Mealer - Final Report

SEG 2105 - Introduction to Software Engineering

To Professor Hussein Al Osman

Adam Wilkins 1352057
Justin Zhang 300178062
Nicolas Bérubé 300239551
Samuel Pierre-Louis 300211427
Shyan Arulaswaran 300232724

Due by: December 7th 2022

Table of Contents

Table of Contents **2**

Introduction **3**

UML Diagram **3**

Contributions **4**

- Deliverable 1 4
- Deliverable 2 4
- Deliverable 3 5
- Deliverable 4 5

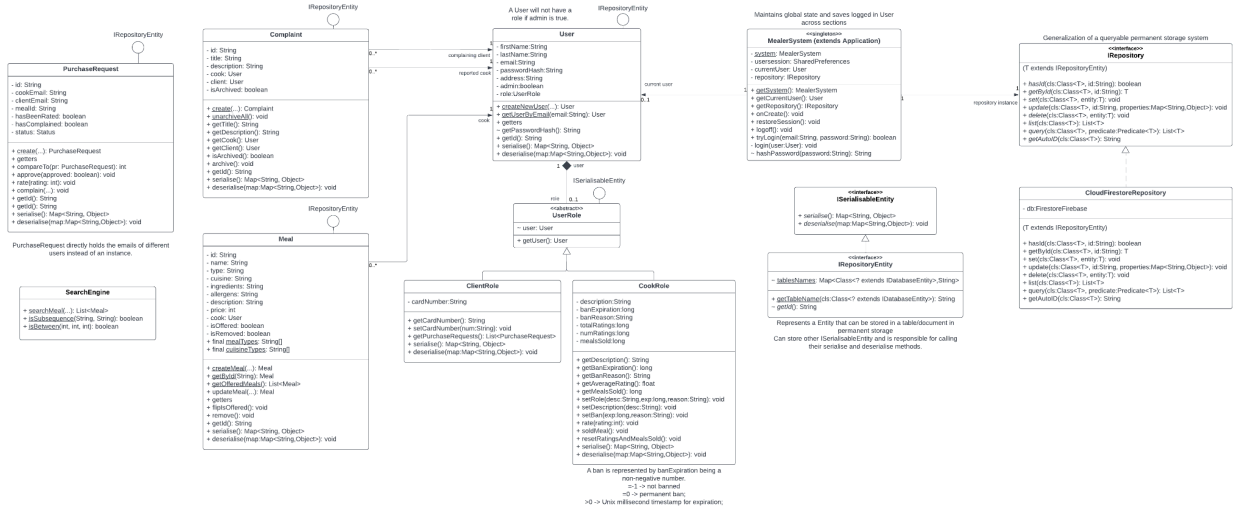
Screenshots **6**

Lessons **10**

Introduction

The Android project for the Fall 2022 semester was to develop a meal-sharing application supporting cooks (selling meals), clients (buying meals) and administrators (responding to complaints from clients about cooks). Besides providing an opportunity to explore the tools, libraries, and design patterns of Android development, the project was designed to emphasize general software development principles, from requirements to finish the project. This report summarizes our approach and results, including a UML class diagram, a table specifying group member contributions for each deliverable, screenshots for each screen of our finished app, and a review of some important lessons learned.

UML Diagram



Contributions

Deliverable 1

Adam	ClientRegister (Implementation), ClientHome, Validation of ClientRegister and CookRegister
Justin	Serialisation and Deserialisation of user data (User, CookRole, ClientRole) UI layouts, Validation function for various data types
Nicolas	MainActivity, ClientRegister (UI), CookRegister, Background Image, Checking void checks Added debug menu for ease of testing
Samuel	Documentation, Architecture Design, Model Class Skeletons, Repository implementation, MealerSystem Slight modifications to MainActivity (redirection to user homes)
Shyan	CookHome, AdminHome, implementation details with Justin

Deliverable 2

Adam	Created the Complaint model class to hold complaint information and updated CookRole to store ban information (All Non-UI)
Justin	Created the suspension menu the administrator can use to temporarily or permanently ban a cook, Validation
Nicolas	Created the ComplaintActivity that the administrator uses to view and action a complaint
Samuel	Documentation, Basic UI layouts, Planning, Unit Tests Modified AdminHome to display a list of all unresolved complaints
Shyan	Created the home menu for suspended cooks, displays the appropriate message and time until unban (if applicable)

