Project Report: Fitbits

SEG 2105: Introduction to Software Engineering

Fall 2021

School of Electrical Engineering and Computer Science

University of Ottawa

Professors name:

Dr. Omar Badreddin

Group #1
Student Name and Number: Ritvik Johar, 300074686
Student Name and Number: Alexandra Tulchinsky, 300165864
Student Name and Number: Kevin Le, 300053360
Student Name and Number: Jacob Kyle Dulce, 300169731
Student Name and Number: Jason Su, 300074626

Table of Contents

List of Figures

List of Tables

1.0 Introduction

1.1 Purpose

The purpose of this project is to get familiarized with practical applications of android application development using Android Studios, Firebase, and Git. In this project, we designed and implemented fitness class booking application for android devices. Software design specifications were followed such as UML and unit testing. The application has three different types of users i.e., the administrator, the instructor and the gym member. All the components of the application were programmed using Java.

2.0 User functionalities

The application has three different types of users i.e., the administrator, the instructor and the gym member. The administrator manages all the possible classes offered to the members at the fitness centre. The administrator also manages the gym instructor and member accounts. Instructor are able to select the class they would like to teach and specify the class date, duration, capacity and difficulty. Gym members are able to search for a drop-in class by class name or day of the week. They can also enrol and unenroll to a class.

2.1 Admin

- Login to the administrator account.
- Create and delete classes offered by the fitness centre.
- Edit the class name and description.
- Delete accounts of instructor and gym members.

2.2 Instructor

- Create an account and login to that.
- Choose a class to teach by selecting the class type created by admin.
- Add and edit class day, duration, capacity and difficulty.
- Cancel a class.

2.3 Gym Member

- Create an account and login to that.
- View all available classes and enrolled classes.
- Search for a class by class name or day of the week.
- Enrol unenroll in a class.

3.0 Deliverables

3.1 Deliverable 1

For deliverable 1 of the project, we implemented and designed user account management and admin functionality. For simplification, the admin account was pre-created whereas there could be many instructors and gym members accounts. All the users' accounts are stored using the database Firebase.

The application allows users to create user accounts via the sign-up page. And once the user logins it takes them to the second page where they see the following message: "Welcome 'firstname/username'! You are logged in as 'role'. "

The admin functionality includes creating, editing, and deleting types of classes. The admin can create a class type by entering the class name and description. The admin can also edit the class name and description of pre-existing class type. The instructor and gym members can create and log in to their accounts. The instructors and gym members can view the class types. The admin can also delete the accounts of instructors and gym members.

3.2 Deliverable 2

For deliverable 2 of the project, we implemented the instructor functionality. This includes viewing all scheduled fitness classes, scheduling a class, and editing their existing classes. The instructor can view a list of all scheduled classes. The classes can also be searched by using a class name or instructor name. The instructor can choose a class to teach and edit class difficulty, day of the week, duration, and the maximum capacity of the class. The instructor can also cancel a class that removes the class and all specified details. The system prevents scheduling the same class on the same day and also prevents the instructor from scheduling a class that is already scheduled by another instructor on the same day. We also created unit tests to make sure everything is working properly.

3.3 Deliverable 3

For deliverable 3 of the project, we implemented the functionality of the gym member and also finalized the entire application.

The gym member functionality includes viewing all the scheduled classes that are offered as well as searching using class names and day of the week. The member can enroll in a fitness class and if the fitness class conflicts with another class an error message is displayed. The members can also view all the classes they have enrolled in. And can unenroll from a class.

