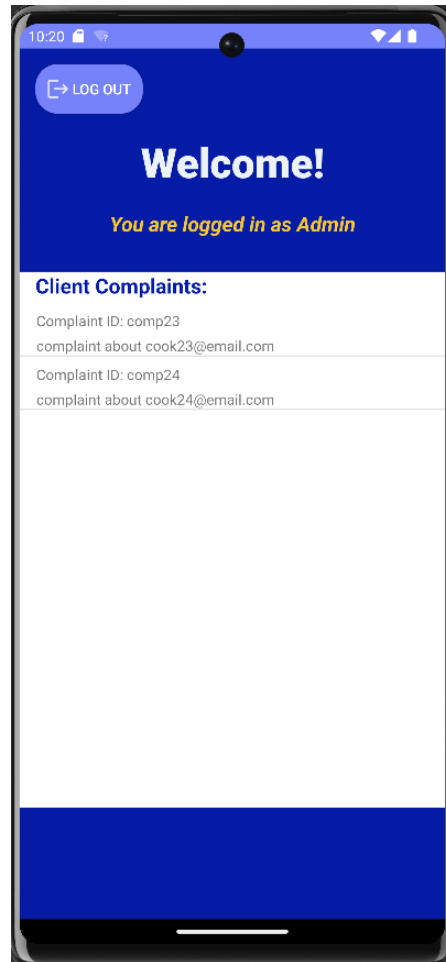
Figure 12: Admin home screen
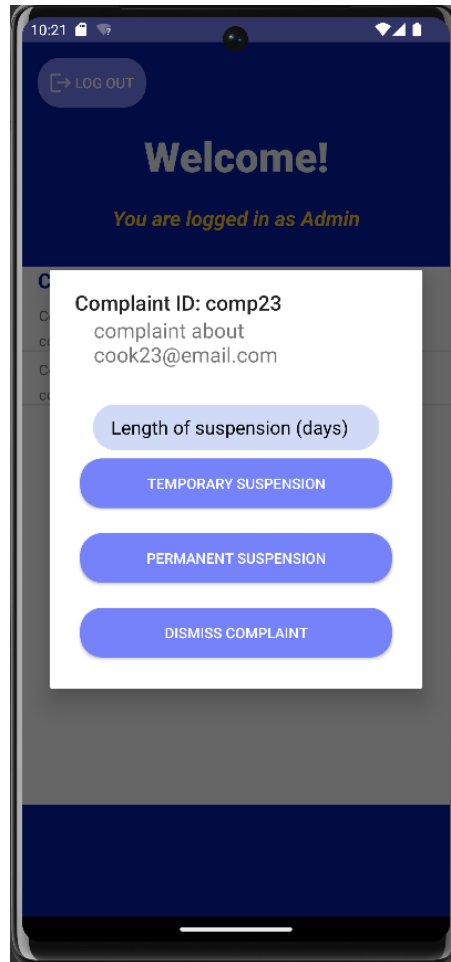
Figure 13: Admin suspension window
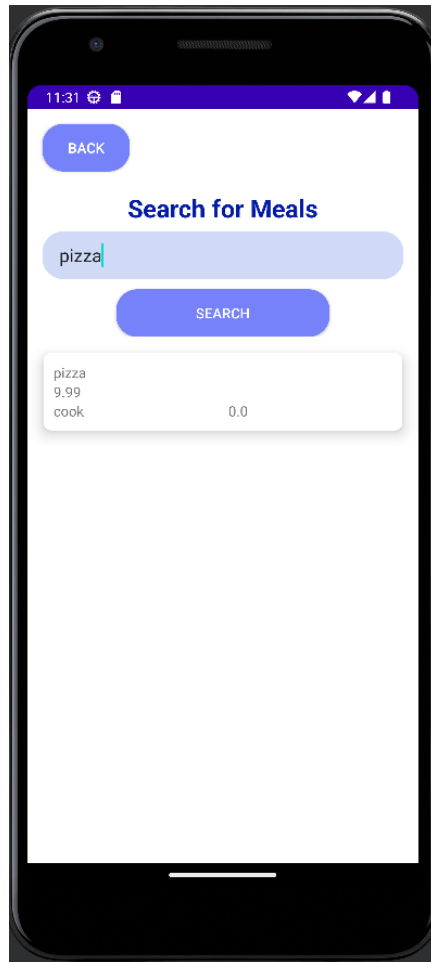
Figure 14: Search meal
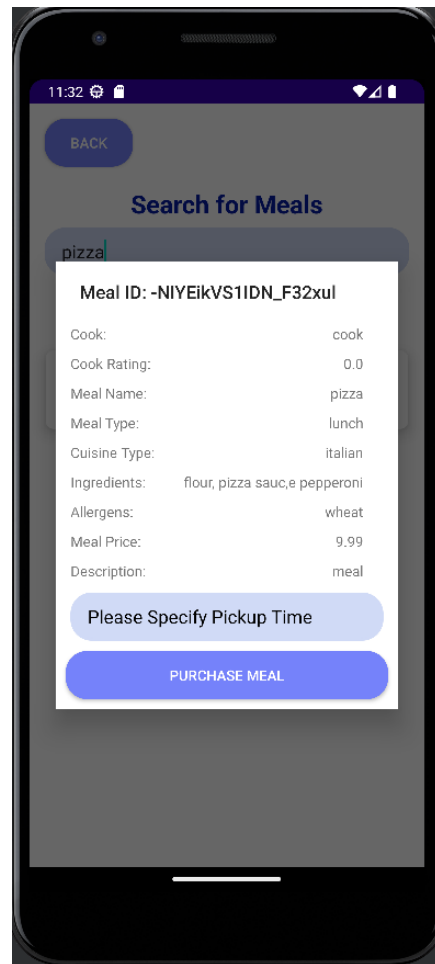
Figure 15: Meal purchase screen