Deliverable 3

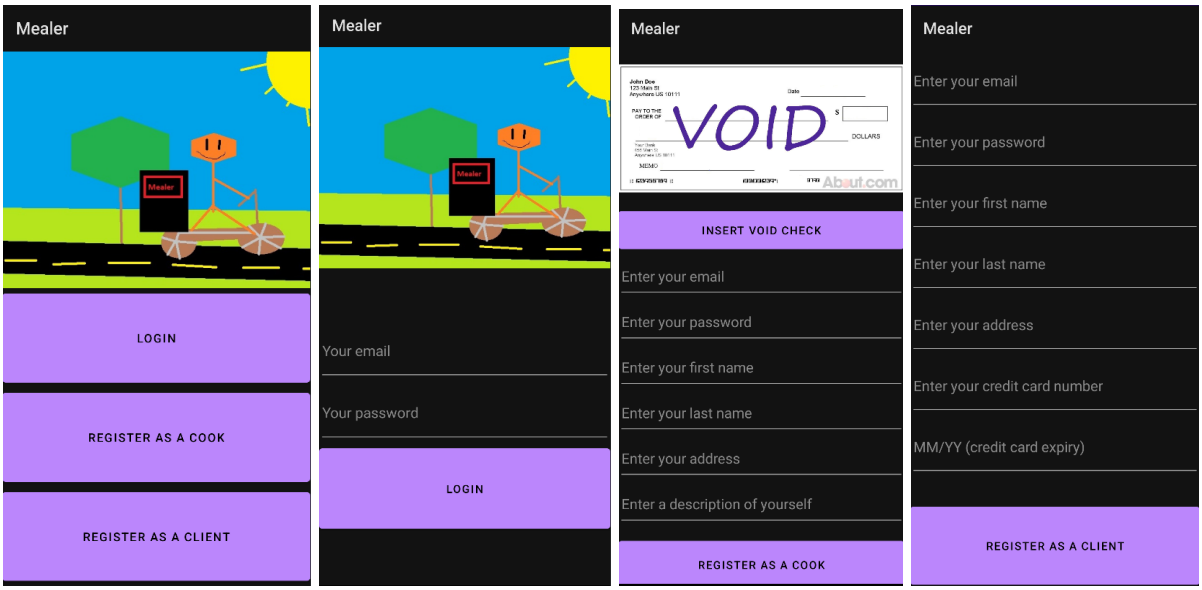
Adam	Created the Meal model class to hold meal information and updated CookRole to manage ratings and number of meals sold (All Non-UI)
Justin	Created the cook's menu managing activity to list a cook's meal and display meal information. Redirects to the meal creation/update menu.
Nicolas	Created the activity to create and update meals, Validation
Samuel	Documentation, Basic UI layouts,Planning, Unit Tests Created the activity that lets a cook change their description.
Shyan	Updated the CookHome to display the cook's name and statistics. Redirects the user to other activities.