4.0 Use Case Model

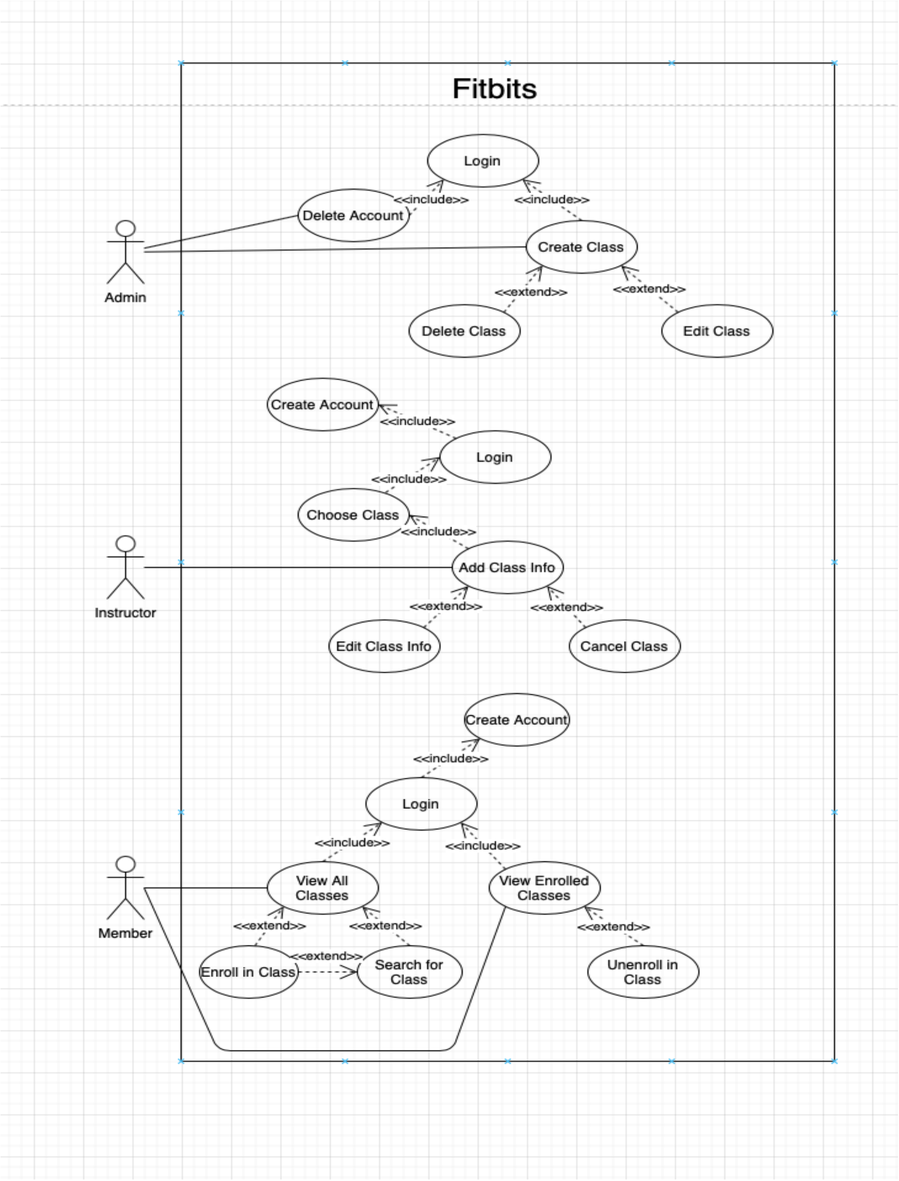


Figure 1: Use Case Model

5.0 UML Class Diagram

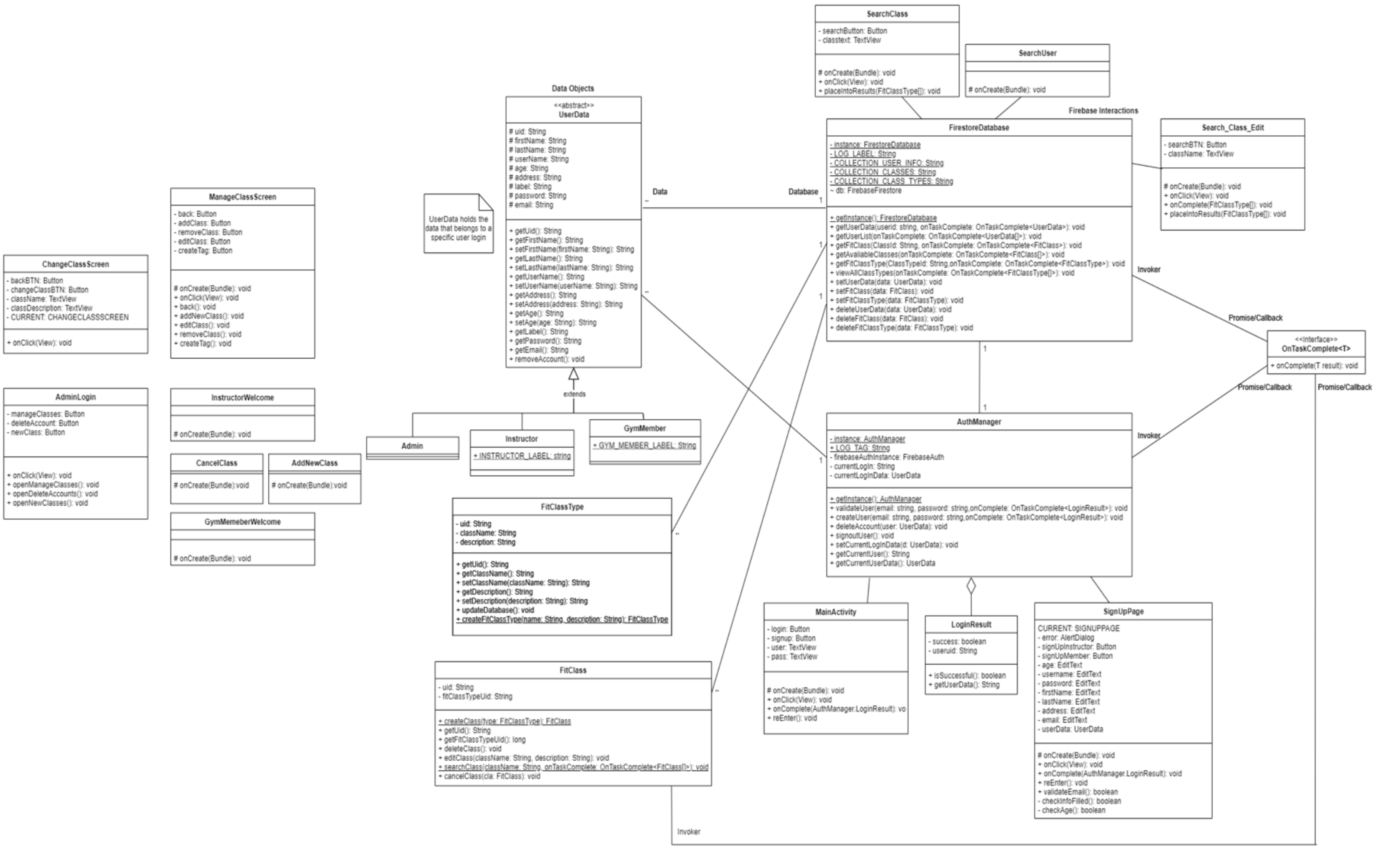

Data Objects
<<abstract>>
UserData
# uid: String
# firstName: String
# lastName: String
# userName: String
# age: String
# address: String
# label: String
# password: String
# email: String
+ getUid(): String
+ getFirstName(): String
+ setFirstName(firstName: String): String
+ getLastName(): String
+ setLastName(lastName: String): String
+ getUserName(): String
+ setUserName(userName: String): String
+ getAddress(): String
+ setAddress(address: String): String
+ getAge(): String
+ setAge(age: String): String
+ getLabel(): String
+ getPassword(): String
+ getEmail(): String
+ removeAccount(): void
UserData holds the data that belongs to a specific user login
extends
Admin
Instructor
+ INSTRUCTOR_LABEL: string
GymMember
+ GYM_MEMBER_LABEL: String
ManageClassScreen
- back: Button
- addClass: Button
- removeClass: Button
- editClass: Button
- createTag: Button
# onCreate(Bundle): void
+ onClick(View): void
+ back(): void
+ addNewClass(): void
+ editClass(): void
+ removeClass(): void
+ createTag(): void
ChangeClassScreen
- backBTN: Button
- changeClassBTN: Button
- className: TextView
- classDescription: TextView
- CURRENT: CHANGECLASSSCREEN
+ onClick(View): void
AdminLogin
- manageClasses: Button
- deleteAccount: Button
- newClass: Button
+ onClick(View): void
+ openManageClasses(): void
+ openDeleteAccounts(): void
+ openNewClasses(): void
InstructorWelcome
# onCreate(Bundle): void
CancelClass
# onCreate(Bundle):void
AddNewClass
# onCreate(Bundle):void
GymMemeberWelcome
# onCreate(Bundle): void
FitClassType
- uid: String
- className: String
- description: String
+ getUid(): String
+ getClassName(): String
+ setClassName(className: String): String
+ getDescription(): String
+ setDescription(description: String): String
+ updateDatabase(): void
+ createFitClassType(name: String, description: String): FitClassType
FitClass
- uid: String
- fitClassTypeUid: String
+ createClass(type: FitClassType): FitClass
+ getUid(): String
+ getFitClassTypeUid(): long
+ deleteClass(): void
+ editClass(className: String, description: String): void
+ searchClass(className: String, onTaskComplete: OnTaskComplete<FitClass[]>): void
+ cancelClass(cla: FitClass): void
Invoker
Data
Database
1
SearchClass
- searchButton: Button
- classtext: TextView
# onCreate(Bundle): void
+ onClick(View): void
+ placeIntoResults(FitClassType[]): void
SearchUser
# onCreate(Bundle): void
Firebase Interactions
FirestoreDatabase
- instance: FirestoreDatabase
- LOG_LABEL: String
- COLLECTION_USER_INFO: String
- COLLECTION_CLASSES: String
- COLLECTION_CLASS_TYPES: String
~ db: FirebaseFirestore
+ getInstance(): FirestoreDatabase
+ getUserData(userid: string, onTaskComplete: OnTaskComplete<UserData>): void
+ getUserList(onTaskComplete: OnTaskComplete<UserData[]>): void
+ getFitClass(ClassId: String, onTaskComplete: OnTaskComplete<FitClass>): void
+ getAvaliableClasses(onTaskComplete: OnTaskComplete<FitClass[]>): void
