Phase 0 Progress Report

I. Domain

We choose our theme to be university student open discussion forum, allowing students to interact with each other based on the courses they enrolled.

II. Specification

Our program provides a service for students to discuss course content, for faculty to post about course and program information, and for professors to post and delete comments. All three types of users should go through the registration process to access this program. All users will need a name, user id and password.

Students and professors can enroll themselves into existing courses, while professors will be added to the course page after the enrollment. Students who enrolled in the same courses can post to the current course discussion board, reply to others' comments, and delete their own comments. They also have access to the past post pages to learn from past students who previously enrolled in the course. They can also see other courses taught by their professors to learn about them.

Inside the course page, students can see course name, course information, professor information, current discussion board and the discussion boards in the past three semesters. Professor and faculty can see the same course page, both with ability to delete comments. Faculty can create and update course pages, while professors have the right to reply to the comments and make announcements.

III. CRC model

Entity classes: User, Comment, Course

Use case classes: Student, Professor, Faculty etc.

Controller: LogController etc.

UI: CommandLineUI etc.

Single responsibility:
Student: all methods delete comment, add comment, enroll in courses are used through implementing different interfaces, and are unrelated to each other.

Open-closed principles & Liskov Substitution Principle:

Any new subtypes of User will still have instances user_name, user_id and password. For example, if we need a new subclass TA, they will still have these three instance variables, and possibly some new instance variable, which is open for extension and will not cause any error.

Interface Segregation Principle:
Our interfaces all serve one basic function. For example, our registration interface only have one method register().

Dependency Inversion Principle:
CommandLineUI <- Inputboundary -> Student/ Professor/ Faculty
We use the input boundary to avoid Upper level use cases to take in values directly from the CommandLineUI, which adheres to this principle.

IV. Scenario Walkthrough

To register as a student, we type in a string student to claim it's a student account, then we will type in whether it's a new student account. (yes for new student and no for already existed account). After that, we will enter student name, student id, and account's password in order. Then, the **LoginInputInterface** act as controller package all the data into string and int objects and passes these objects to the **Signupandlogin**. If the student indicates that it's a new account, then, we will build a new student and stored them to the attribute of **AllStudent** class(. This is a class store the all the students' information implementing the **student** class. **Student** class is a subclass of entities **User**), then we will present "successfully signup" at **UI**. If the student indicates that it's a not new account, we will check them in **LoginUser** by compare weather the input data matches the information stored in **AllStudent.** If it matches, present "success login" at **UI.** If doesn't present "login in fails, please try again".

V. Skeleton Programs

Demonstration

VI. Labor division

Meetings (together): pick domain, decide software architecture and relationship between layers; create CRC cards.

- Junhao Saw: project specifications
- Kuang Jiang: wrote the scenario walkthrough section, and draw the layers of architecture
- Hanqi Zhang: wrote basic code for skeleton program, the prospect of program, and revised and organized the progress report

- Zhijun Wang: elaborated on the code for the skeleton program, implementing all methods needed for scenario walkthrough

- Zeyuan Li: wrote the CRC model, clean architecture and SOLID principle section

- Chuanyang Qiao: made PPT, wrote the things we do well, the things we struggle, and open questions.

VII. Prospect

- Implement the core part of Comment class and use breath recursion to delete comment
- Write Unit tests for all basic functions and reach a certain coverage
- Decide the way of packaging and make decision for the access modifiers
- Learn about Andriod Studio IDE and may implement it in future

VIII. Open questions

- Are we able to use the Andriod Studio IDE to make an app?
- How should we revise our structure to make it adhere to the clean architecture and SOLID principles more?