Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Automation and Applied Informatics

Sahejpal Singh Arneja

MASTER MINDS

Android based Study Group Application

SUPERVISOR

Sik Dávid

BUDAPEST, 2022

Contents

1. Content Summary 3

2. Introduction 4

3. Theory 5

3.1. Kotlin 5

3.2. Firebase 5

3.2.1. Firebase Authentication 6

3.2.2. Firebase Realtime Database 6

3.3. Android Studio 7

3.3.1. XML Layouts 7

4. Design 8

4.1 Description of Package contents 8

4.2 Layout file structure 9

5. Implementation 11

5.1. Application flow 11

5.2. Data Control flow 12

6. Code Snippets 14

7. UI Elements 17

8. Summary 18

9. References 19

10. Weekly Progress 20

1. Content Summary

The aim of the project is to create a Kotlin based Study-Group application which will help students share subject material and ask each other questions and create forums for different topics. The students will be able to share materials which would be available for use to all members of a forum and would be a one stop for all the information required for a particular subject/topic .

2. Introduction

The Project is based on the concept of knowledge sharing among students, the overview was to create an Android application which would be used by students to create groups/forums to discuss concepts of different subjects and other topics that they might be interested in, all forums would be freely accessible by any person interested in the subject/topic.

3. Theory

The entire business end of the application was made using Kotlin. We used Firebase as the Data-Access Layer. The frontend of the application is made using Android Studio and XML.

3.1. Kotlin

Kotlin [1] is a modern programming language. It is concise safe, interoperable with Java and other languages and provides many ways to reuse code between multiple platforms for productive programming. Kotlin is also the recommended programming language [2] for developing android applications by Google.

3.2. Firebase

Firebase [3] is a Backend-as-a-service (BaaS) app development platform that provides hosted backed services as a Realtime Database, cloud storage, authentication, crash reporting, machine learning, remote configuration, hosting for your static files and analytics for your application(web/mobile).

Firebase is easily integrable with other services as well example: Google Play, Slack, Jira, Google Ads, Big Query.

In order to use Firebase services, the project has to be registered as a Firebase project.

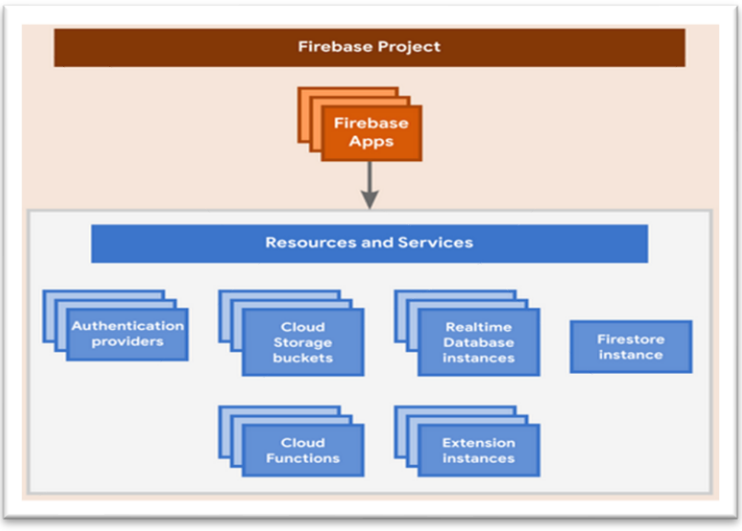

Figure 1:Firebase Project heirarchy

When we create a Firebase project, we are actually creating a Google Cloud project.

3.2.1. Firebase Authentication

Firebase Authentication [4] provides backend services, easy-to-use SDKs and ready-made UI libraries to authenticate users to the app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter. Firebase Authentication integrates tightly with other Firebase services, and it leverages industry standards like OAuth 2.0 and OpenID Connect, so it can be easily integrated with your custom backend

3.2.1.1Email and password based authentication

Authenticate users with their email addresses and passwords. The Firebase Authentication SDK provides methods to create and manage users that use their email addresses and passwords to sign in. Firebase Authentication also handles sending password reset emails.

