SEG2105 – Introduction to Software Engineering – Fall 2022

Android Project: Mealer App (20%)

Instructions

1. You will work in the same team you have joined for your labs
2. You will submit each deliverable as a release on GitHub. You will submit to brightspace a zip file that has the apk file and all other submission material, in addition to a README.txt file that contains the following information:
    a. Link to the github repository you used where all team members are contributing.
    b. Names and student numbers of all team members
3. At the end of the semester, the team must demonstrate a completed application. For instance, one team member demonstrating a screen with one functionality while another member showing another screen with another functionality (running on a different phone), **WILL NOT be accepted**. The team must produce a single application with all the required features.

Academic Honest

All the work that you do towards the fulfillment of this course's expectations must be your own unless collaboration is explicitly allowed. Viewing or copying another individual's work (even if stored in a public directory) or lifting material from a book, magazine, website, or other source-even in part-and presenting it as your own constitutes academic dishonesty, as does showing or giving your work, even in part, to another student.

Project Description

You are developing an Ottawa-based meal sharing application called *"Mealer"* where local cooks can sell meals to clients from their home. The application supports three types of users:

- ***Cook***: a user that makes meals at home and sells them to ***Clients***.
- ***Client***: a user that buys meals from ***Cooks***. They order the meal through the application and pick it up from the ***Cook's*** home.
- ***Administrator***: a user that receives complaints about a ***Cooks*** from a ***Client*** and may suspend the ***Cook*** if necessary.

In the next sections, we will describe the application in detail from the perspective of each user.

Client

To become a ***Client***, a user must register by submitting the following information:

- First name
- Last name
- Email address (which also serves as the username)
- Account password
- Address
- Credit card information (for payment purposes)

Once registered, a ***Client*** can search for meals by specifying one or more of the following information:

- Meal name
- Meal type
- Cuisine type

The returned results should only show meals that are currently offered by ***Cooks***. Moreover, the result for each meal shows the ***Cook's*** information (name, address, and description they wrote about themselves) and the meal's information (price, meal type, cuisine type, list of ingredients, allergens, price, and description). The ***Client*** can then select a meal, specify a pick-up time, and submit a request to purchase the meal. The ***Client*** can view their purchase requests. Each purchase request can have one of three states:

- Pending: the ***Cook*** has not processed the request yet.
- Approved: the ***Cook*** has approved the request and therefore the purchase is completed and the ***Client*** can pick-up the meal at the time they specified.
- Rejected: the ***Cook*** has rejected the request and therefore the purchase is not completed.

A ***Client*** can view the list of meals they purchased. From the list, they can press on a meal to rate its ***Cook*** and/or submit a complaint. The complaint is assessed by the ***Administrator*** which can dismiss it or suspend the ***Cook***.

Administrator

The ***Administrator*** is a pre-registered user. This means that the ***Administrator's*** account information (i.e., username and password) are already in the database when the system is first launched.

The ***Administrator*** has an inbox to receive complaints from ***Clients***. Each complaint is associated with a ***Cook*** and contains a description of the complaint written by the ***Client***. Based on the complaint, they have the authority to suspend the ***Cook*** temporarily or indefinitely. They can also dismiss the complaint. Once the ***Administrator*** actions the complaint (dismisses or suspends), then the complaint disappears from their list.

Cook

To become a ***Cook***, a user must register by supplying the following information:

- First name
- Last name
- Email address (which also serves as the username)

- Account password
- Picture of a void cheque (so that they can be paid)
- Address (from which the Client can pick-up the meal)
- Short description of themselves

Once registered, the ***Cook*** can add meals to their *menu*. To add a meal to the *menu*, they must specify the following information:

- Meal name
- Meal type (e.g., main dish, soup, desert, etc.)
- Cuisine type (e.g., Italian, Chinese, Greek, etc.)
- List of ingredients
- Allergens
- Price (the price they would like to charge the Client for the meal)
- Description (other information that the ***Cook*** would like to add, such as how many people is the meal intended for)

The ***Cook*** can also delete a meal from the *menu*.

A meal added to the menu does not necessarily mean that it is currently offered by the ***Cook***. It simply means that it can be offered by the ***Cook***.