Deliverable 4

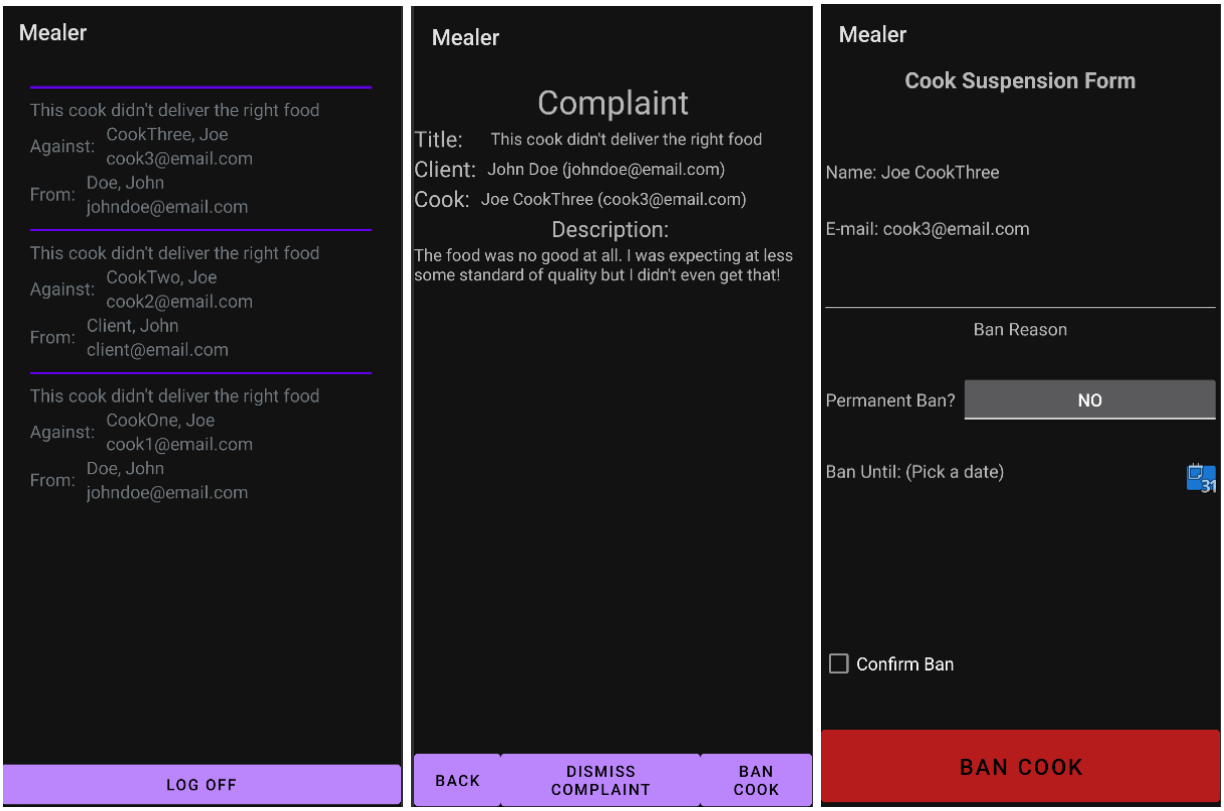
Adam	Created the PurchaseRequest model class to hold request information. Updated ClientRole and CookRole to return lists of purchase requests. Updated Meal to return offered meals from non-banned cooks
Justin	Created activities responsible for listing and managing purchase requests for both the clients and the cooks New app icon
Nicolas	Updated ClientHome to display a welcome message to the client and to redirect to other activities.
Samuel	Documentation, Planning, Unit Tests. Created the search engine and the search activity clients can use. Created the purchase activity that lets clients buy a specified amount of an offered meal + Validation
Shyan	Update CookHome to allow redirection to the purchase managing menu. Create the activity that lets a client change their credit card number. Create the activity that lets a client complain about an approved meal request

Screenshots

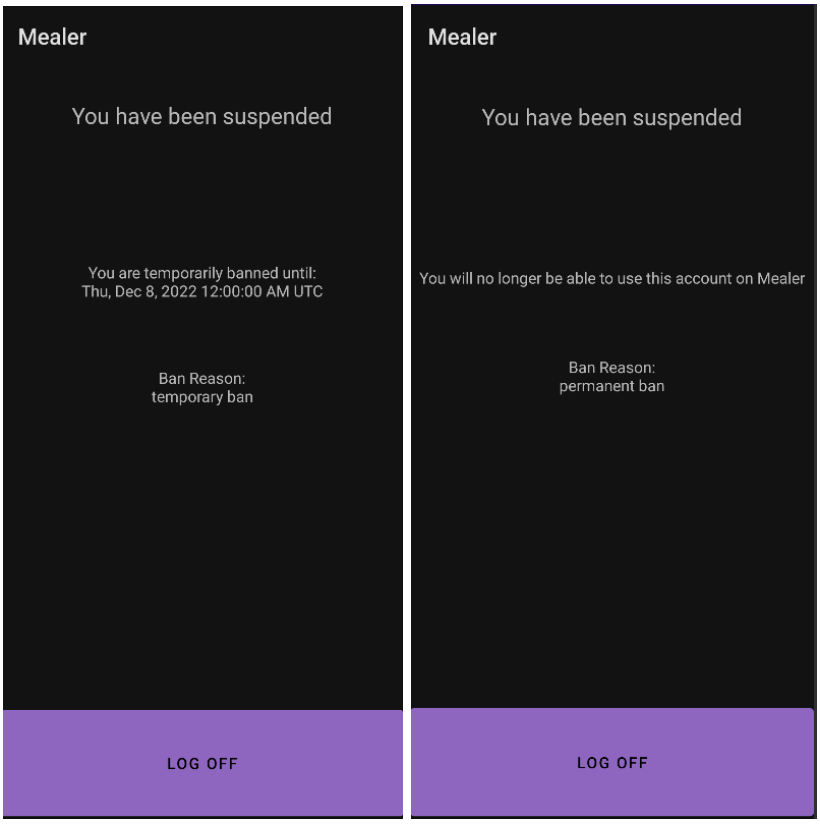
LoginActivity (MainActivity), LoginPage, CookRegister, ClientRegister