3.2.2. Firebase Realtime Database

The Firebase Realtime Database is a NoSQL cloud database . Data is synced accross all clients in realtime , and remains available when your app goes offline. Th data is stored as JSON.

3.2.2.1. Realtime

Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receives that update within milliseconds. Provide collaborative and immersive experiences without thinking about networking code.

3.2.2.2. Offline

Firebase apps remain responsive even when offline because the Firebase Realtime Database SDK persists your data to disk. Once connectivity is reestablished, the client device receives any changes it missed, synchronizing it with the current server state.

3.3. Android Studio

Android Studio is the official IDE for Android development and includes everything you need to buildan android app. It is based on Intellij IDEA. The building of the project in Android studio is Gradle based . GUI tools in Adroid studio make the designing of user interfaces very easy.

3.3.1. XML Layouts

XML tags define the data and used to store and organize data. It's easily scalable and simple to develop. In Android, the XML is used **to implement UI-related data**, and it's a lightweight markup language that doesn't make layout heavy. XML only contains tags, while implementing they need to be just invoked.

4. Design

The application design was inspired by the Kotlin android development [5] structure. The whole project has been divided into various packages depending upon there usages for the simplicity of the person reviewing and maintaining the code.

4.1 Description of Package contents

a. adapters - The package contains all the adapters for the various Recycler Views implemented in the application, they include, Subjects, Students, User Classes and Messages.

b. Button Activities - The package contains all the activities that are instantiated by an button press event: Add a Subject, View Subject Details.

c. Chat - The package has the chat window activity

d. data - The package contains Templates for the data being fetched form the Firebase Database

e. Handlers – The package contains the Kotlin class responsible for handling all the data being processes in the application. The responsibilities include reading the requested data from the Firebase database storing them in the format of objects provided by the data templates, writing the new data and modified data back to the database.

f. login – The package includes the activities for logging in and registering the user

g. main – Includes the main activity of the application

h. menu_options – Contains the activities instantiated by clicking the menu options

i. splashscreen – This contains the activity of the Splash Screen activity that is displayed when the app is started.

j. utils - Has the class responsible for the transfer of all the data between all the classes.