To offer a meal, the ***Cook*** must select the meal from their *menu* and confirm that they would like to offer it. Then, the meal is added to the ***Cook's*** *offered meals list* and will start appearing in ***Client*** searches. The ***Cook*** can remove the meal from the *offered meals list*, which means ***Clients*** can no longer view it in searches or purchase it.

A ***Cook*** has list of purchase requests. The list contains the purchase requests submitted by the ***Clients***. A ***Cook*** can view the purchase requests and approve or reject them. A purchase request specifies the meal, ***Client*** name, and pick-up time.

A ***Cook*** can view their profile which shows the data they specified during registration along with the following information:

- Number of meals sold
- Rating

The rating is a number between 1 and 5 and corresponds to the average rating she/he has received from ***Clients***.

User Interface Design

This course does not focus on user interface design. However, students are encouraged to produce aesthetic user interfaces. Consider the Android Design Guidelines when designing your application: https://developer.android.com/design.

Deliverables

The project is divided into 4 incremental deliverables. Students are required to submit each deliverable by the posted deadline online using Brightspace.

Deliverable	Due date
1 – GitHub and User Accounts (3%)	October 21st
2 – Mostly Administrator Features (3%)	November 4th
3 – Mostly Cook Features (3%)	November 18th
4 – Mostly Client Features (and Integration) (9%)	December 7th
5 – Demo (2%)	Last week of classes

The project must be carried out throughout the session and students are **strongly** encouraged to maintain a log of their project activities as such information will be needed for the final report.

Your application must be written in Java and built using Android Studio (you can use another IDE, but the TAs will only provide support for Android Studio). By the end of the semester, you must implement and submit a working application based on the specifications. Firebase or SQLite can be used for storing and retrieving the application data.

Deliverable 1 – GitHub and User Accounts

(Read the Project Description section carefully before you start working on this deliverable.)

In this deliverable, you need to implement the ***Client*** and ***Cook*** account management component. The app needs to allow ***Clients*** and ***Cooks*** to register for user accounts. ***Administrators*** do not need to register as they are pre-registered.

Once the user logs in, they should see a second screen with the following message, 'Welcome! You are logged in as "role" (where "role" can be ***Cook***, ***Client***, or ***Administrator***). The user can also log off. No additional functionality needs to be implemented at this point.

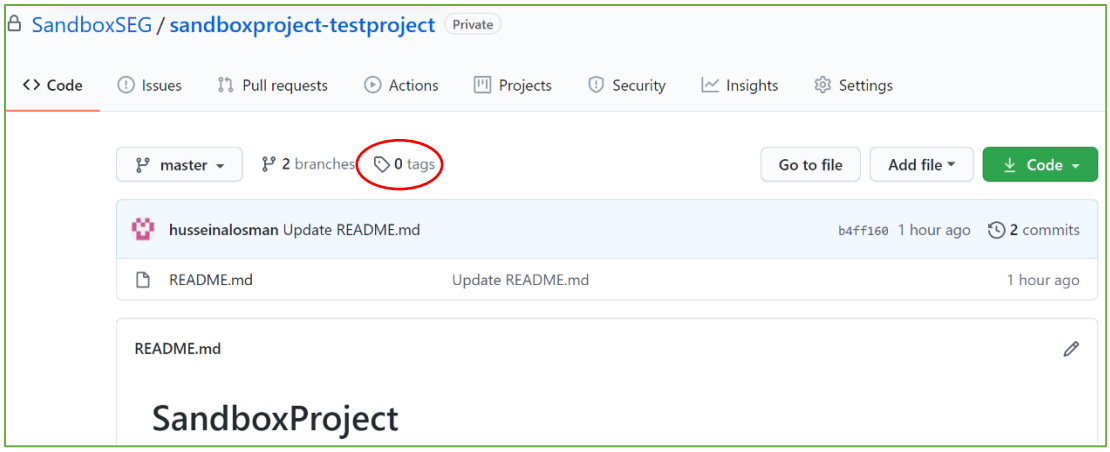
You can use Firebase or SQLite for DB support. If you do not use a DB at this point, you can simply store the information in memory (but it would be lost when you terminate the app) or on a file.

