School of Electrical Engineering and Computer Science | École de science informatique et de génie électrique

University of Ottawa | Université d'Ottawa

SEG2105 | Fall 2022

uOttawa

SEG 2105 - Intro To Software Engineering
Mealer Final Report

Class Section: **A** Group: **20**

Date: December 9th, 2022

Team members:

Name:	Student ID:
Ibrahim Daoud	300186163
Layan Omar	300249213
Jiayi Ma	300263220
Sihan Sun	300218907
Emiliano Bustamante	300229811

For the fall semester course offering of SEG 2105, students are tasked with creating an android app called "*Mealer*" using Java in Android Studio. The app allows for three types of users; client, cook and administrator. All users upon signing in with the correct credentials will be able to see their role and have access to certain conditions. Cooks can create meals and offer them to be sold. Users can search for meals and purchase them and are also able to rate a cook as well as submit a complaint. The administrator role is hard-coded in the app and can not be signed up for. Their role is to view complaints and address them either by dismissing them (for something minor), or they can suspend them for a certain period of time (for something more serious).

In a way, this app can be thought of as a competitor of UberEats where people living in a city may opt to eat home-cooked meals rather than food from restaurants.

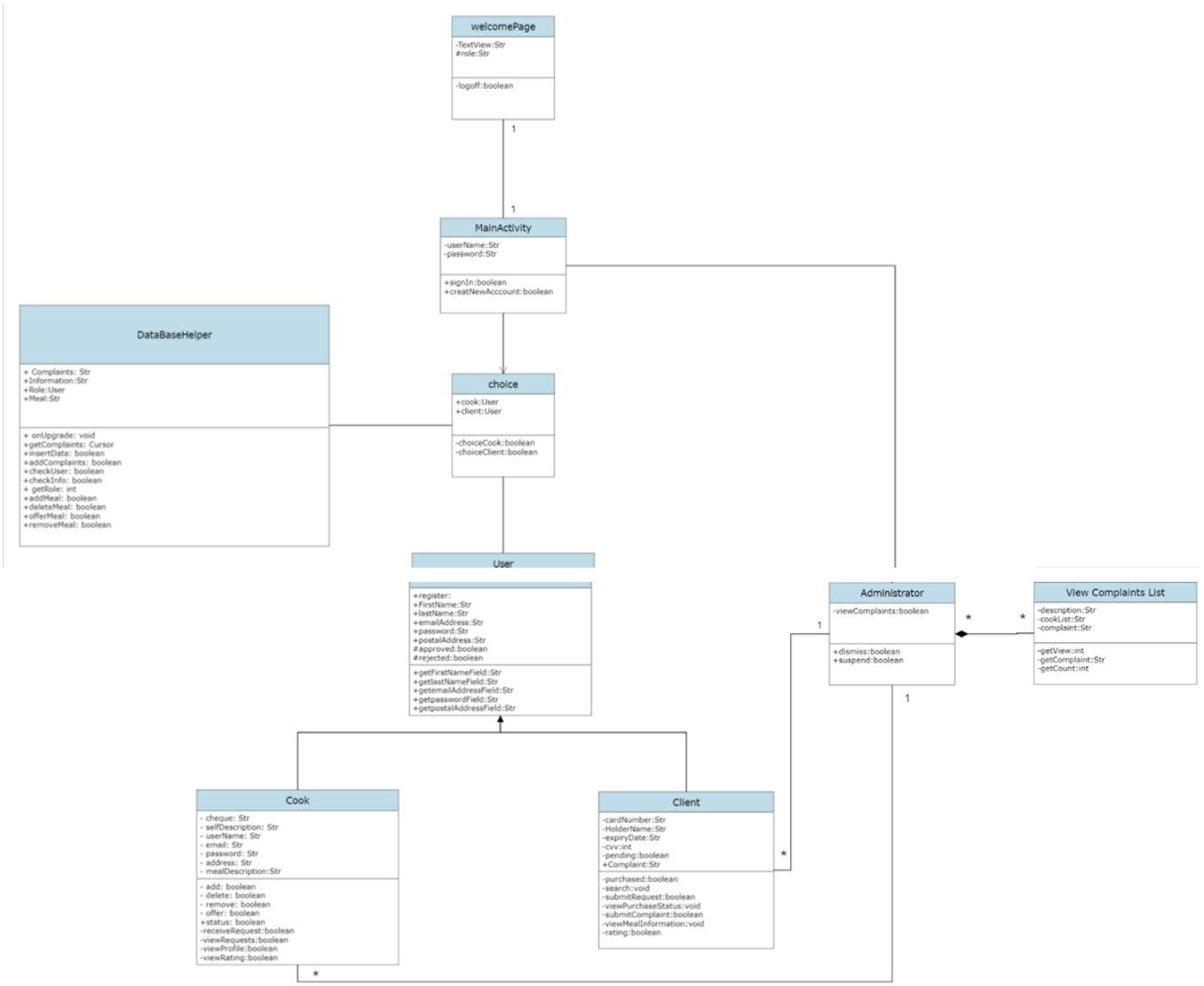
This version of Mealer is created by Group 20, with the following members; Ibrahim Daoud, Layan Omar, Jiayi Ma, Sihan Sun & Emiliano Bustamante. The group worked on the app throughout the semester in deliverables, where the goal of each deliverable is to target a certain aspect of the app's functionality until the app is mostly done.