+ getFitClassType(ClassTypeId: String,onTaskComplete: OnTaskComplete<FitClassType>): void
+ viewAllClassTypes(onTaskComplete: OnTaskComplete<FitClassType[]>): void
+ setUserData(data: UserData): void
+ setFitClass(data: FitClass): void
+ setFitClassType(data: FitClassType): void
+ deleteUserData(data: UserData): void
+ deleteFitClass(data: FitClass): void
+ deleteFitClassType(data: FitClassType): void
Search_Class_Edit
- searchBTN: Button
- className: TextView
# onCreate(Bundle): void
+ onClick(View): void
+ onComplete(FitClassType[]): void
+ placeIntoResults(FitClassType[]): void
Invoker
Promise/Callback
<<Interface>>
OnTaskComplete<T>
+ onComplete(T result): void
Promise/Callback
Promise/Callback
Invoker
AuthManager
- instance: AuthManager
+ LOG_TAG: String
- firebaseAuthInstance: FirebaseAuth
- currentLogIn: String
- currentLogInData: UserData
+ getInstance(): AuthManager
+ validateUser(email: string, password: string,onComplete: OnTaskComplete<LoginResult>): void
+ createUser(email: string, password: string,onComplete: OnTaskComplete<LoginResult>): void
+ deleteAccount(user: UserData): void
+ signoutUser(): void
+ setCurrentLogInData(d: UserData): void
+ getCurrentUser(): String
+ getCurrentUserData(): UserData
MainActivity
- login: Button
- signup: Button
- user: TextView
- pass: TextView
# onCreate(Bundle): void
+ onClick(View): void
+ onComplete(AuthManager.LoginResult): vo
+ reEnter(): void
LoginResult
- success: boolean
- useruid: String
+ isSuccessful(): boolean
+ getUserData(): String
SignUpPage
CURRENT: SIGNUPPAGE
- error: AlertDialog
- signUpInstructor: Button
- signUpMember: Button
- age: EditText
- username: EditText
- password: EditText
- firstName: EditText
- lastName: EditText
- address: EditText
- email: EditText
- userData: UserData
# onCreate(Bundle): void
+ onClick(View): void
+ onComplete(AuthManager.LoginResult): void
+ reEnter(): void
+ validateEmail(): boolean
- checkInfoFilled(): boolean
- checkAge(): boolean

Figure 2: UML Class Diagram

6.0 Work Distribution

6.1 Deliverable 1

Task	**Group Member**
Welcome admin screen	Alexandra Tulchinsky
Option screen	Kevin Le
Add new class screen	Alexandra Tulchinsky
Search user screen	Kevin Le
Welcome instructor screen	Kyle Jacob Dulce
Edit class screen	Ritvik Johar
Search class screen	Kevin Le
Confirm popup	Kyle Jacob Dulce
Status popup	Kyle Jacob Dulce
Add class screen	Kevin Le
Firebase authorization middleware	Kyle Jacob Dulce
Cancel class screen	Ritvik Johar
Setup firebase	Kyle Jacob Dulce
Edit class screen	Alexandra Tulchinsky
Fitclass class implementation	Ritvik Johar
Implement users classes	Ritvik Johar
Login layout	Jason Su
Signup screen	Alexandra Tulchinsky
Signup backend	Alexandra Tulchinsky

Welcome admin screen backend	Ritvik Johar
Option screen backend	Ritvik Johar
Search user class screen backend	Ritvik Johar
Add new class screen backend	Ritvik Johar
Edit class admin	Alexandra Tulchinsky
Search class for user	Alexandra Tulchinsky
Search class	Kevin Le
Login backend	Alexandra Tulchinsky
Class diagram	Jason Su
Welcome gym member screen	Kyle Jacob Dulce
Finish refactor	Kyle Jacob Dulce

