SEG2105 ANDROID PROJECT - GROUP 64 FINAL REPORT

SEG 2105 | B00
PROFESSOR M. GARZON

Written by

Benjamin Sam | 300298874
Olena Naim | 300322912
Sully Montgomery | 300326826
Carl Cheng | 300309128
Markus Gandia | 300303155

2023/12/06

TABLE OF CONTENTS

INTRODUCTION

The app ‘Cycling64’ is an online platform for cycling enthusiasts to congregate, schedule, and participate in cycling events. Created in Android Studio, the app features three roles that can be adopted by users of the app: participant, club, and admin. Admin accounts are allowed to manage all the functionalities on the app, with the ability to delete the types of events that can be offered by clubs and the club accounts themselves. Club accounts are publicly available and can schedule events with various criteria, including difficulty, length, and location. Participants are the second publicly available role and can search for clubs, event types, or scheduled events. Furthermore, the app employs Firebase due to its use in mobile applications and Cloud storage as opposed to SQLite’s storage on a local device.

PROJECT DESIGN

The project was divided into four deliverables. The first saw the setup of the UML class diagram, the creation of the GitHub repository, and the sign-in feature for club and participant accounts. The second implemented admin-level functionality, while the third and fourth introduced features for the club and participant accounts, respectively.

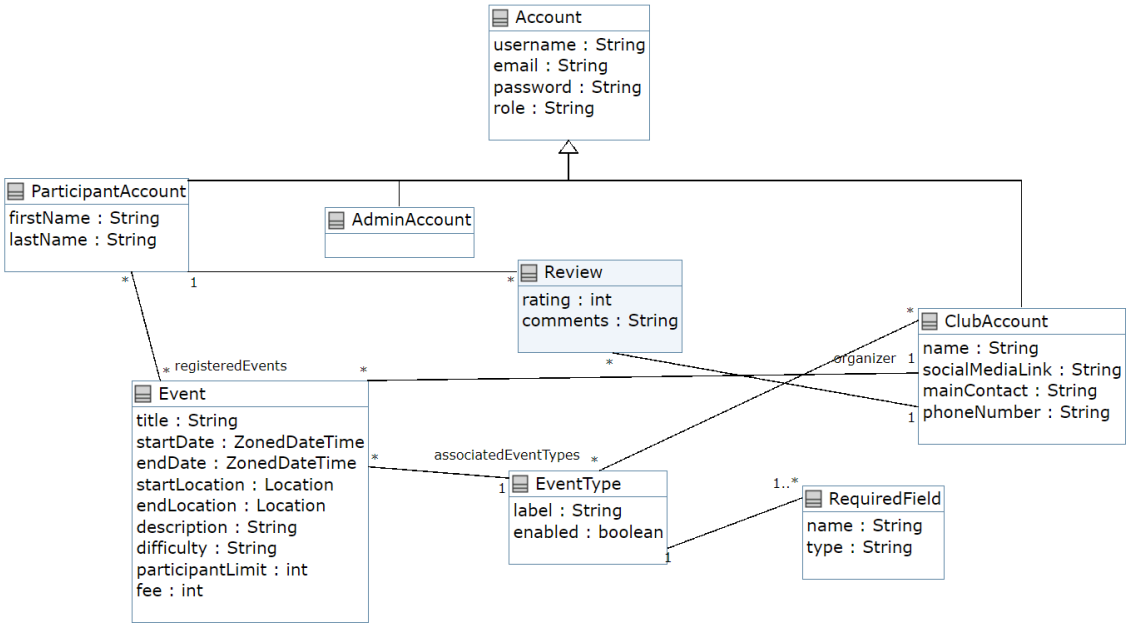

Figure 1: UML class diagram of Cycling64.

Figure 1 shows the UML class diagram of Cycling64. The heavyweight classes are the three Account subclasses: AdminAccount, ClubAccount, and ParticipantAccount. EventTypes act as a

blueprint for the kind of Event that clubs can schedule. As such, ClubAccounts have two separate data structures for storing lists of both of these classes, as clubs can host real events (instances of class Event) and store the kinds of events their club participates in (instances of class EventType).

The differentiating factor from the three subclasses of Account is the methods. Each shares login information, such as username and password, and ClubAccount and ParticipantAccount share name and surname variables. Despite these similarities, as different roles, the three have distinct methods and operations.