As any app which has login credentials, a database must be used. There are several options available but SQLite was chosen as the database to be used since it has valuable options. It was also chosen as a way to get familiarised with SQL as it is an important language to understand for databases.

With the addition of a database and other features implemented in the app, some things may get slightly confusing and so there was a UML diagram depicting the app at each deliverable. Below is the UML diagram at deliverable 4 showing the finalised app.

Updated UML Diagram:

*****Note: A more clear version of this UML diagram is included in the github repository*****



Contributions From Each of The Team Members:

= Ibrahim
= Layan
= Jiayi
= Sihan
= Emiliano

Deliverable 1 Marking Scheme

Feature or Task	**% Weight (out of 100)**
The team created in GitHub classroom contains all members of the group	10
Each member of the group has made at least **one** commit to the repository.	20
UML Class diagram of your domain model: • (-2 for each missing class) • (-2 for each incorrect generalization) • (-0.5 for each incorrect multiplicity) • (-0.5 for each missing attribute)	25
The APK is submitted	5
A user can create a ***Client*** or ***Cook*** account.	10
The ***Administrator***, ***Cook***, or ***Client*** user can see the "welcome screen" after successful authentication. The welcome screen specifies the user role.	10
The user can log off.	10
All fields are validated. There are appropriate error messages for incorrect inputs. (-1 for each field in which the user input is not validated)	10
Optional – The group uses a DB (e.g., Firebase, SQLITE, or another similar technology)	+5 (bonus)

Deliverable 2 Marking Scheme

Feature or Task	**% Weight (out of 100)**
Updated UML Class diagram of your domain model • (-2 for each missing class) • (-2 for each incorrect generalization) • (-0.5 for each incorrect multiplicity) • (-0.5 for each missing attribute)	15
The APK is submitted	5
You implemented 4 Unit test cases (simple local tests). There is no need to include instrumentation or Espresso Tests (UI).	20
When a user registers, their account information is stored in the DB	10
The ***Administrator*** can view the list of complaints	10
The ***Administrator*** can action the complaint (dismiss complaint or suspend ***Cook***) For a temporary suspension, the ***Administrator*** can specify the date when the suspension can be lifted The complaint disappears from the ***Administrator's*** list once it is actioned	15
The complaints list is stored in the DB	15
The ***Cook*** can see a message informing them that they have been suspended For temporary suspension, the message informs them when the suspension will be lifted For permanent suspension, the message informs them that they can no longer use the application	10