Table 1: Task allocation for deliverable 1

6.2 Deliverable 2

Task	**Group Member**
Search scheduled class layout	Jason Su
Search scheduled class backend	Jason Su
Fitclass Implementation	Kevin Le
Schedule class layout	Ritvik Johar
Schedule class backend	Ritvik Johar
Instructor cancel class layout	Alexandra Tulchinsky
Instructor cancel class backend	Alexandra Tulchinsky

Unit tests	Kyle Jacob Dulce
Update class Diagram	Kevin Le
Add class screen	Kevin Le

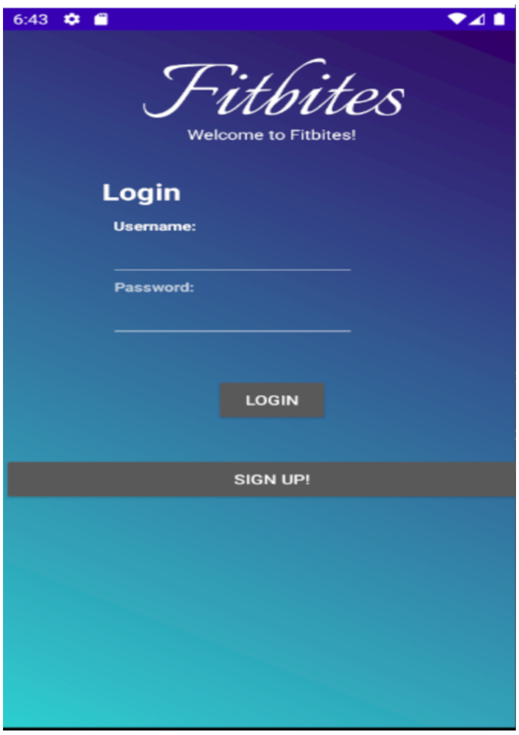
Table 2: Task allocation for deliverable 2

6.3 Deliverable 3

Task	**Group Member**
Member enroll page	Alexandra Tulchinsky
Edit gym member class	Kyle Jacob Dulce
Member enrolled page	Jason Su
Unit test	Kyle Jacob Dulce
UML diagram update	Kevin Le
Final Report	Ritvik Johar

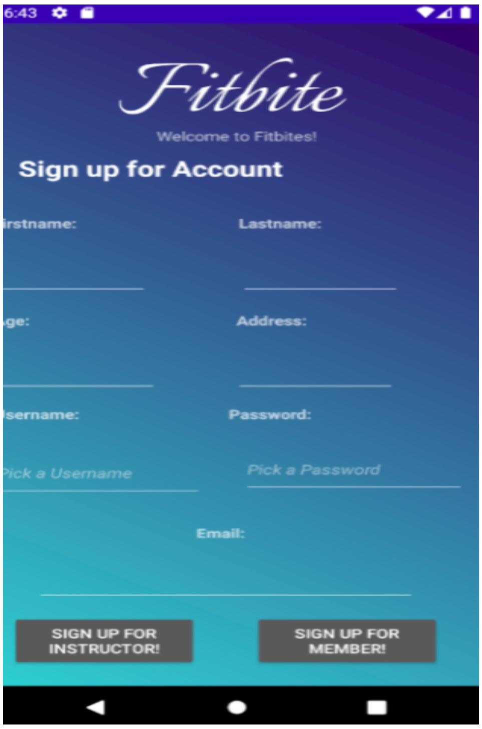
Table 3: Task allocation for deliverable 3

7.0 Screenshots



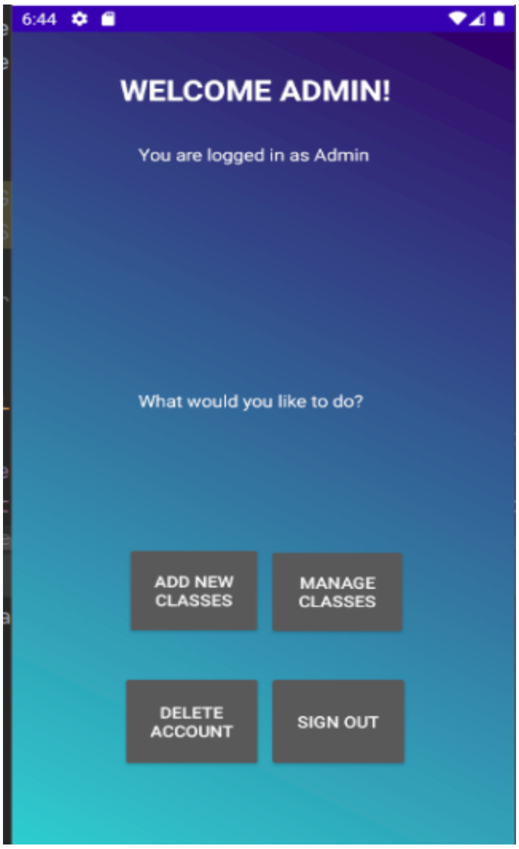
Welcome to Fitbites!

Figure 3: Login Page



Welcome to Fitbites!

Figure 4: Signup Page

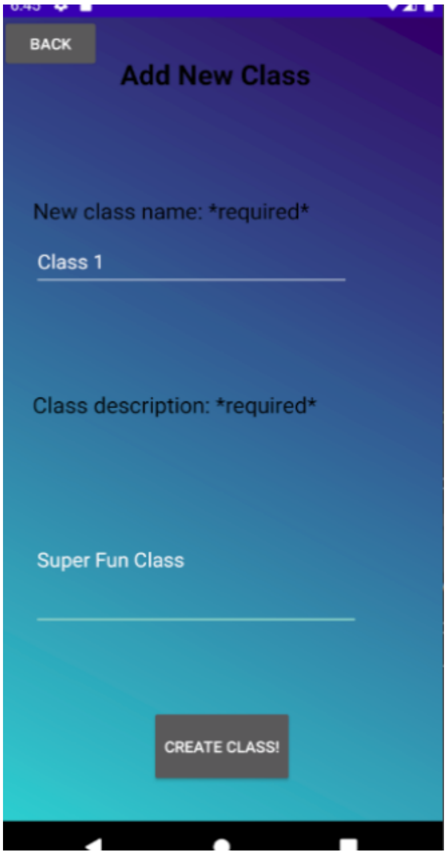


WELCOME ADMIN!