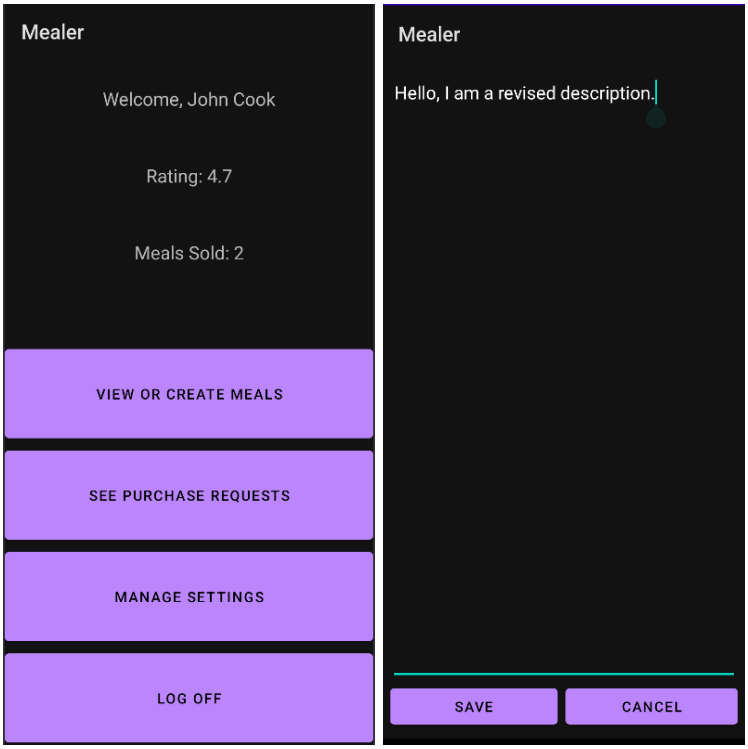
AdminHome, ComplaintActivity, SuspensionActivity



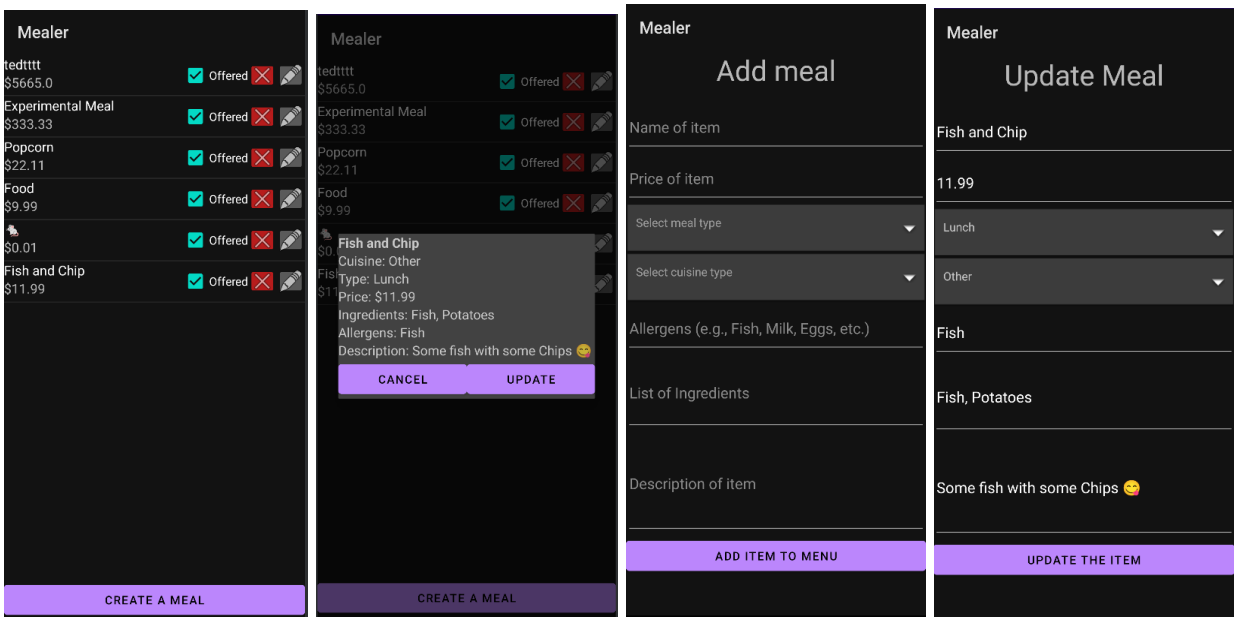
SuspensionHome (Temporary ban, Permanent Ban)