Deliverable 3 – Mostly Cook Features

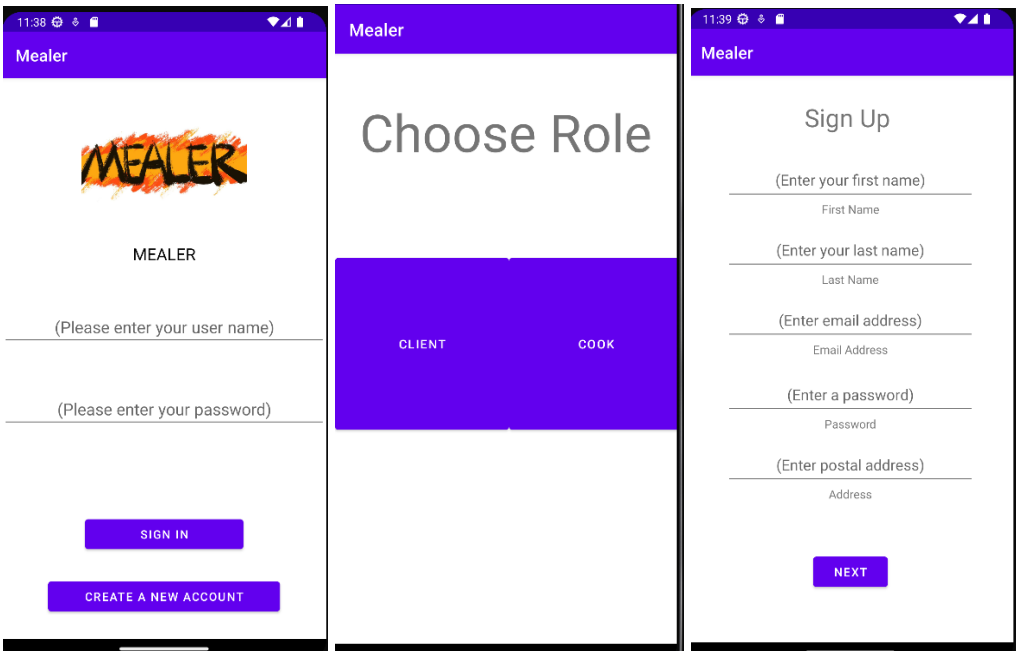
Feature or Task	% Weight (out of 100)
Updated UML Class diagram of your domain model • (-2 for each missing class) • (-2 for each incorrect generalization) • (-0.5 for each incorrect multiplicity) • (-0.5 for each missing attribute)	15
The APK is submitted	5
You implemented 4 Unit test cases (simple local tests). There is no need to include instrumentation or Espresso Tests (UI).	20
The ***Cook*** can add a meal to the *menu*	10
The ***Cook*** can add a meal to the *offered meals list*	10
The ***Cook*** can delete a meal from the *menu* The ***Cook*** cannot delete a meal from the *menu* if it is currently in the *offered meals list*	10
The ***Cook*** can remove a meal from the *offered meals list*	10
When a suspended ***Cook*** logs on, they can only see the suspension message and log off. They cannot perform any other action.	10
All fields are validated. There are appropriate error messages for incorrect inputs. (-1 for each field in which the user input is not validated)	10

Deliverable 4 Marking Scheme

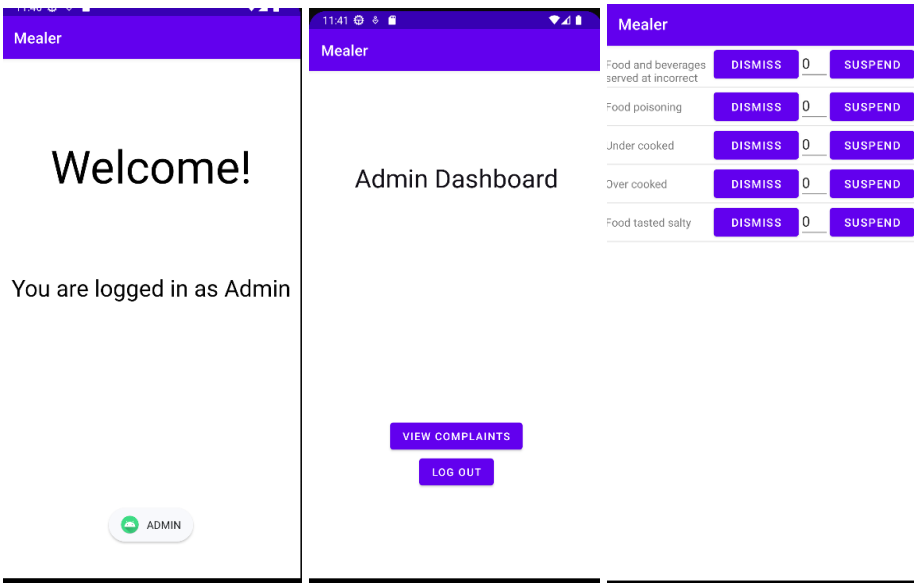
Feature or Task	**% Weight (out of 100)**
Updated UML Class diagram of your domain model • (-2 for each missing class) • (-2 for each incorrect generalization)	10
• (-0.5 for each incorrect multiplicity) • (-0.5 for each missing attribute)	
The APK is submitted	5
You implemented 4 Unit test cases (simple local tests). There is no need to include instrumentation or Espresso Tests (UI).	10
Final report including: • Title page (2.5 points) • Short Introduction (2.5 points) • Updated UML class diagram • Table specifying the contributions of team members for each deliverable (10 points) • All the screenshots for your app (10 points) • Lessons learned (5 points)	30
The ***Client*** can search for a meal The ***Client*** can see search results for meals offered by non-suspended ***Cooks***	5
The ***Client*** can view the ***Cook's*** information and rating for each meal in the search result The ***Client*** can view the meal's information for each meal	5
The ***Client*** can submit a purchase request The ***Cook*** can receive the purchase request submitted by the ***Client***	5
The ***Client*** can view the status of their purchase (pending, approved, or rejected)	5
The ***Client*** can rate the ***Cook*** from which they have purchased a meal	5
The ***Client*** can submit a complaint about a ***Cook*** to the ***Administrator***	5
The ***Cook*** can view and approve/reject purchase requests received from ***Clients***.	5
The ***Cook*** can view their profile and rating.	5
All fields should be validated. There should be appropriate error messages for incorrect inputs. (-1 for each field in which the user input is not validated)	5
Optional – ***Client*** receives a notification when their purchase request is approved/rejected	+10 (bonus)

