Final Project Report

Android Mealer App

2022 Fall SEG2105[B]
Course Coordinator: Wassim El Almar

Members:
Patrick Meyer - 300220498
Allen Mei - 300238743
Ashvin Ramanathan - 300242541
Oscar Li - 300248450

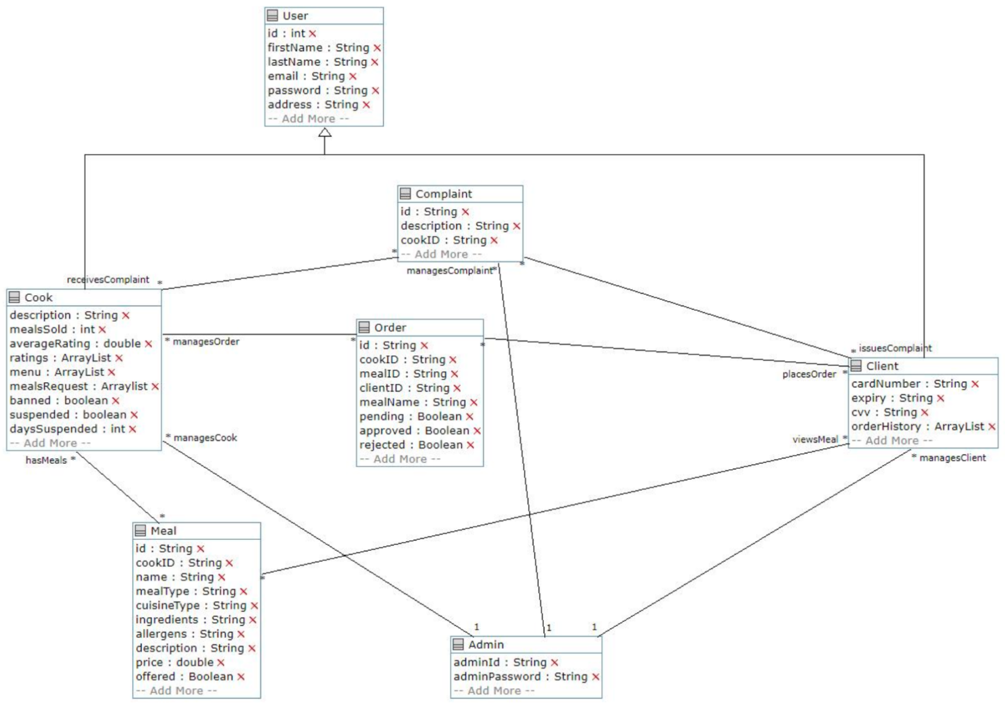
Table of Contents

Introduction

Mealer is a simple Android application that allows the purchase of meals by clients from cooks. Mealer has an admin who manages a list of complaints filed by clients and they can administer suspensions to cooks they deem problematic. The application was programmed entirely in Java and user information was stored into a Firebase Database.

This report aims to provide an overview of the functions of the Mealer app and contributions of group members. Included is an updated UML diagram, each group member's contribution, screenshots, and lessons learned.

Final UML Diagram

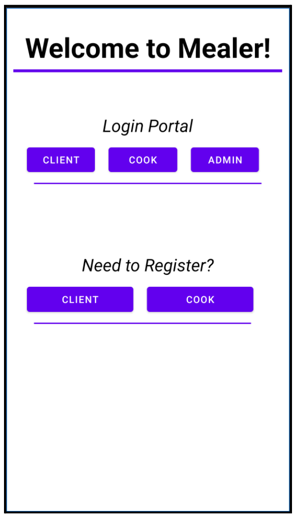


Contributions

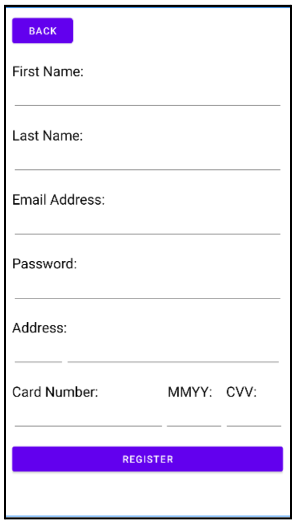
Member	Dev. 1	Dev. 2	Dev. 3	Dev. 4
Patrick	Firebase + Backend coding	Backend	Backend	Backend
Allen	Firebase + Backend coding	Backend coding	Backend coding	Backend coding
Ashvin	Separated Cook+Clients w/ LogOuts	JUnit (tried android testing)	UI + JUnit Testing	UI + JUnit Testing
Oscar	UML Diagram + Deliverable code outline (functions and classes needed)	UML Diagram + Deliverable code outline (functions and classes needed)	UML Diagram + Deliverable code outline (functions and classes needed)	Report + UML Diagram + Deliverable code outline (functions and classes needed)