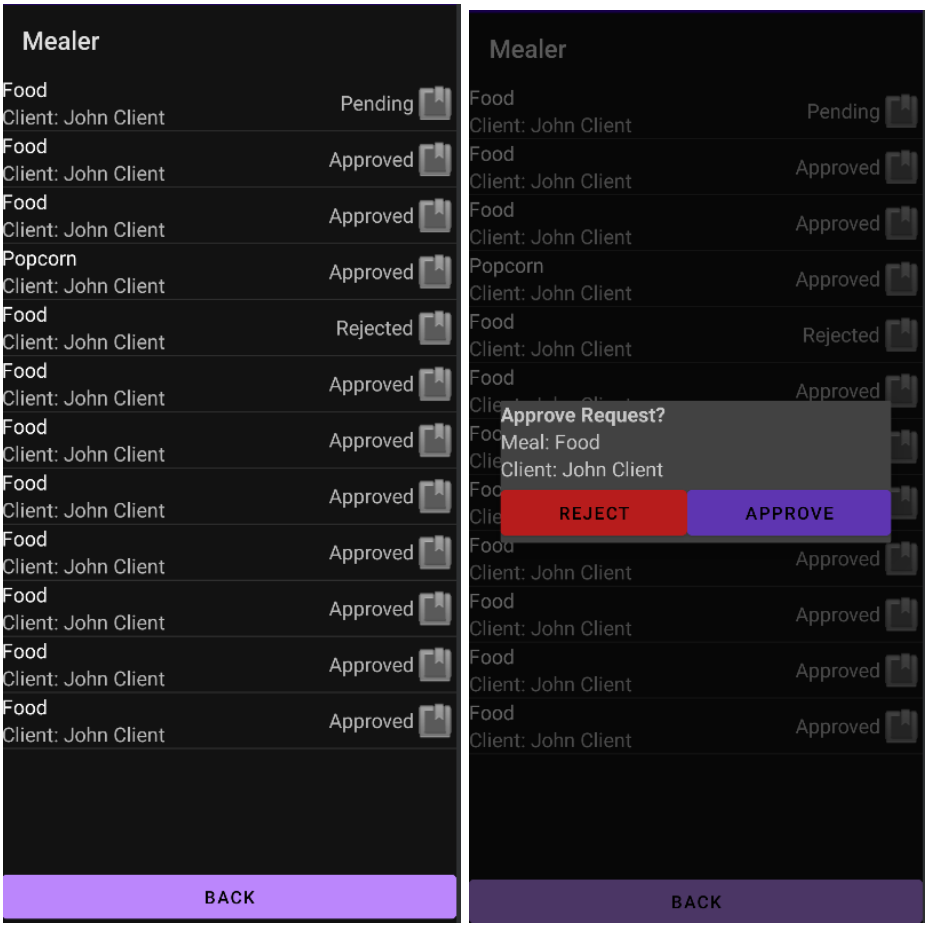
CookHome, CookSettingsActivity



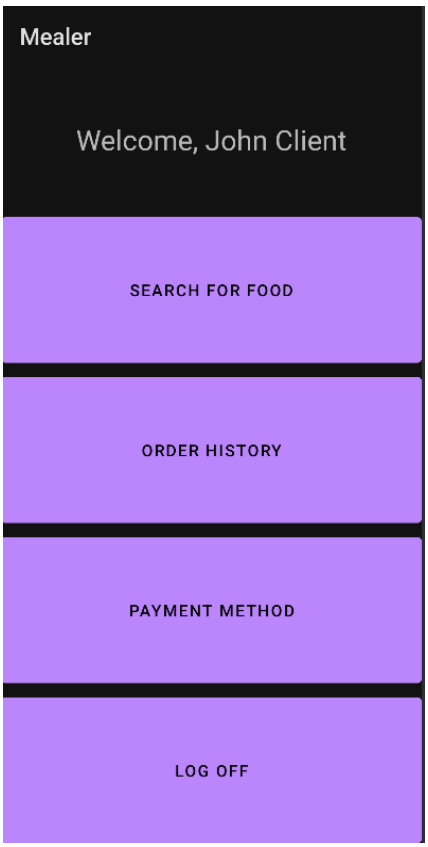
MealListActivity, MealListActivity (Info), CookAddMeal, CookAddMeal (Update)

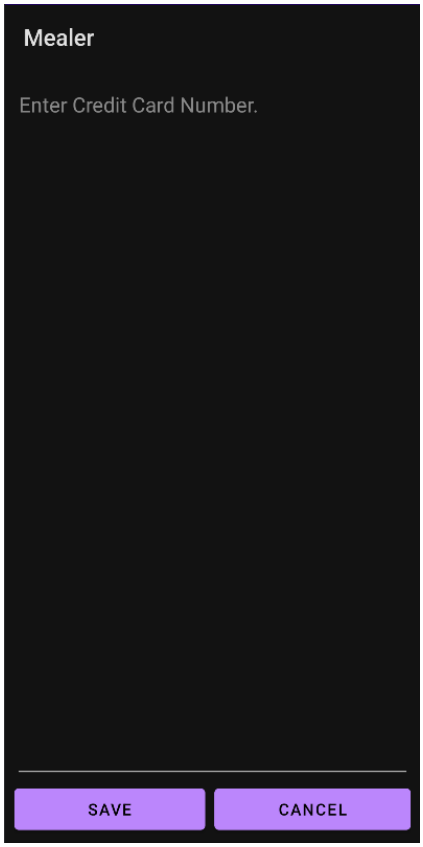


ViewCookRequestActivity, ViewCookRequestActivity (Approval/Rejection)