TABLE OF CONTRIBUTIONS

GROUP MEMBER	**DELIVERABLE 1**
Sam, Benjamin	Helped to implement basic signup/login functionalities. Setup Firebase DB. Helped with UML diagram.
Naim, Olena	Updated the UML Diagram, so it complied with the deliverable 1 class's fields. Implemented the following classes: Admin, Participant, Role(Enum), User. Tested realTime database (Firebase Auth), which was replaced by Firestore Firebase; implementing sign-up and log-in classes using FirebaseAuth.
Gandia, Markus	Helped create the UML diagram, establishing the basis of the project Created unused concept code (for the SignIn feature).
Cheng, Carl	Implemented field validators for LoginActivity and SignupActivity, including

GROUP MEMBER	DELIVERABLE 1
Sam, Benjamin	Helped to implement basic signup/login functionalities. Setup Firebase DB. Helped with UML diagram.
	fields for name, emails, and passwords
Montgomery, Sully	Helped implement the Event, EventType, Account, AdminAccount, ClubAccount, and ParticipantAccount classes.

GROUP MEMBER	DELIVERABLE 2
Sam, Benjamin	Helped with admin side event type related functionality and account viewing (for deletion, etc.). Helped with UML. Helped implement a sidebar for the app, fragments, etc.
Naim, Olena	Tried implementing admin-related functionalities, which weren't pushed/ used
Gandia, Markus	UML updating and editing
Cheng, Carl	Transitioned from individual validation methods now into using a validation field utility class to reuse the same code across all the fields to be validated. The fields were now validated in regards to the datatype (either int or float numbers) and can be specified to match a custom regex.
Montgomery, Sully	General code editing

GROUP MEMBER	DELIVERABLE 3
Sam, Benjamin	Helped merge code for different features together, worked on improving the event creation activities, etc. Helped with UML.
Naim, Olena	Implemented with EventActivity.java and activity_event.xml, the module uses TextViews, a Spinner with adapter, RadioGroup, and Button for user interaction. Users(Admin/Club) can conveniently fill in the required information such as event name, location (zip code), description, participant limit, and registration fees. Notably, date and time are entered separately as Strings but ultimately saved as a combined "zonedTimeDate" for efficient storage and retrieval. Depending on the selected event type, additional fields are dynamically displayed and labeled. EventActivity.java serves as the core logic behind the XML file, implementing the onCreate method and enabling functionalities like saving, retrieving, deleting, and updating event data within the database.
Gandia, Markus	Provided unused concept code (ProfileActivity). UML updating and ensuring cohesion with the added deliverable functionalities and features.
Cheng, Carl	Tried to implement a navigation bar for clubs, but ultimately not used.
Montgomery, Sully	Developed the Java and XML files for the Event Fragment which allows a club account to view/add/edit their events. Developed the EventList.java file which searches the database for all events that have their organizer field equal to the username of the signed-in club account. Gives the user the option to create a new event or edit an existing event by either pressing the “Add an event” button or clicking one of the listed events.

	Implemented the separation of tasks throughout the app. The navigation menu and default page that opens after logging in changes depending on the type of account currently logged into the app. Implemented the ability to send data through fragments using bundles such as information about the club account to find their events in the database.

GROUP MEMBER	**DELIVERABLE 4**
Sam, Benjamin	Finished fixup of event creation activities for clubs. Worked on search related functionality on the participant side. Worked on participant side event signup and club account review features. Helped with UML.
Naim, Olena	Updated the final UML diagram
Gandia, Markus	Implemented search functionality UI and design onto MainActivity. UML updating and editing, including new functionalities added in Deliverable 4 Final report writing
Cheng, Carl	Implemented unit test cases using Junit and tested the field validations for the Signup Activity, 4 of the test cases tested for an invalid validation for the field. Implemented unit test cases for EventActivity regarding the field validation for future or past dates and times.
Montgomery, Sully	Implemented the search function for participants to find clubs, events, and event types that included the searched phrase. This involved searching through the database for clubs/events/event types corresponding to the search. I combined all results from the database and presented them in a list dynamically as the