You are logged in as Admin

What would you like to do?

Figure 5: Admin Login



New class name: *required*

Class 1

Class description: *required*

Super Fun Class

Figure 6: Admin add new class Page

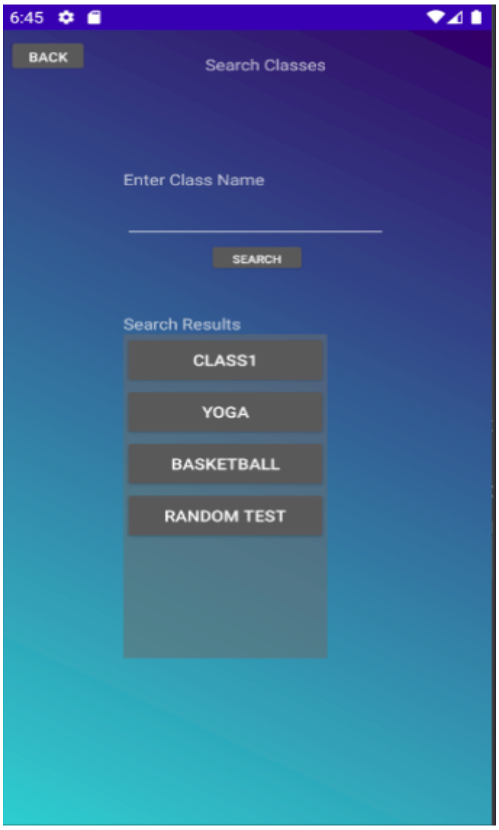
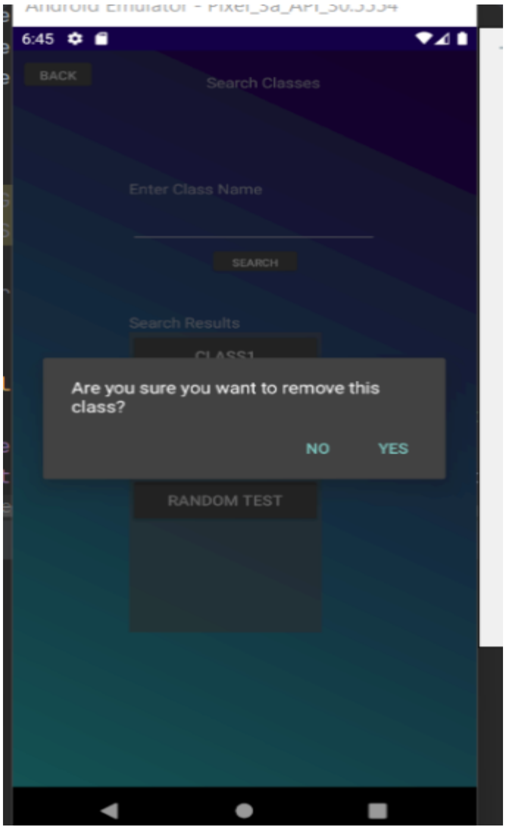
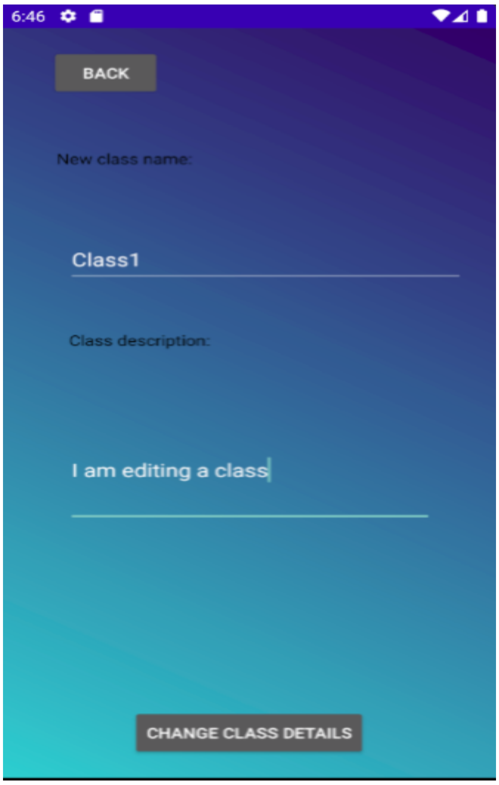


Figure 7: Admin search class Page



Are you sure you want to remove this class?

Figure 8: Admin delete class Page



New class name:

Class1

Class description:

I am editing a class

Figure 9: Admin edit class Page

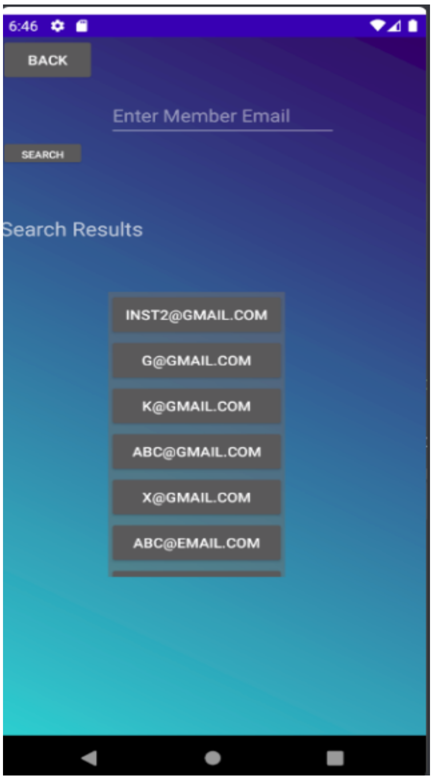
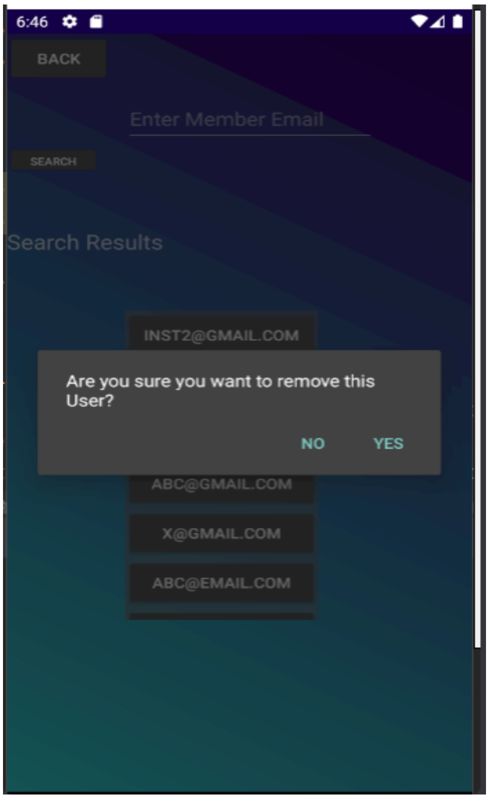