Screenshot For The Mealer App:

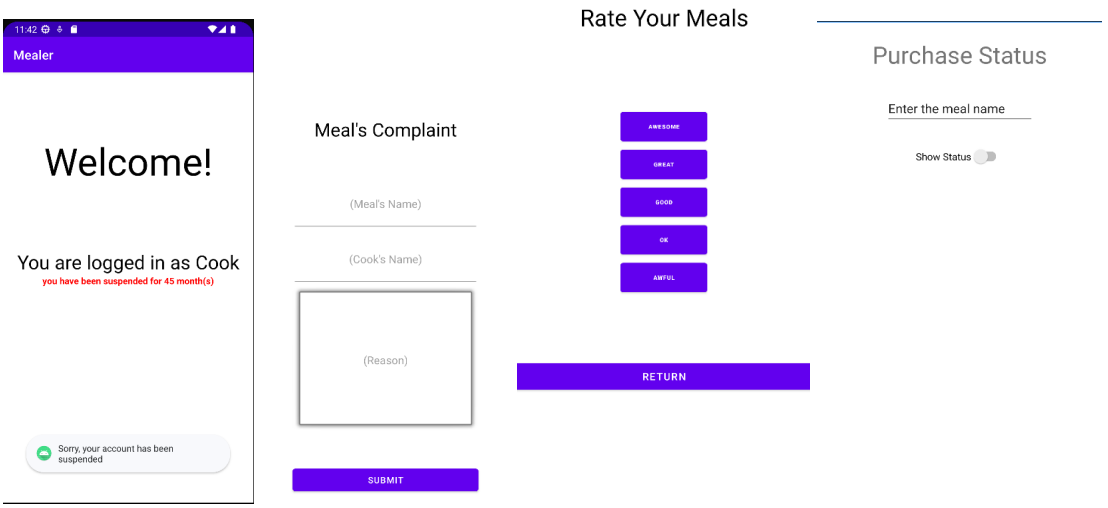
Sign in page & registration:



Role notification page, admin dashboard & complaints list



Cook Suspension page:



Lessons Learned:

Throughout this semester, there have been 5 key dates in regards to the project. Deliverables 1-4, as well as a project demonstration showcasing up to deliverable 3.

At times it has been rough submitting some deliverables because some teammates might become unreachable at the time on the day of the submission and their part(s) of the project are still incomplete. One lesson that can be learned from working on an Android Studio project – or any university project, is the importance of effective communication and planning. It is important to set clear expectations and deadlines for each team member, and to regularly check in with each other to ensure that everyone is on track. Additionally, it can be helpful to have a backup plan in place in case a team member is unable to complete their tasks on time, such as assigning their tasks to someone else or rearranging the project timeline. It is also important to be flexible and adaptable, as unexpected delays or challenges are common in team projects. Overall, effective communication and planning can help ensure that the project is completed successfully, even when team members do not always finish their tasks on time.

Luckily, software engineering students must take a course called GNG 2101 in the same semester as SEG 2105 where they teach about the importance of effective communication and teamwork when doing a group project with other students. Much of the knowledge and skills obtained and learned in that class are transferable to other things such as the group project for this class where we can further refine these skills and even apply it into our careers as software engineers.

Another lesson that was learned is the importance of Version Control Software (VCS) such as github. Prior to VCS, people would save different versions of their code locally on their machine which could technically work but things tend to get a little bit tricky when working in a team project. It is very inefficient to be sending files back and forth between teammates when each person is working on a different part of the code. This can be especially tricky for Android Studio due to the nature of the Gradle Build files and how they can work a little bit differently based on the dependencies installed and various other factors. Sometimes this can even cause the project to not be able to build itself. This was experienced first hand as some of the members on the team were having issues with Github and were not able to correctly commit changes to the repository, and instead would send their piece of code to another member so they would be able to correctly push it to Github. These issues were later resolved but it's important to address issues as soon as they come up since in our case it caused some bugs due to the way the Gradle Build files were being transferred.