PROGRESS REPORT

OPEN QUESTIONS

- Would other Android architectural structures such as the MVP with controller or the MVVP fit better with our program?
- Should a class have multiple methods that each depend on different interfaces?
- How do we avoid creating extremely similar instance attributes in two classes for the purpose of retrieving information about entities? (For example, Dish List and Dish Information both process information about dishes in the menu, and both keep separate menu instance attributes).
- Now we are just using an online server to store data, instead of using a local database. How would the usage of local or server databases change our project structure, and how does database make the program more efficient?
- Our program is using google storage as a server to hold the data files, in order to simplify the process of generating a unique ID for each user, we decided not to implement the functionality of removing existing users. How does the database optimize this process and allow user deletion while keeping ID uniqueness?
- Now we are using the functionality provided by Google Maps to help our delivery staff to find the destination, will using a web app to display the map give the app cross-platform flexibility?
- Now we are using Android framework to deploy our app, if there is a user using iOS, how can we give our app cross-platform ability?

WHAT HAS WORKED WELL

- We are using the online server to store our data, so that when a user changes the data (for instance when the manager adds new staff or edit menu), the change can be applied to multiple devices. If we instead store data in internal storage, these changes cannot be seen by users other than the one who made the changes.
- We have implemented the Android GUI instead of the command-line UI for presentation of our project to the users. This design better corresponds with real-life situations and helps us develop knowledge in Android UI implementations.
- We implemented the MVP structure for clean architecture for Android GUI. This structure excludes all controller classes, only leaving presenters, so that the view classes (in our project the activity classes) interact with the presenter classes via the view interfaces to interact with use case classes via output boundaries, instead of interacting with use case classes directly which violates the dependency rule.
- We used the app to interact with Google map inside the android emulator to call the functionality of maps with destination information, it helps the deliver staff to design a route to the destination easily.
- We are getting more proficient with git-hub features. We used to have tons of conflicts each time we push and pull from main, but practice makes perfect,

and we can push and pull smoothly to main repo, review each other's changes and leave comments, assign issues to corresponding group member now.

- We are splitting tasks more appropriately now. We used to split them based on number of tasks and assign same deadlines, but in fact some of them are much harder to establish and some of them might be dependent on another, thus we worked inefficiently throughout phase1 and did not receive an ideal score. Now, as we dig deeper into our program, everyone gets familiar with it and understands the amount of time and effort required to accomplish a mission, hence our cooperation is becoming more adequate.

INDIVIDUAL PROGRESS

Evelyn Chou

In Phase 2, I implemented the Android GUI for functionalities related to viewing the menu and placing orders. I also implemented a new functionality where the user can edit the dishes they've added to their order, so they don't have to go back and create an entirely new order if they select the wrong dish or change their mind. I also implemented the MVP structure and fixed the issue regarding lack of output boundaries for classes related to these functionalities. Furthermore, I wrote tests for most of the entities and use cases.

Significant Pull Request 1:

https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/77

In this pull request 1, I implemented the Android GUI and added new output boundaries for the use cases.

Significant Pull Request 2:

https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/87

In this pull request 2, I applied the MVP structure to my code and refactored so that it fits Clean Architecture. Use case now updates values in presenters through an interface, and similarly the presenter updates the activity (which implements the view) through an interface.

Shaojie Dong

In Phase 2, I created the Android GUI for functionalities related to checking and updating inventories. I have also created an adapter for DeliveryInputBoundary where a delivery staff can be adapted as serving staff. I implemented the MVP structure for the android activities I created as well by letting them implement the view interface. I checked the code smell and reduced duplication in the inventory class as well.

Significant Pull Request 1:

https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/83#issue-1065554486

This is the pull request set the final version of the Android GUI for Inventory Staff.

Significant Pull Request 2:

https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/111#issue-

1071231016

This pull request adding additional design patterns to the project.

Naihe Xiao

I am responsible for the implementation Android GUI for functionalities of manager in phase2. This includes seeing the menu, seeing the reviews, deleting reviews under a certain rate decided by the manager, and editing or deleting a specific dish. I changed the packaging strategy from a single layer based purely on clean architecture to multiple layers: the first layer depends on the clean architecture while the second or third one are based on the functionality or user of the class. I implemented the MVP structure for all manager related functionalities including the view interfaces and output boundaries to avoid any hard dependencies between classes. I separated the use case classes that used to have multiple responsibilities into several classes, each only responsible for one functionality, so that our project follow the single responsibility principle. I added iterator design pattern for review list class. I participated in the design of data access problems caused by the implementation of Android GUI, and decided the usage of online server to store data. I specifically cooperated with Raymond, Chan and Evelyn to decide the context passings and data passings in the MVP structure and found the most efficient way.