Figure 10: Admin search user Page

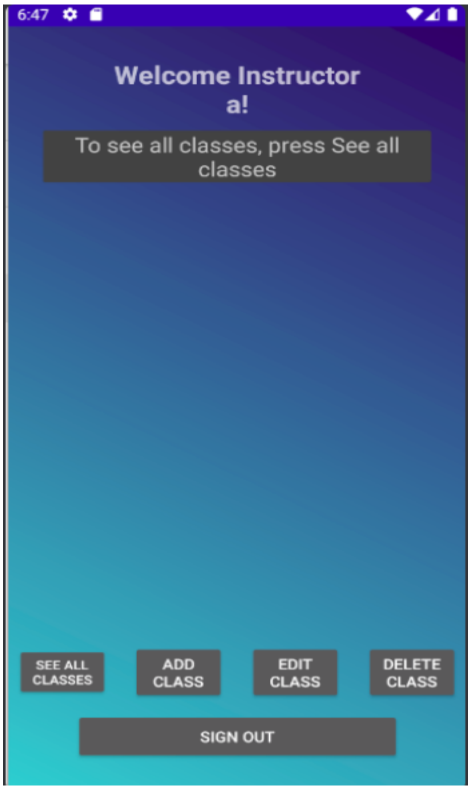


Enter Member Email

Search Results

Are you sure you want to remove this User?

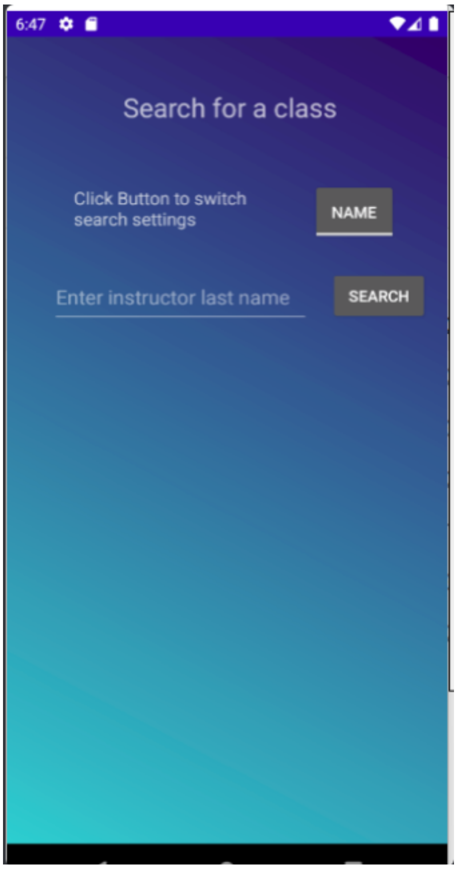
Figure 11: Admin delete user Page



Welcome Instructor a!