	user types in the search bar

APP SCREENSHOTS

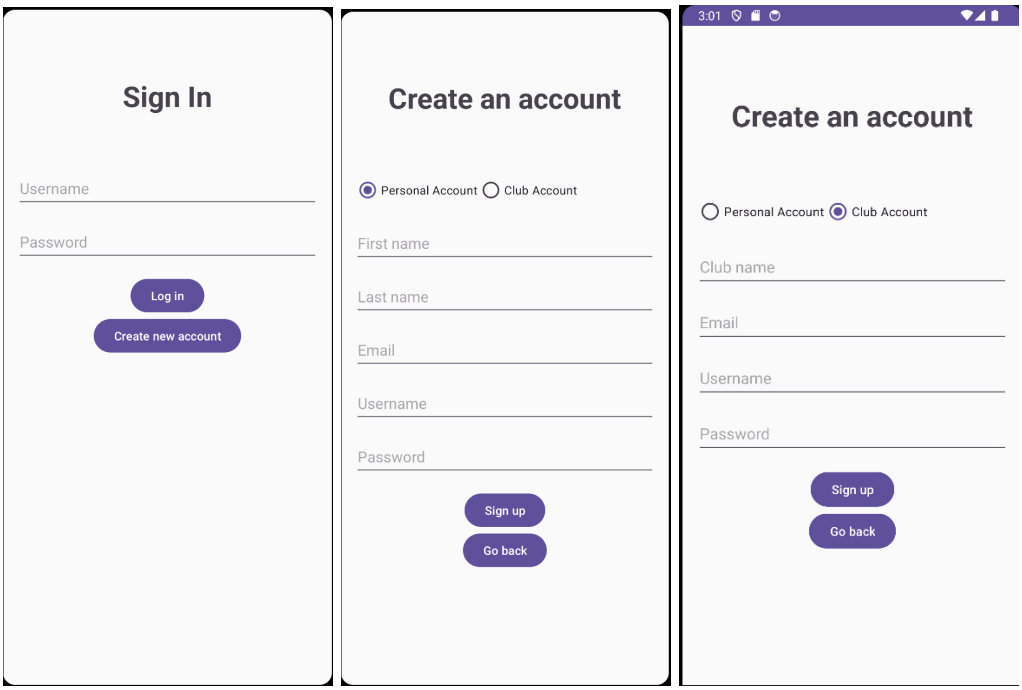

Figure 2: Login and signup pages.

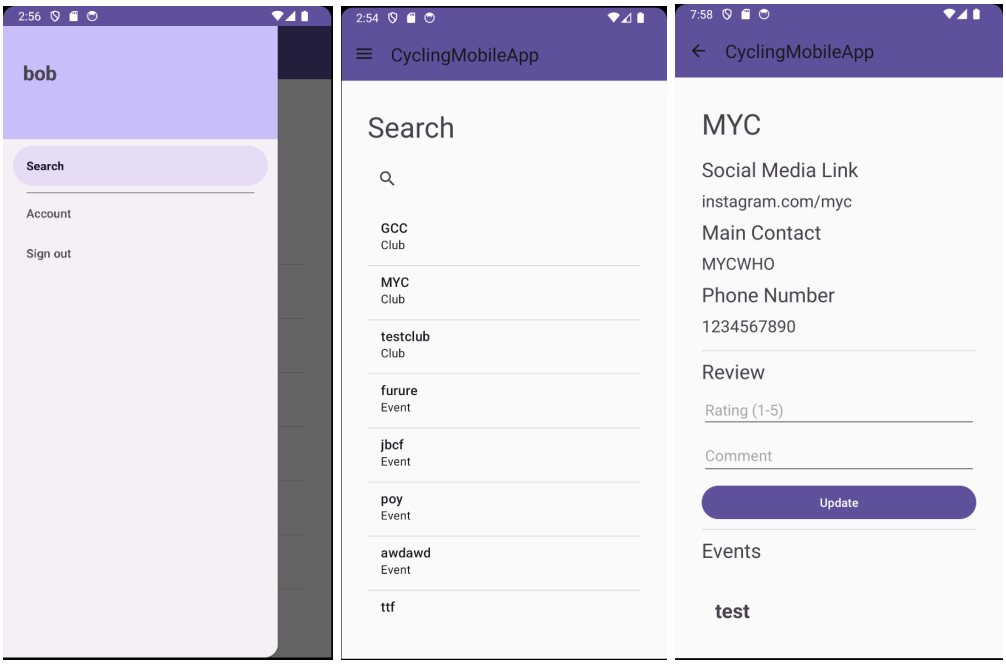

Figure 3: ParticipantAccount view, search activity, club.