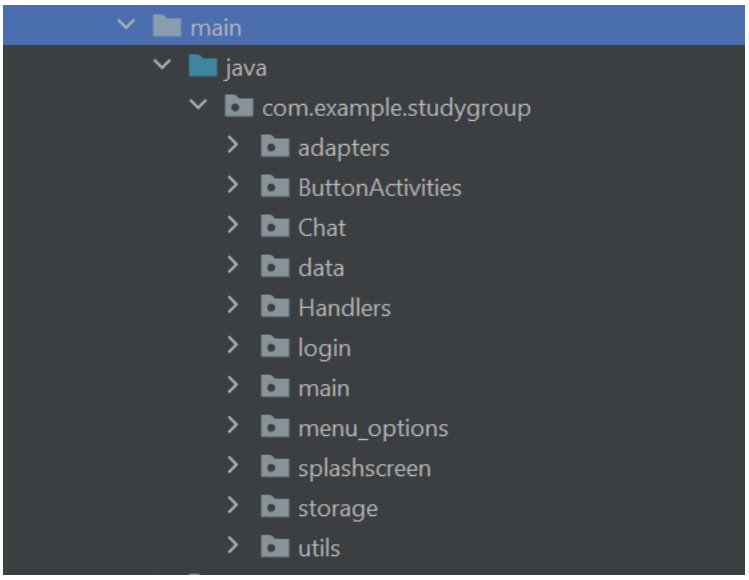

Figure 2:Project Structure

4.2 Layout file structure

The layout files have the naming convention defined by Kotlin

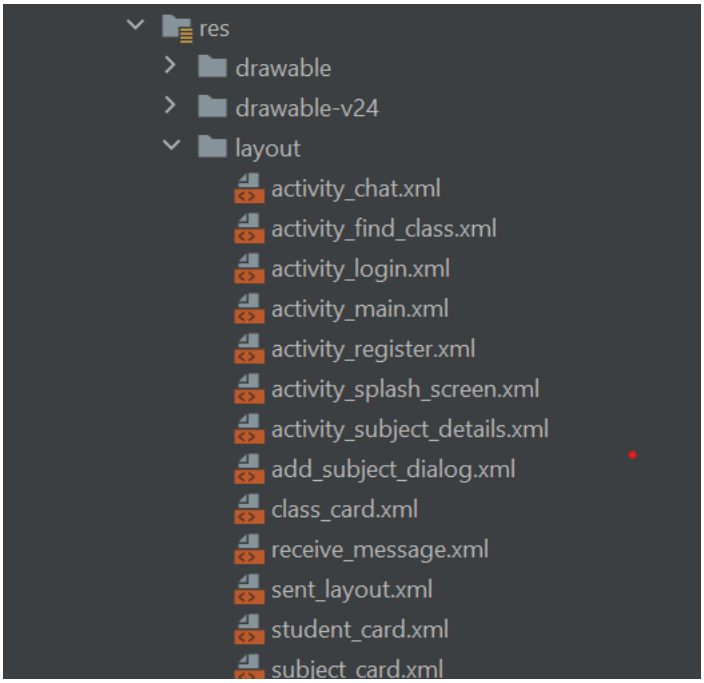

Figure 3:Layout file structure

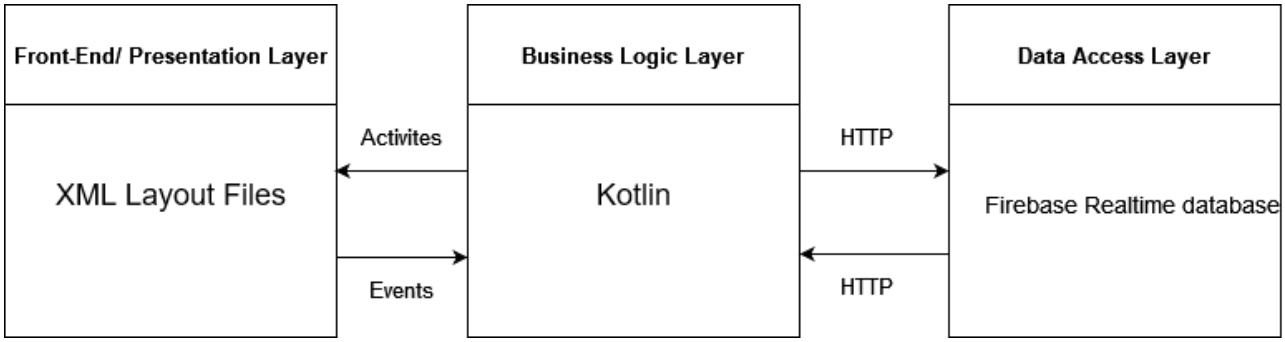

Figure 4:Application Stack

5. Implementation

5.1. Application flow

The application begins when the application is launched from the launcher. The first activity to be displayed is the splash screen, the splash screen contains the apps logo, the splash screen displays for 1 second before launching the Login activity.

The User can then login to the application with an email and a password, the data is validated before it is sent to Firebase Authentication for authentication, if the user is already registered, the Main Activity of the application will be launched.

If the user wants to register themselves, they can click on the register text and the Register Activity will be launched. The user has to enter some details before trying to register , all the data is validated before the user is registered in the Firebase Authentication list, then the Main Activity is registered.

If the user has a previous account and had joined any class they will be displayed on the Main activity as a list of Cards with the name of the class, if it is a new User the list would be empty.

The toolbar will have the users name displayed on the right and 2 options on the left, Join A Class, Logout. The user will be logged out on clicking logout and the Find a Class Activity would be launched.

The Find class Activity has a List of Subject already created, a user can join any of these classes, the new class will be added to the user classes in the main activity, the user can also add a new Subject by clicking on the add subject menu option which would open a dialog box, where the user will add the details of the new subject he wants to create, upon clicking the add button on the dialog box the subject would be written to the database and the user will be automatically be added to the Subject class they created.