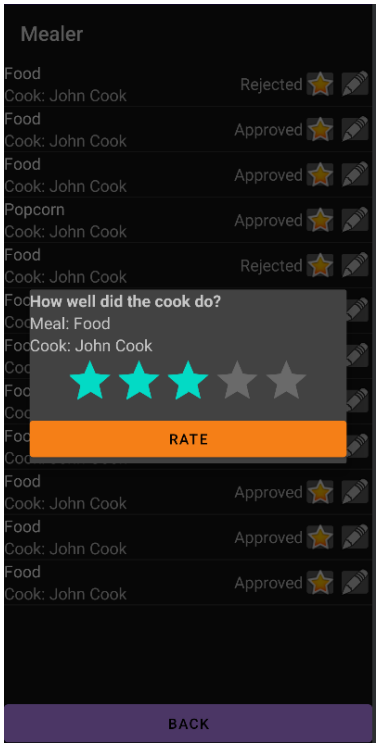


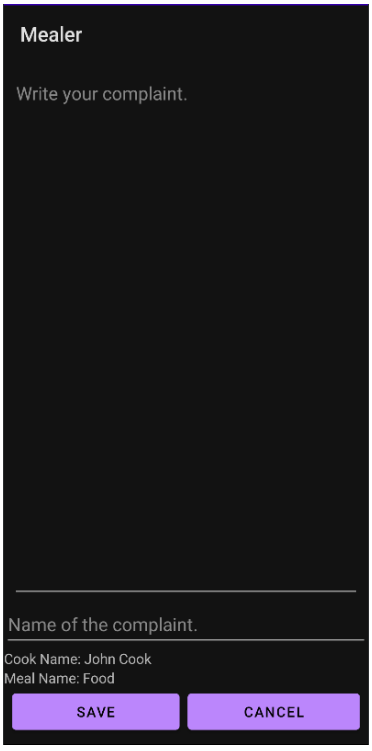
ClientHome, CreditCardActivity



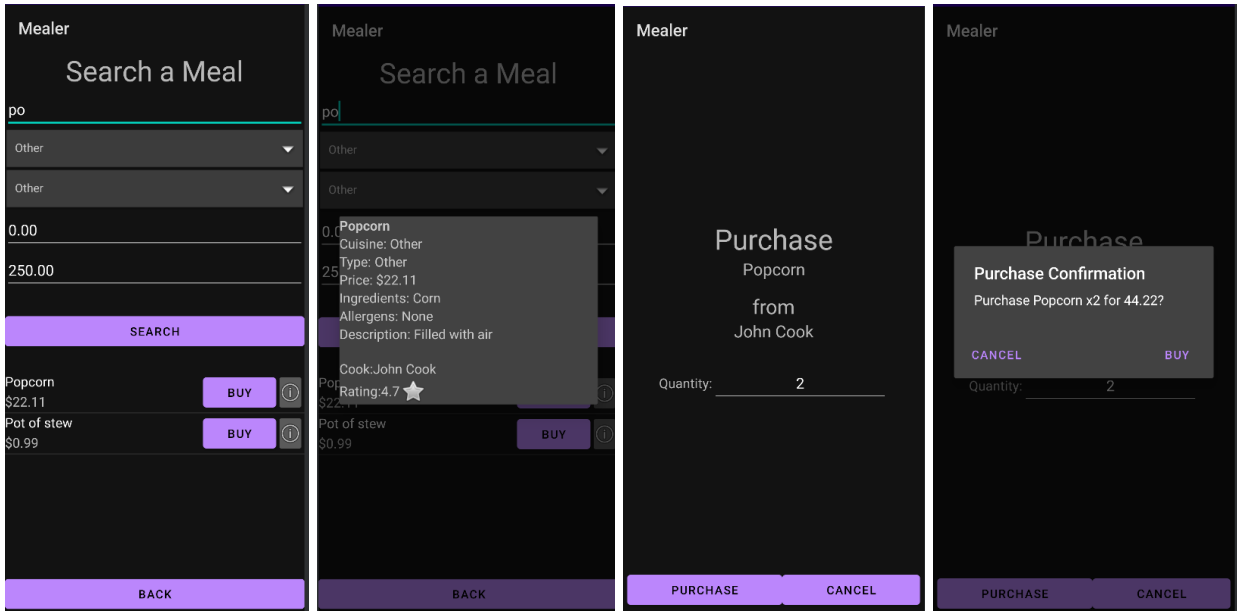


ViewClientRequestActivity, ViewClientRequestActivity (Rating), ClientComplaintActivity





SearchActivity, SearchActivity (Info), PurchaseActivity, PurchaseActivity (Confirmation)



Lessons

We would like to highlight three main lessons from the project:

1. Expect the first deliverable to be surprisingly time-consuming. On paper, the first project deliverable had a relatively short list of requirements, and in terms of app capabilities those listed were quite simple. What doesn't show up in this list is the amount of work that goes into planning and setting up the framework to get those initial capabilities off the ground. It seems likely that this less visible up-front work would be a feature of almost any project, so it's probably best to go in expecting the initial steps to take some time.

2. Splitting the work into planned deliverables is essential. The most obvious reason for setting intermediate deadlines is to avoid putting decisions and final implementations off until it becomes challenging to meet deadlines at all, and that's probably a good enough reason on its own. However, another benefit of splitting up the work this way is the opportunity it provides to move between a higher-level view of the project and a detail-level view. Working on the implementation of specific features tended to draw us into the details: exactly what sort of verification is ideal for this name text field, do we need an int or a long for the UTC timestamp, etc. Finishing a deliverable provided an opportunity to step back and revisit the big picture: is the design still working as we hoped? Is usability suffering in some unexpected way that we didn't notice when we were down in the details?

The split into deliverables was also essential for debugging. Getting all the bugs out of deliverable 1 provided a tested foundation for deliverable 2 and made it more likely that the bugs that needed to be tracked down for deliverable 2 were where they ought to be: in the new code. Then deliverable 2 provided the same service for deliverable 3, and so on.

3. A good design really does help. The design work that went into the first deliverable paid off with the remaining ones. As we added new capabilities and code, it was rare for there to be any hesitation about what belonged where or how the new pieces would fit with the old. In this case, design success was made a little more likely thanks to the detailed requirements of the project description. So Lesson 3.5 is that it's essential to create those requirements, and at a similarly high level of detail, if you don't already have them to begin with.