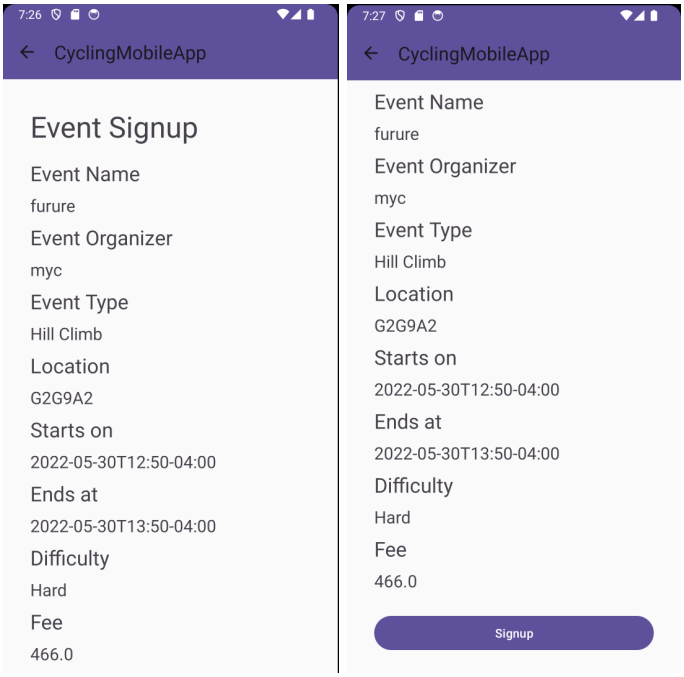

Figure 4: Event signup.

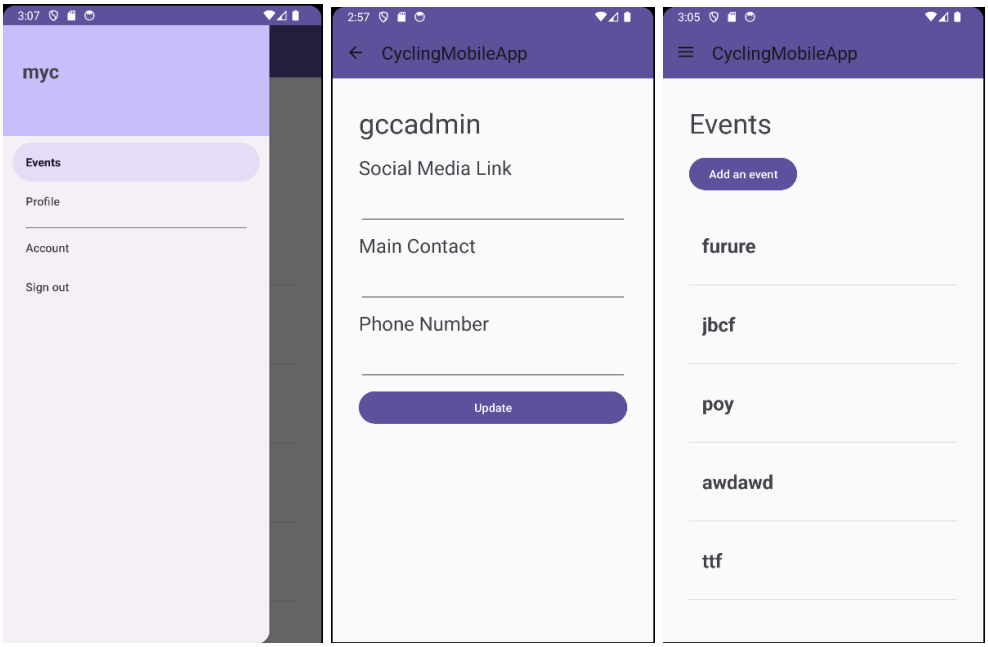

Figure 5: ClubAccount view, profile, and event activities.