To see all classes, press See all classes

Figure 12: Instructor Welcome Page

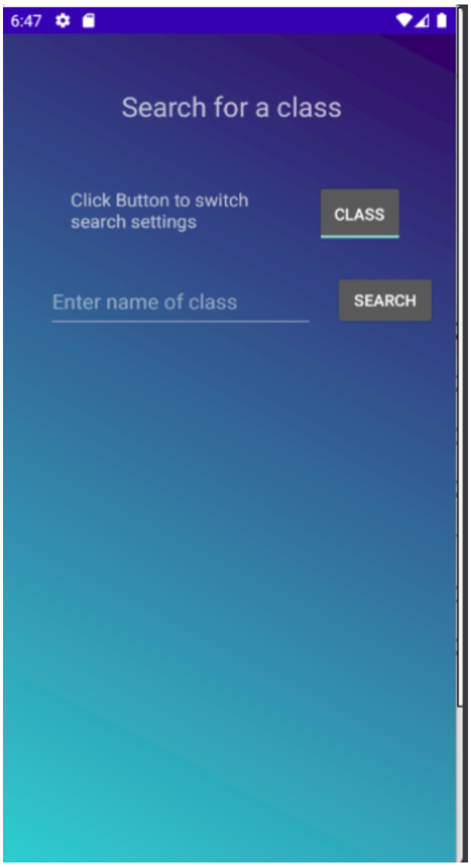


Search for a class

Click Button to switch search settings

Enter instructor last name

Figure 13: Instructor search class using last name Page



Search for a class

Click Button to switch search settings

Enter name of class

Figure 14: Instructor search class using class name Page

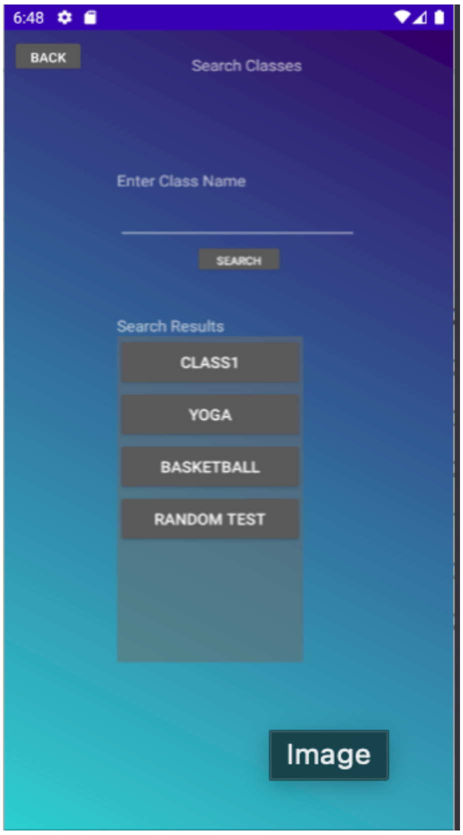
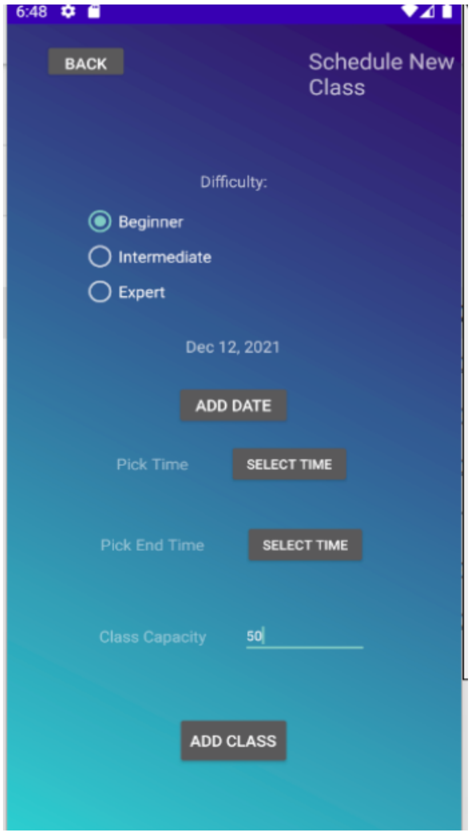


Figure 15: Instructor search list Page



● Beginner
○ Intermediate
○ Expert

Dec 12, 2021

Class Capacity 50

Figure 16: Instructor schedule class Page

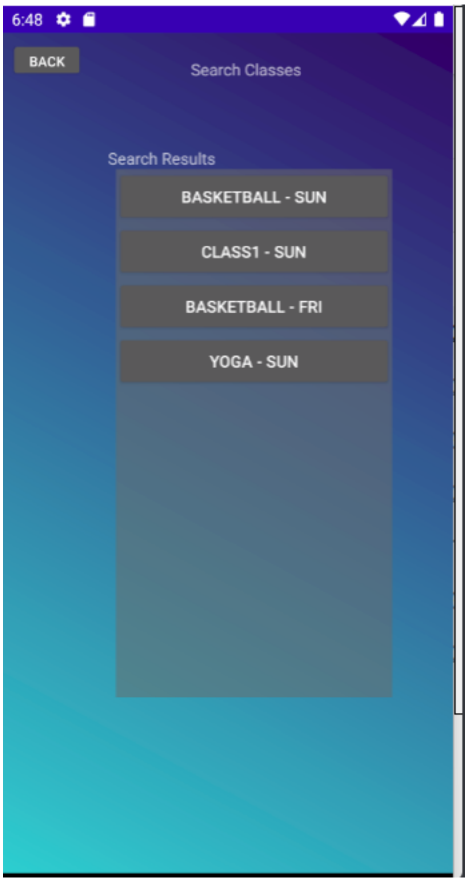


Figure 17: Instructor scheduled list Page

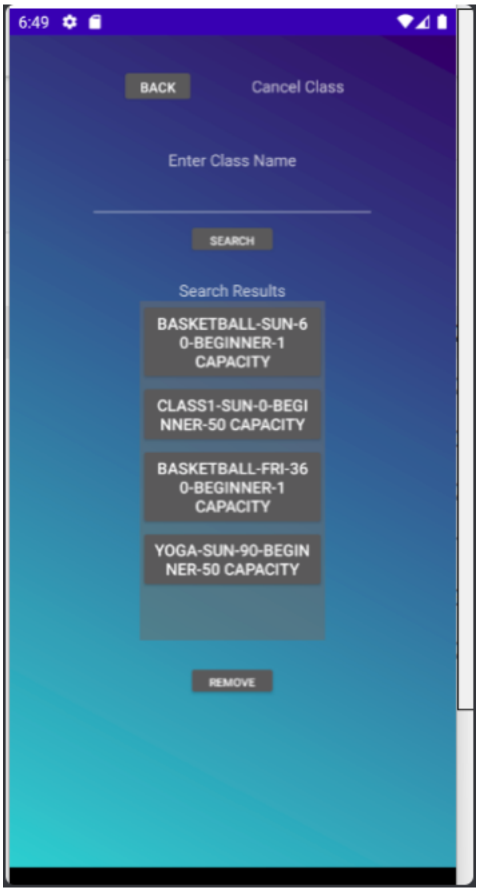
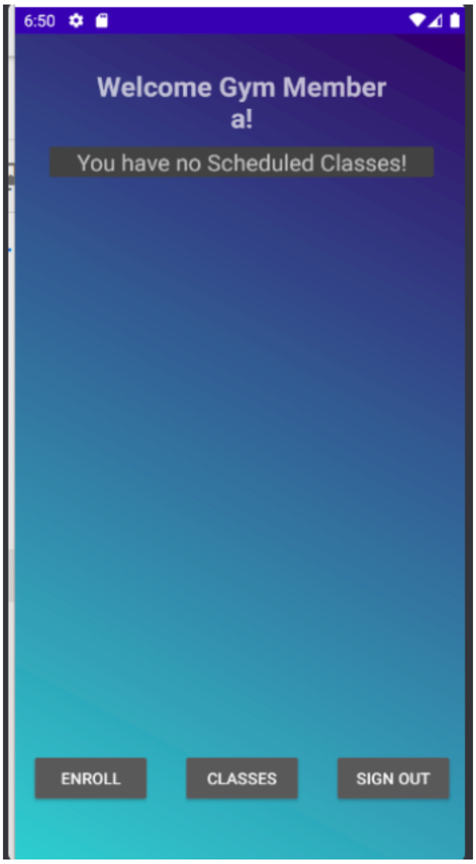