In your GitHub repository, include a PDF document that contains an UML Class diagram of your domain model. This will only include the UML classes related to deliverable 1. Moreover, in your repository, provide a README file that contains the credentials needed to sign-in as ***Administrator***.

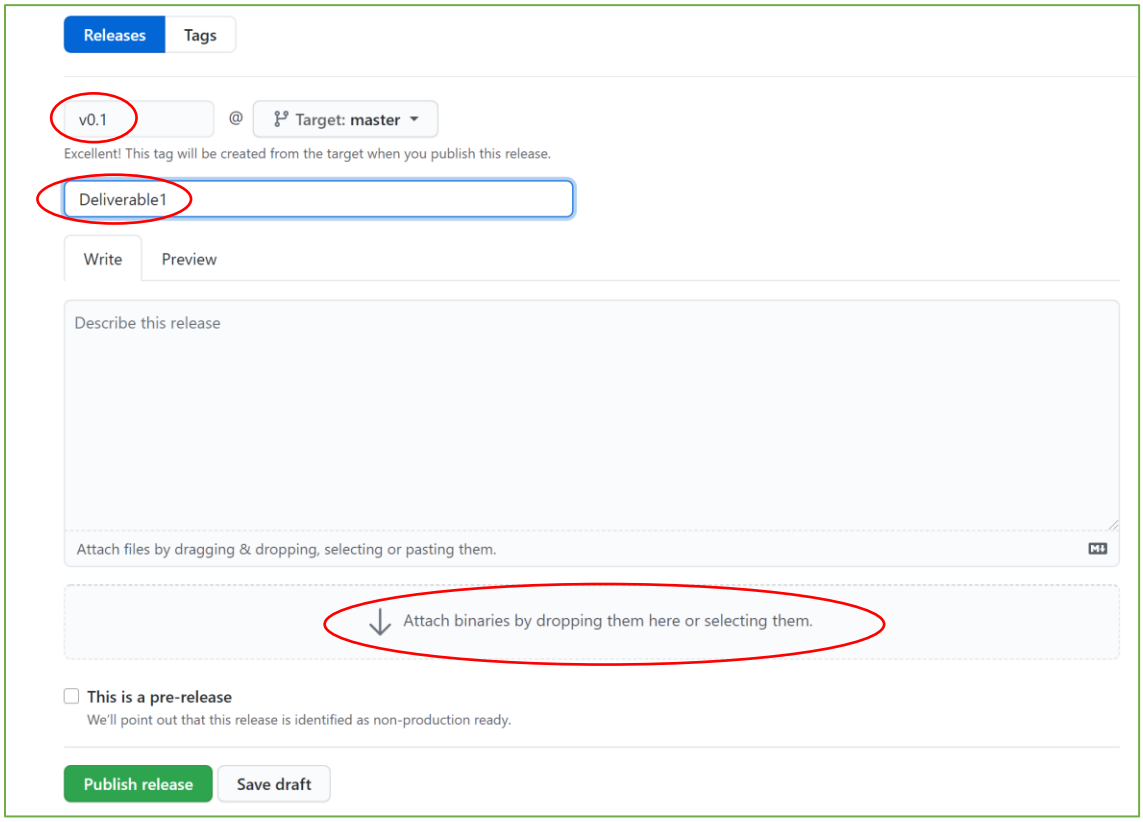
Submission Instructions

To submit your code, you will create a release from your repository as follows:

1. In your repository, click on "tag" (see figure below)



2. Click on the “Create new release” button and fill the form as follows (see figure below)
    a. For “Tag version” enter v0.1
    b. For “Release title” enter Deliverable1
    c. From the section of the form where you can attach binaries, upload the APK of your app (the APK can be found in <yourAndroidProject>/app/build/outputs/apk/app-debug.apk).
    d. Make sure to rename the APK after the name of your team “group1_debug_apk”.



Deliverable 1 Marking Scheme

Feature or Task	**% Weight (out of 100)**
The team created in GitHub 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 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 – Mostly Administrator Features

(Read the Project Description section carefully before you start working on this deliverable.)