Further elaborating on what everyone worked on. Patrick and Allen strictly worked on the backend side of the application. From setting up the Firebase and getting it to work, to implementing required functionalities. On the other hand Ashvin worked primarily on the unit testing and on the user interface of the application. Great applications require great UI's! Lastly, Oscar was in charge of designing the UML diagrams and making outlines of the functions required to fulfill the given requirements. Without his help we would not have been able to code as effectively.

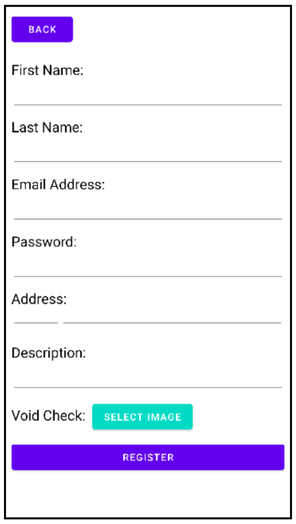
Screenshots



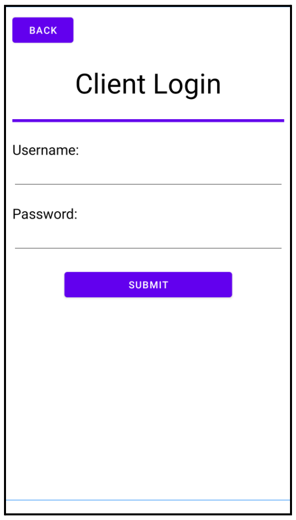
Main menu



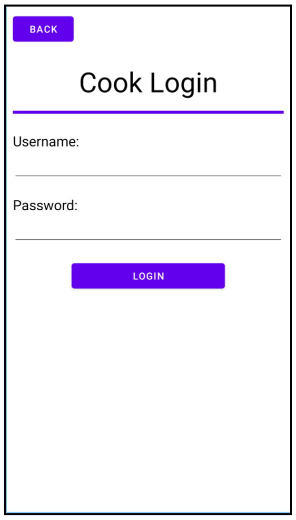
Client registration



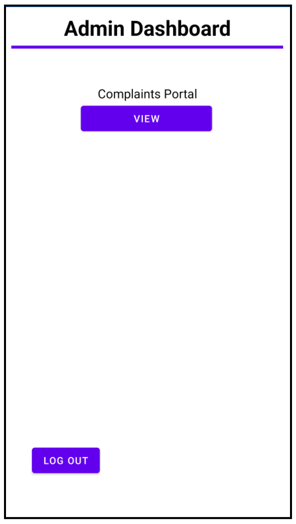
Cook registration



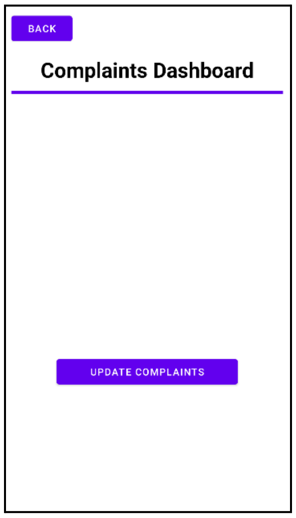
Client login



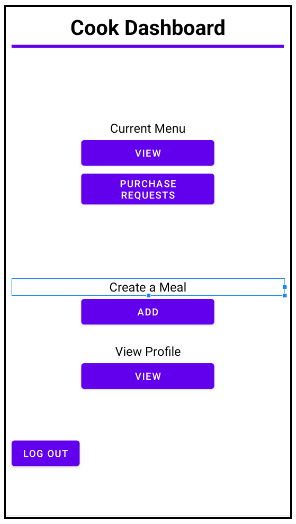
Cook login



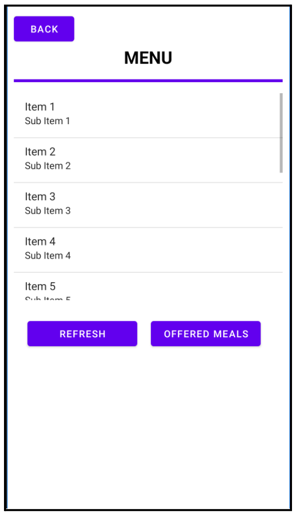
Admin dashboard (homepage)