There are 2 major pull requests I made:

1. First significant pull request

https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/117/

In this pull request I established the structure for data accessing for our project. The data accessing is causing tons of problems to us after implementing Android GUI and thus this is a major improvement. I designed the method for the activity to pass contexts to use cases through passing parameters, decided to make the context static in each use case so they are more accessible, and determined to pass the contexts in the main activity class at the beginning of the program.

2. Second significant pull request

https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/81

In this pull request I implemented the entire structure of Android GUI for the functionality of manager, including managing menu, deleting reviews and seeing reviews. I designed the views of the activities and their interactions with each other.

Chan Yu

In phase 2, I am responsible for the implementation of Android GUI for the functionality of user login. I also implemented the new functionality for the manager to view all existing users, as well as enroll staff by setting name, password, and the user type for the new staff. Meanwhile, I fixed the issue that was pointed out by TA of lacking the output boundary between the use case and presenter. After the group made the design decision of using the MVP structure, I refactored the code to implement view interfaces that are implemented by Android activities. Additionally, I worked together with Raymond and Howard to explore the possibilities of implementing an online database to solve the previous issue of using project files and Android internal storage, and assisted Raymond to implement Google storage with the account credential process. Furthermore, I wrote tests for the functionalities that I have implemented and added tests to achieve the group goal. Also, I refactored the Android XML files to ensure the application views have a consistent outlook.

Significant pull requests:

1. https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/79

   In this pull request, I implemented the functionality of the manager enrolling new staff.
2. https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/94

   In this pull request, I refactored the functionalities of login and staff enrollment to implement MVP structure.

Dedong Xie

I am responsible for the implementation of staff system. I implemented the Android GUI for serving staff and delivery staff. I used MVP view model in the design: On the view level, there are serving staff pick option screen, which goes to serve dish screen. There are also deliver staff pick option screen, deliver order screen where delivery staff can get a new order, find route to an order’s destination. Implemented strategy design pattern, adaptor design pattern, private class data design pattern. Created tests for nearly all use case and some of the entities to increase test coverage.

Significant pull requests:

1. https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/80

   In this pull request, I implemented maps functionality as well as activity view for the staff users, using a combined input boundary for use cases of serveDish and deliverOrder, trying to implement the design pattern of strategy.

2. https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/110

   In this pull request, I implemented the private class data design pattern in entity level which prevents unintentional change to core attributes of every user. Plus, I implemented MVP view model with clear input/output boundary between presenter and use case. Which further make the program satisfied clean architecture.

Raymond Liu

In phase 2, I implemented the Android UI for Kitchen system, as well as the new google cloud storage database system to replace the old database we had (which is unusable in Android project due to its unique file structure). The google cloud storage database read and write from data files in the cloud storage by their public url. A service key is also required for data read write; it serves as the identity of our google cloud bucket as well as the authentication required for writing data through url. Every group member is given a copy of the key. In addition, I Implemented the MVP model, output boundary of the Kitchen system. The Kitchen Presenter now acts as a media between the view (the activities) and the business logic (the use cases), and allows its own data to be updated by the use case by implementing the output boundary. The observer design pattern is also implemented between PlaceOrderActivity and CurrentOrderDishActivity such that the Kitchen GUI refreshes its display when a new order comes in.

Significant pull request:

- https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/115

  In this pull request, I implemented the new google cloud database system, as well as the MVP model, output boundary, and the Observer design pattern for Kitchen.

  This pull request is significant because it resolves the long hanging database issue that we have encountered after migrating to Android. As well, it allows all the activities that require data read/write to no longer hard code their data initialization to test the functionalities. Moreover, it opens the possibility of making our program multi-devices, as now any changes to the data (e.g. modifying the quantity of an inventory item) are uploaded to the cloud, and can be accessed by another device by reading the updated file.

Mingyang Li

In Phase 2, I created the Android GUI for users to provide a review and rate their experience. The review may be anonymous. It follows the MVP structure for the android activities by directly instructs what users need to implement the view interface. Updated unit tests and fix error found in entity Review.

Significant pull request:

1. https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/120

   Updated unit tests and fix error found in entity Review

2. https://github.com/CSC207-UofT/course-project-group-002-tut5101/pull/76

   This pull request is Android GUI for AddReivew.