The user can click on the class cards in the Main Activity, which would open the Chat Activity with the chat room of that particular class and will display the old messages of that chat room, the new user can then send messages to the members of that class. The user can also view the details of the class by clicking on the view details menu option

This description is better visualized in the following diagram

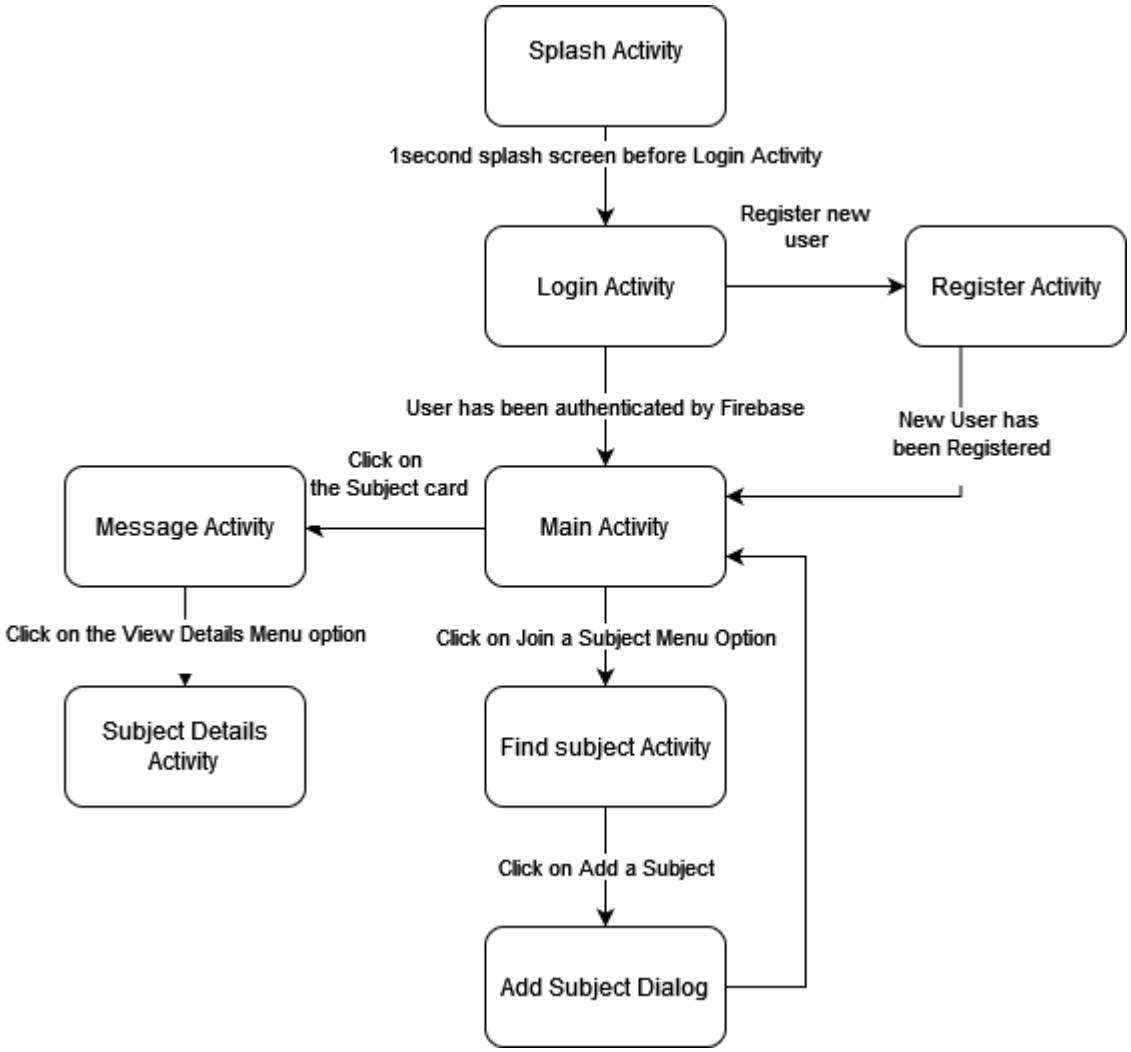

Figure 5:Application Flowchart

5.2. Data Control flow

Here I will describe the flow of the data in the application. From reading the data from the database, modelling the data to the templates described in the project, modifying the data and writing new data back to the database.