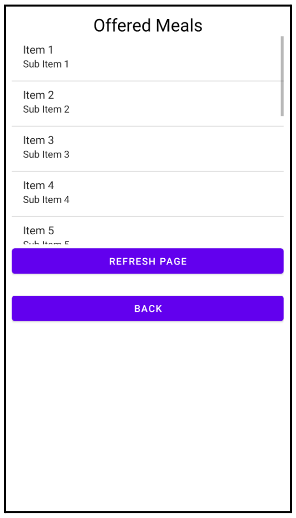
Complaints dashboard



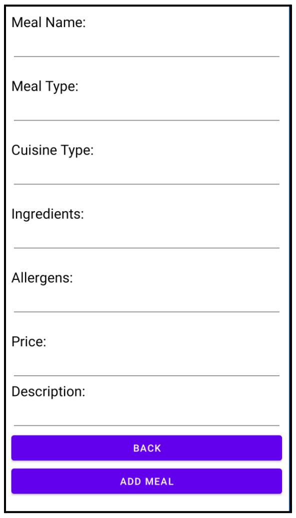
Cook dashboard (homepage)



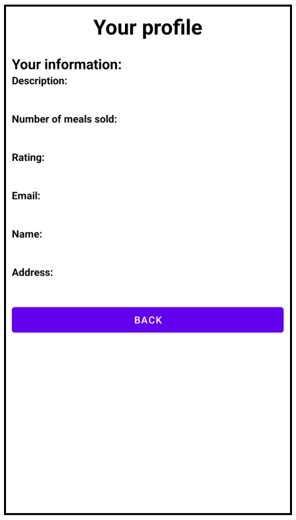
Cook menu page



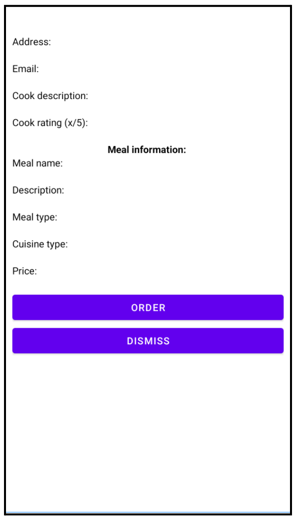
Cook offered meals page



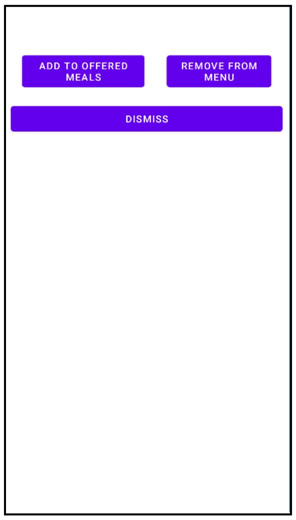
Cook create a meal



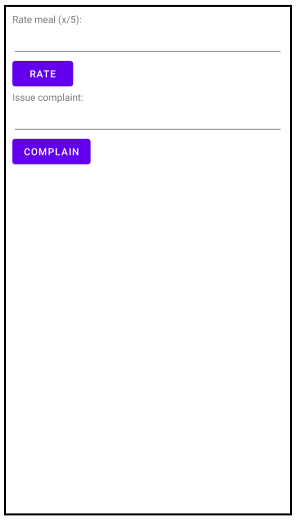
Cook profile



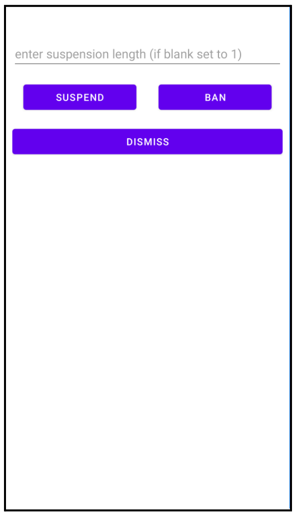
Meal information pop-up



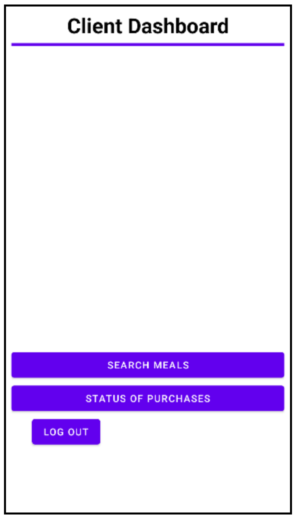
Add/fremove meal pop-up



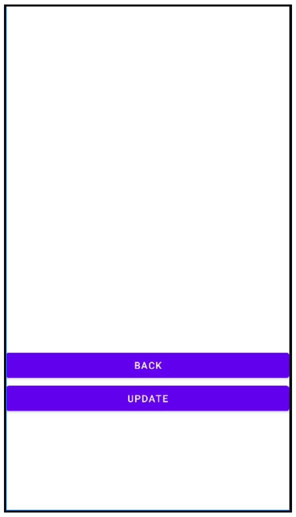
Rate/complain about Cook pop-up



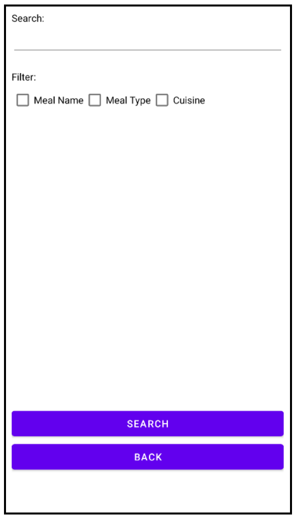
Admin complaint pop-up



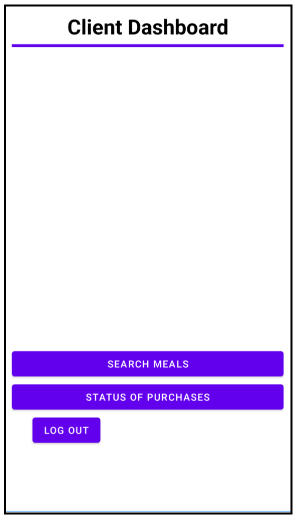
Client dashboard (homepage)



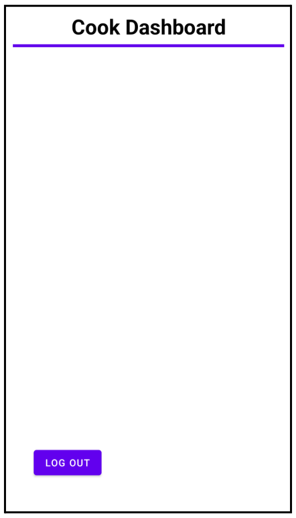
Cook/Client requests page



Client meal search



Client dashboard (homepage)



Banned Cook dashboard (homepage)

Lessons learned

Throughout the process of creating Mealer we encountered multiple problems. The following is a list of some issues that were fixed and the lessons learned from them:

- Most members on our team had never worked with a real-time database so hearing that we were gonna need to implement one for our project initially scared us. The initial setup of the Firebase database was, to be honest, pretty challenging and required a lot of searching on the web. Over the course of the term we got increasingly comfortable with Firebase and discovered its true potential. We now know the importance of databases and how they can effectively store data and make our lives easier.
- Code DRY. The implementation of every deliverable required us to code similar functionalities for different classes. Coding over and over the same functions/pieces of code would take longer and reduce the maintainability of our application. Reusing pieces of code helped us maintain and limit issues caused by rewriting them.
- Plan ahead. For the first deliverable we waited until the last possible day to submit our work. Thankfully, we were able to implement everything required and it functioned, but it would have been equally possible that it would not have functioned and we would have been stuck on a small issue that we could have fixed with an extra day. For the next three deliverables we made sure to work and finish the deliverables at least three days before they were due. This ensured a lengthy amount of time to fix problems.
- Use Github. Many members on our team hadn't used Github to manage files/projects. Since we were forced to use Github for our project, we learned very rapidly the impressive functionalities provided. With Github we can very easily receive and upload files that others can then pull. This made it extremely easy to collaborate on the project remotely.