In this deliverable, you need to implement the ***Administrator*** functionality. Moreover, if you have not done so, you should start using a DB (e.g., Firebase, SQLITE, or another similar technology).

The app should allow the ***Administrator*** to view a list of complaints. For each complaint, the ***Administrator*** can either dismiss the complaint or suspend the ***Cook*** **associated** with the complaint temporarily or indefinitely. Once the ***Administrator*** actions the complaint (dismisses or suspends), then the complaint disappears from the list of complaints.

The complaints should not be hard coded. They should be stored in a DB. However, given that ***Clients*** cannot yet purchase meals and submit complaints at this stage, you will pre-create a list of complaints stored in the DB. Each complaint should be associated with a registered ***Cook*** (you already implemented the registration functionality for deliverable 1).

If an ***Administrator*** suspends a ***Cook*** based on a complaint, then once that ***Cook*** logs on, they should see a message that informs them that their account is suspend and when the suspension will be lifted in the case of temporary suspension.

You must include at least **4 additional unit test cases** (simple local tests). There is no need to include instrumentation or Espresso Tests (UI).

In your GitHub repository, include a PDF document that contains an UML Class diagram of your domain model. This will only include the UML classes related to deliverables 1 and 2. Moreover, in your repository, provide a README file that contains the credentials needed to sign-in as ***Administrator***.

Submission Instructions

To submit your code, create a release v0.2 in your repository. Make sure the APK file is added to your release.

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 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*** 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

Optional – The group integrates with CircleCI to see the automated builds and automated test units execution.	+5

Deliverable 3 – Mostly Cook Features

(Read the Project Description section carefully before you start working on this deliverable.)

For this deliverable, you will implement the majority of the ***Cook's*** features.

The ***Cook*** must be able to add/delete meals from their menu. They must be able to add/remove meals in their *menu* from the offered meals list. A suspended ***Cook*** should no be able to perform any of the latter actions.

You must include at least **4 additional unit test cases** (simple local tests). There is no need to include instrumentation or Espresso Tests (UI).

In your GitHub repository, include a PDF document that contains an UML Class diagram of your domain model. This will only include the UML classes related to deliverables 1, 2, and 3. Moreover, in your repository, provide a README file that contains the credentials needed to sign-in as admin (if a predefined admin account has been created).

Submission Instructions

To submit your code, create a release v0.3 in your repository. Make sure the APK file is added to your release.

Deliverable 3 Marking Scheme

Feature or Task	**% Weight (out of 100)**
Updated UML Class diagram of your domain model • (-2 for each missing class) • (-2 for 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 – Mostly Client Features (and Integration)

(Read the Project Description section carefully before you start working on this deliverable.)

For this deliverable, you will implement the remainder of the app's functionality.

The ***Client*** must be able to search for meals by specifying the meal name, meal type, and/or cuisine type. The search result should show the meals currently offered by non-suspended ***Cooks*****.** For each meal, the ***Client*** must be able to view the ***Cook's*** information and rating and the meal's information. The ***Client*** can then submit a purchase request for a meal they select from the search results.

The ***Client*** must be able to view the status of their purchase request. For simplicity, once a purchase request is approved by the ***Cook***, we assume that the meal price is charged to the Client's credit card (which was entered during registration) and the purchase process is completed. You will not implement any functionality related to charging of the credit card.

The ***Client*** must be able to rate the ***Cooks*** from which they purchased meals *(for simplicity, if a purchase request is approved by the* ***Cook****, that means that the meal has been purchased)*. Moreover, they can submit a complaint about a ***Cook*** from which they purchased a meal.

The ***Cook*** must be able to view and approve/reject purchase requests received from ***Clients*****.** The ***Cook*** must be able to view their profile which contains their information and rating.

You must include at least **4 additional unit test cases** (simple local tests). There is no need to include instrumentation or Espresso Tests (UI).

In your GitHub repository, include a PDF document that contains you final report (see the marking scheme for information on what to include in the document). Moreover, in your repository, provide a README file that contains the credentials needed to sign-in as ***Administrator***.

Submission Instructions

To submit your code, create a release v0.4 in your repository. Make sure the APK file is added to your release.

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 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