The activity initiates the Handler classes to read function. The handler class initiates a list containing of objects of the Data Template.

The Subject Data Handler creates a list of objects of the Subject Data Template.

This list is Static and can be used directly by any other activity that needs the data.

The data lists are usually passed to different adapters so that the data can be displayed in a Recycler View. The Recycler view is made up of cards which display all the details of the Data Object.

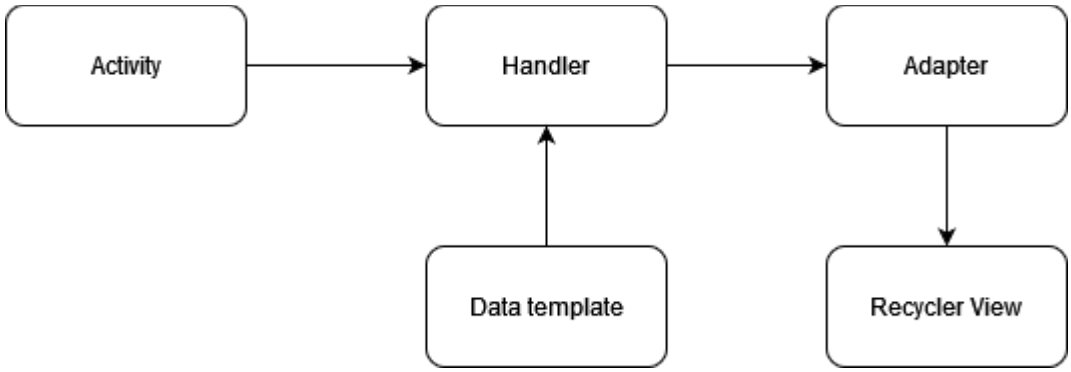

Figure 6: Data flow for all data reads

The writing of the data is also handles by the Handlers, the modified data is in the for of Data template models, do instead of just writing only the changes the handler replaces the old object with the new one. New Data, which is also in the form of Data template object are written directly to the database.

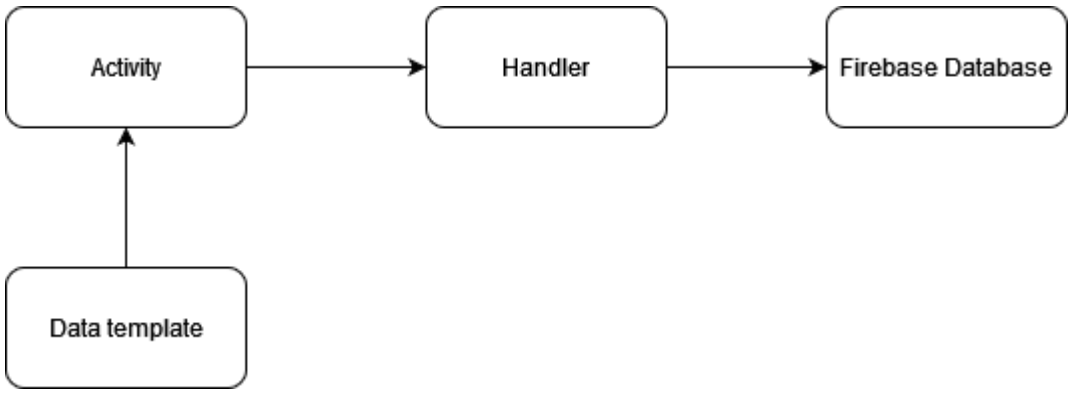

Figure 7: Writing of Data to firebase

6. Code Snippets

This section will contain snippets of the more complex parts if the code and a small explanation for them.

```
fun FirebaseLogIn(Email: String, Password: String){
    mAuth.signInWithEmailAndPassword(Email, Password).addOnCompleteListener { task ->
        if (task.isSuccessful)
        {
            User = task.result!!.user!!
            Toast.makeText( context: this, text: "Login Successfull!",Toast.LENGTH_SHORT).show()
             UserDataHandler.getUser(User.uid)
            //SubjectUserUtils.setUser(user)
            LaunchMainActivity()

        }
        else
        {
            Toast.makeText( context: this,task.exception?.message,Toast.LENGTH_LONG).show()
        }
    }
```