Lessons Learned

Unlike other assignments involving code, the design of the Mealer application was more intensive and involved. The development of an android app requires the cohesive interaction between different subsystems, such as the activities, the database and supporting classes. Many of the lessons we learned involved the development of these subsystems in conjunction with designing a comprehensive system. Our first challenge was based on how we wanted to organize our app and divide it into different activities. We were unfamiliar with the development of android applications, so we quickly had to learn how activities were created and how we could use them to create our app's functionality. We learned how to mix UI elements with java code to create an interactive design that could react to user input and display results sourced from a database. Our second largest lesson learned would be learning how to set up the database and the ongoing maintenance required to update the database when we update our app. We originally struggled with uploading and retrieving data from the database, but we managed to implement it

successfully. One design decision we could have changed was to disentangle the way we accessed data from the database with how a user interacts with the app. Currently, we have some methods that the user interacts with directly send data to the database. Instead, we could have had the users input represented through objects modeling the system that would occasionally be sent/retrieved from the database, thus saving on resources. This would be a fundamental design decision that we think should be implemented were we to create another application. In addition to figuring out how to send and retrieve data from the database, we did not realize how greatly changing the objects of our system would affect the application in relation to the database. When we implemented additional responsibilities, such as creating Meals or Complaints, we had to modify the objects that modelled the user accounts to account for the additional functionality. This in turn caused conflict when we were trying to load old accounts to test the app because the data stored no longer matched the object representation which caused the app to crash. We learned that the continuous development of our system could introduce bugs into code we thought was already completed. Ever since, we were cautious and analyzed our design choices critically to map out every possibility of introducing unwanted bugs in the system. Finally, one of our significant hurdles was learning how to display information from the database to the app in the form of listviews and recyclerviews. Displaying information from the database was crucial to how our app worked. We learned to create separate UI elements to display objects from the database, as well as creating custom adapter classes to merge the UI with the data being retrieved. In conclusion, the development of the Mealer app taught us a lot about developing a system from subsystems, interacting with a database and quality control when implementing new features.