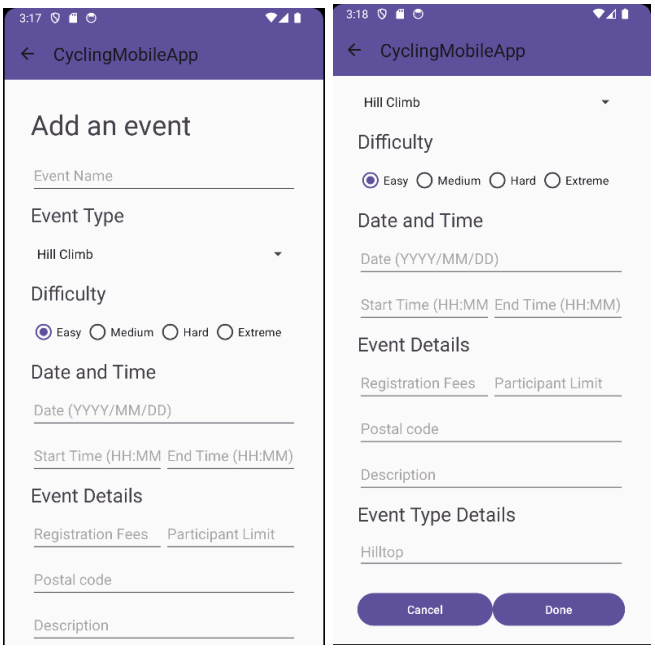

Figure 6: Event creation.

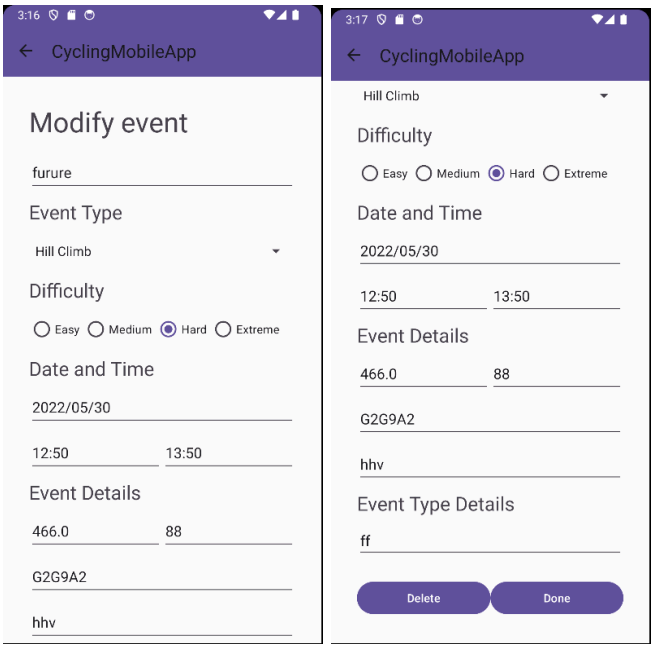

Figure 7: Event modification.

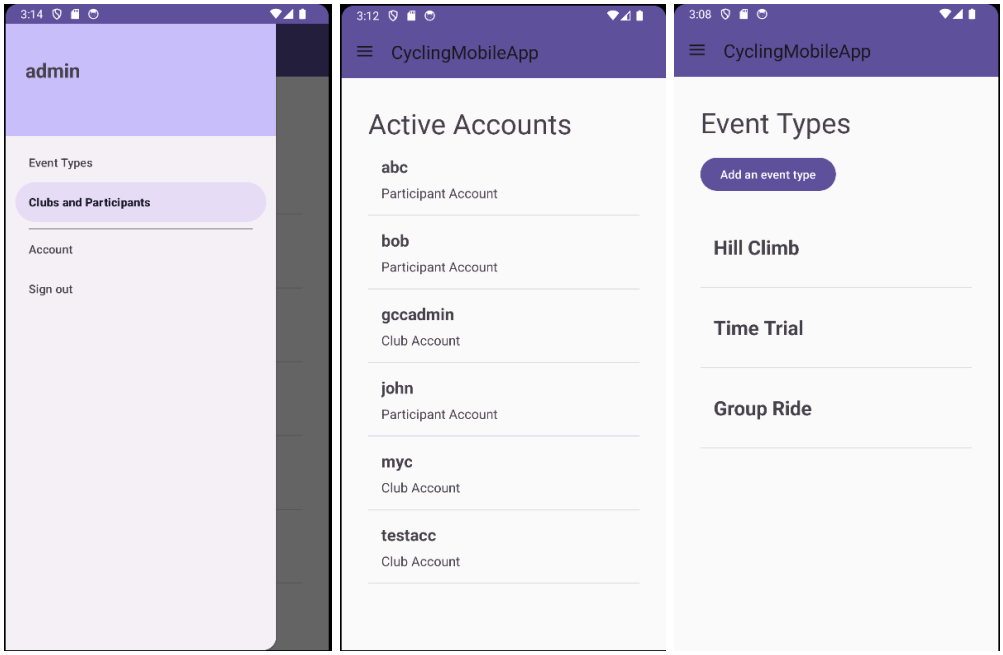

Figure 8: AdminAccount viewing clubs, participants and event types.