Figure 8:Firebase Login

The login function is a function provided by firebase that takes the already validated data to authenticate the user. A Firebase User object is created on successful login, I use this data to get the current user from the database and initialize it as the current user.

```
fun ValidateInput():Boolean
{
    FirstName= binding.etFirstName.text.toString().trim{it <=' '}
    LastName = binding.etLastName.text.toString().trim{it <=' '}
    Email = binding.etEmail.text.toString().trim{it <=' '}
    Password= binding.etPassword.text.toString().trim{it <=' '}
    RepeatPassword = binding.etRepeatPassword.text.toString().trim{it <=' '}
    when {
        TextUtils.isEmpty(Email)-> {
            Toast.makeText( context: this,  text: "Please Enter email", Toast.LENGTH_SHORT).show()
            return false
        }
        TextUtils.isEmpty(FirstName) -> {
            Toast.makeText( context: this,  text: "Please Enter First Name", Toast.LENGTH_SHORT).show()
            return false
        }
        TextUtils.isEmpty(LastName) -> {
            Toast.makeText( context: this,  text: "Please Enter Last Name", Toast.LENGTH_SHORT).show()
            return false
        }
        TextUtils.isEmpty(Password) -> {
            Toast.makeText( context: this,  text: "Please Enter Password", Toast.LENGTH_SHORT).show()
            return false
        }
        TextUtils.isEmpty(RepeatPassword) -> {
            Toast.makeText( context: this,  text: "Please Enter Password Again", Toast.LENGTH_SHORT).show()
```

Figure 9:Validation of Registering User

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder {
    if(viewType ==1){
        return ReceiveViewHolder(
            ReceiveMessageBinding.inflate(
                LayoutInflater.from(parent.context),
                parent,
                attachToParent: false
            ))
    }
    else{
        return SentViewHolder(
            SentLayoutBinding.inflate(
                LayoutInflater.from(parent.context),
                parent,
                attachToParent: false
            ))
    }