6:49
BACK
Cancel Class
Enter Class Name
SEARCH
Search Results
BASKETBALL-SUN-60-BEGINNER-1 CAPACITY
CLASS1-SUN-0-BEGINNER-50 CAPACITY
BASKETBALL-FRI-360-BEGINNER-1 CAPACITY
YOGA-SUN-90-BEGINNER-50 CAPACITY
REMOVE

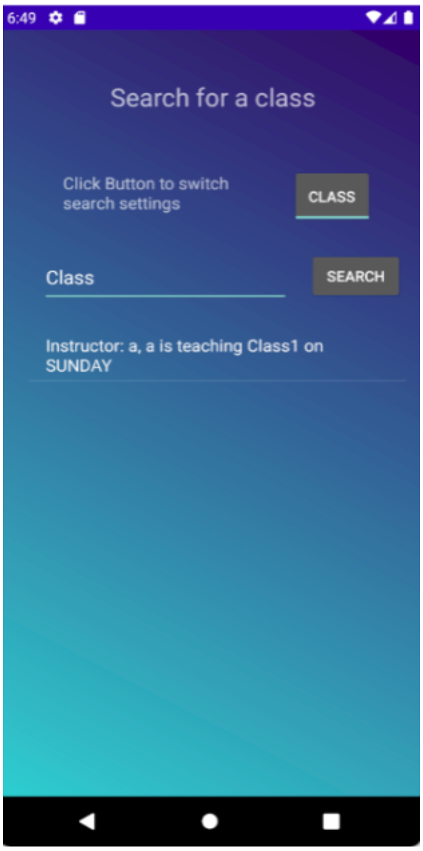
Figure 18: Instructor cancel class Page



Welcome Gym Member a!

You have no Scheduled Classes!

Figure 19: Gym member welcome Page



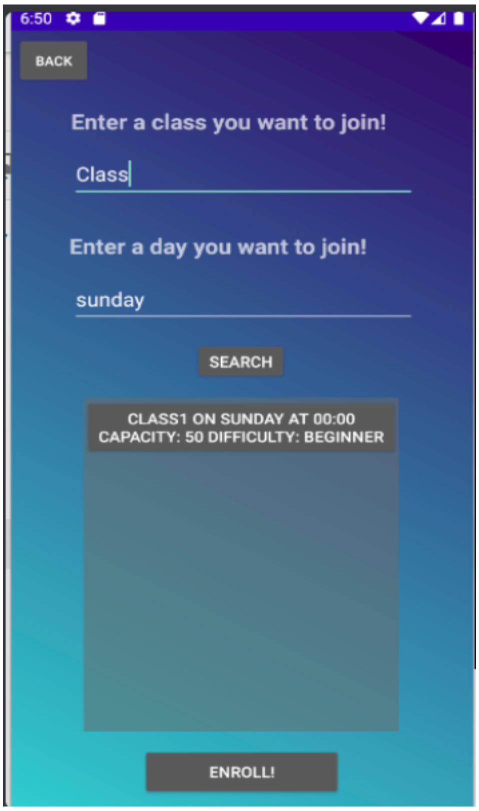
Search for a class

Click Button to switch search settings

Class

Instructor: a, a is teaching Class1 on SUNDAY

Figure 20: Gym member search for class page



Enter a class you want to join!

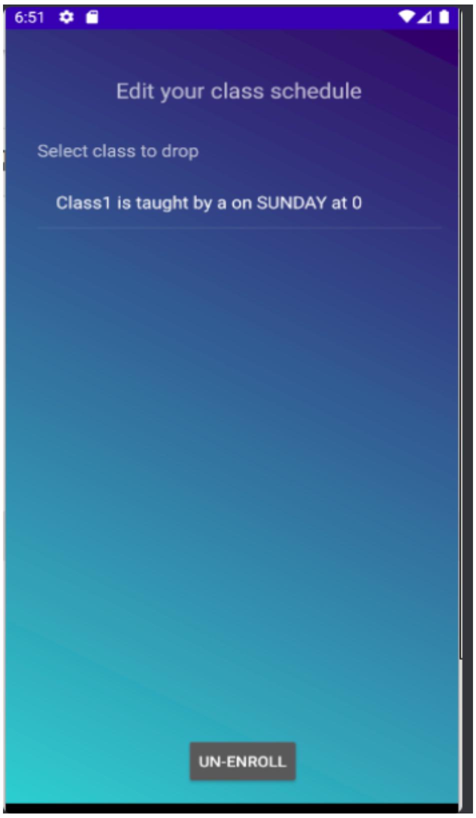
Class

Enter a day you want to join!

sunday

CLASS1 ON SUNDAY AT 00:00 CAPACITY: 50 DIFFICULTY: BEGINNER

Figure 21: Gym member enroll class Page



Edit your class schedule

Select class to drop

Class1 is taught by a on SUNDAY at 0

Figure 22: Gym member unenroll class Page

8.0 Summary

8.1 Challenges faced

Over the period of the project, we encountered numerous challenges such as the Firebase database would not allow to store arrays this issue was solved using the data structure linked lists. We also faced some challenges for the admin to delete a user, this was solved by storing email and password in database, and logging in pretending to be a user to delete them. As the project was quite large it was really difficult to keep track of everything and this lead to missing fields in the UML class diagram. The class diagram was checked multiple times to make sure each and every method and field was included.

8.2 Summary and Conclusion

In conclusion, we have a better understanding of software design life cycle. After getting the practical knowledge of UML, GitHub Android Studios, Firebase, Java, and Unit testing we have better understanding of all these concepts and we understand the process behind android application development.