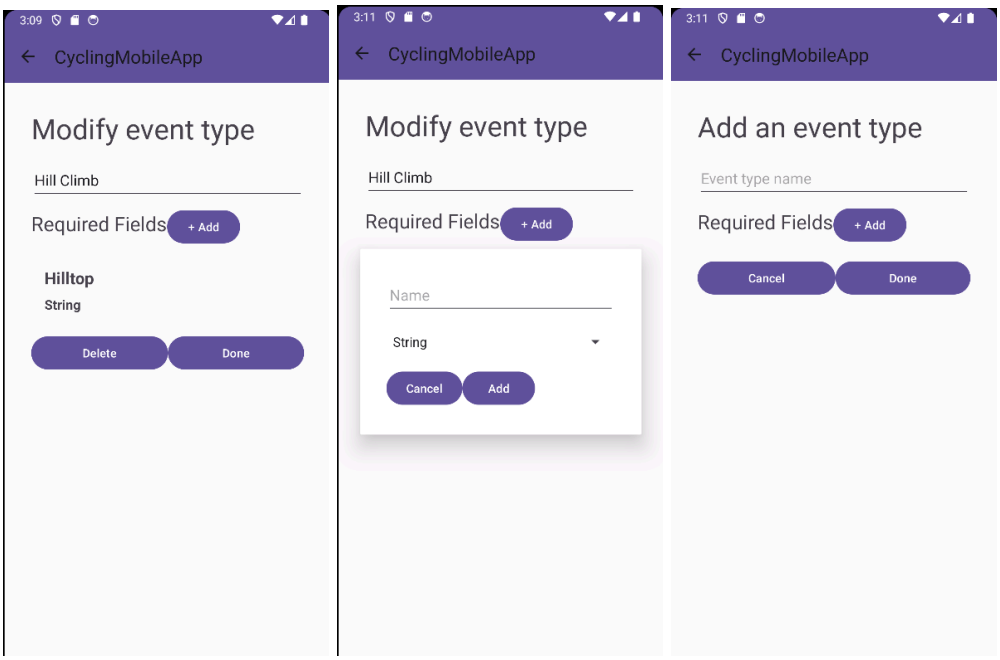

Figure 9: Event type creation and modification.

LESSONS LEARNED AND SUGGESTIONS

As part of our ongoing project-based learning journey, we explored Android Development, using Android Studio to create a mobile application(Grimpeurs Cycling Club (GCC) App). We applied key concepts of what we learned in class and labs of our software engineering course like object-oriented programming, design patterns, implementation of testing methods, and adopted version control to keep everyone updated. This approach ensured the development of a maintainable application

We also learned “Database Management”. We not only acquired the skills to design and implement databases but also leveraged this knowledge to store, retrieve, and modify information about events, users, and results, utilizing the Firestore Firebase database as our primary platform.

We had to “identify requirements” where we analyzed project specifications and user needs. This critical step provided the essential foundation for designing applications that fulfilled all functionalities specified within the project scope.

Finally, we also gained valuable exposure to create an intuitive and user-centered interface. While also placing a particular emphasis on adhering to Android Design Guidelines.

Much of the deliverable's work was repetitive and, after four deliverables, became tedious. The use of external software or improving reusability in code is very important to reduce this frustration. This problem was most apparent with the reuse of a lot of methods, such as findViewById() or the process behind retrieving and storing with the database. With a large variety of modern software that can be used to design UI, the use of Android Studio's XML code option felt inefficient.

Communication was vital throughout the project. Initially, a lot of work was compartmentalized in its allocation, with each member individually performing their own given tasks. However, this was prone to confusion and an incohesive workflow in the group, as there was no cooperation and some components were dependent on external ones.
Some examples of clearer communication are frequent use of group chats, updating team members on work status, cooperative working habits, and commenting/documenting the code.

Appendix

UML Diagram:
https://cruise.umple.org/umpleonline/umple.php?model=231103ra3exen69hbf#genArea

Github link:
https://github.com/SEG2105BC-uOttawa/seg2105f23-project-project_grp_64.git