}
```

Figure 10: Message Adapter

The Message adapter has to create 2 different ViewHolders as the cards to be displayed for the messages have to be different for sent and received messaages. The Recycler View layout looks like this.

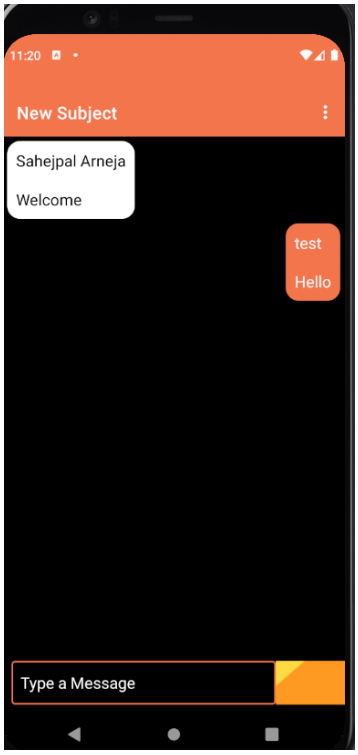

Figure 11:Chat Window

7. UI Elements

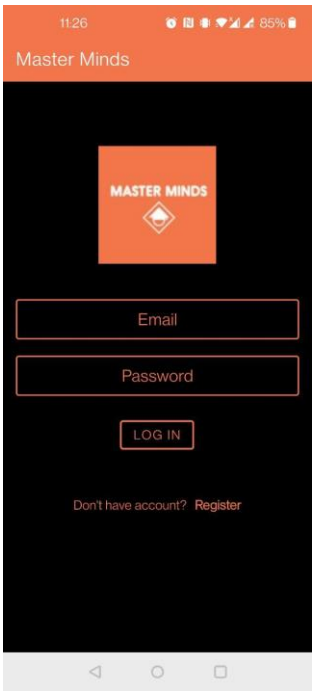

Figure 12:Login activity

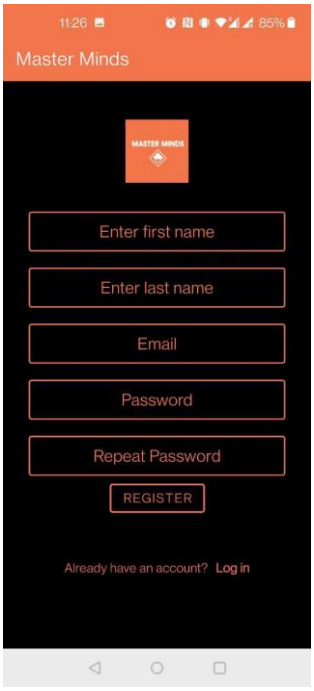

Figure 13:Register Activity

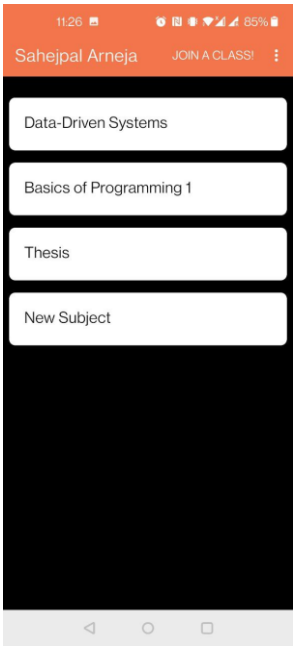

Figure 14: Join Subject

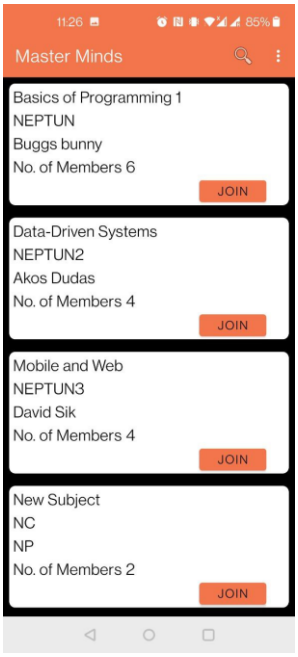

Figure 15: Join Subject

8. Summary

During the development of the project, I learned many new concepts that were unfamiliar to me:

1. Integration a backend with the frontend.
2. Importance of a definite and maintainable project structure.
3. How to write code to make the process easier for future extensions of the application and scaling.
4. Became more familiar with the android development environment and other Google services

Achievements

1. I was able to make a fully functional messaging application which is able to handle multiple users and manage real-time data

9. References

1] “Kotlin Programming language,” [Online]. Available: https://kotlinlang.org/.

2] "Android Developers," Google, [Online]. Available: https://developer.android.com/kotlin/first.

3] "Firebase Docs," Google, [Online]. Available: https://firebase.google.com/docs.

4] "Firebase Authentication docs," Google, [Online]. Available: https://firebase.google.com/docs/auth.

5] "Kotlin Coding Convention," Kotlin Foundation, [Online]. Available: https://kotlinlang.org/docs/coding-conventions.html.

10. Weekly Progress

Week No.	Date	Description
Week 1	14.02 – 20.02	Choosing of the Project Topic, Kickoff meeting
Week 2	21.02 – 27.02	Creation of the GitHub repository, writing of the one pager describing the aim for the project
Week 3	28.02 – 06.03	Creation of a login Activity, Logo for the application.
Week 4	07.03 – 13.03	Creation of the database, integration of Firebase authentication
Week 5	14.03 – 20.03	Implemented Automatic login. Reading of Subject Data from database. Displaying in a adapter
Week 6	21.03 – 27.03	Created Dialog box for adding a new Subject.
Week 7	28.03 – 03.04	Writing of user data into the database.
Week 8	04.04 – 10.04	Created cards for the display of user classes.
Week 9	11.04 – 17.04	Ability to join a class.
Week 10	18.04 – 24.04	Implemented the message activity, Created custom ViewHolder for the Chat

		feature.
Week 11	25.04 – 01.05	Integrated the messages with firebase, Added Data entity to the database
Week 12	02.05 – 08.05	Worked on synchronizing the database with real-time data
Week 13	09.05 – 15.05	Manual testing of the application, added pull-to-refresh feature
Week 14	16.05 – 22.05	Finalizing the project, refactoring, documentation.