Team Temp
28 November 2023
CS481

Team Temp Final Project Report

Demo Video

Summary/Abstract:

LearnIt is an android app made for learning with a focus on quiz-taking. It includes many features such as notifications, quiz creation and customization, Firebase/Firestore databases, permissions, sign up page, login page, forgot password functionality, user authentication, etc. To implement our features and build our app we used navigation between activities, recycler view for quiz questions, and a scoring system/scoreboard. Fragments are used for scoreboard, navigation, and quiz viewing. ViewModel is utilized for our ui. We chose our ten features throughout implementation for what would best suit our goal. We used firebase for storing quizzes themselves, the scores, and users. The data was fed in to the database so we could view what a certain user's score was for a particular quiz. To ensure proper functionality we used unit testing.

About LearnIt (introduction):

LearnIt is a quiz taking app that encourages users to build strong studying habits. Within LearnIt you can create, edit, and take your own custom quizzes as well as compare your score to other users with our scoreboard feature.

App Description (Describe the features you implemented):

Create Custom Quizzes: LearnIt allows you to create your very own custom quizzes! These quizzes can be numerical, topical, or anything in between. On LearnIt, your quizzes are fully persistent and saved to our cloud database, so you never have to worry about losing your precious quizzes/scores.

Build Good Study Habits: At LearnIt we believe the best way to expand your knowledge is to learn something new every day, so we've decided to feature push notifications to remind you to take a daily quiz if you haven't opened the app in more than 24 hours.

Discussion (Challenges, merit, etc):

When starting the Edit and Create quiz functionality, there was discussion with how exactly to execute the flow and logic of such activities. Initially we created a "storyboard" like transition template for fragments that would allow us to cycle within the page, to different questions and then simply reuse a template for a question fragment to then edit

said questions. With the storyboard template in place, we could also implement the create quiz functionality in the same way. Yet, there were a lot of dependencies with this tactic and errors arose within the storyboard template itself. This limited our productivity as a team as we needed this implementation to be done as soon as possible so we could fine tune it and establish databases for each respective system. Therefore, we had to scrap this overly complex template system and go back to what we know and trust. We continued on with the implementation of Create quiz and Edit quiz in a more straightforward way that allowed us to catch up to our initial timeline and ensure that all systems work as expected before the addition of databases.

List of Implemented Concepts (prof wants us to mention which files are used for each concept here):

Firestore Database for Quiz Storage (CreateQuiz.kt):

We are utilizing a firestore database for a few reasons, one of which is storing quizzes. When users go to create a quiz, they enter a title which is stored in the database, and then the user enters however many questions and answers they would like to have in that specific quiz. These questions and answers are sent to the database and are stored in separate arrays in the database, which allows us to easily check if the user entered the right answer when they are taking the quiz by just comparing the user input with the correct answer stored in the database.

Notifications (Alarm.kt, MainActivity.kt):

We wanted to encourage users to not only keep learning but also keep using our app day in and day out. To do this, a 24-hour push notification was implemented so the users would be reminded to open the app and take a quiz if the app had gone unopened for more than a day. To get the notification to go off every 24 hours I utilized an alarm class which schedules the notification for 24 hours ahead of the current time when they open the app.

Permissions (Alarm.kt):

In order to get the push notifications working properly we also had to implement permissions, mainly just permissions.POST_NOTIFICATIONS. If the user has not already given the app permission, then permission will be requested.

RecyclerView for Quiz Questions (QuestionsAdapter.kt, QuizDetail.kt):

We wanted to neatly display the questions and since the length of the quiz is unknown and could potentially be large, we decided that a recycler view would be the best option. When the user clicks on a specific quiz then all of the questions for that quiz are pulled from our firestore database and then loaded into the recycler view adapter.

Navigation(mainActivity.kt, activity_main_drawer.xml, activity_main.xml):

We wanted to make transitioning between fragments as seamless as possible. We decided to use a navigation drawer to do this. The drawer is activated by clicking the three lines in the top right of the app or simply swiping from the left side of the screen. The Navigation drawer enables the users to easily switch between different categories

of the application such as the Dashboard, Quizzes and Scoreboard.

Firestore Database for Users Storage(SignupFragment.kt, LoginFragment.kt):

One of the features of the firestore database is to store users information such as first and last name, username, and email. When signing up, if the username or email already has an account associated with them, the user is prompted with an error message. When logging in, the application retrieves the user data from the database. If the credentials match, the user may successfully log into their account. If the credentials do not match, an error message appears. In an instance where the user has forggotten their password, they are able to reset it. When resetting the password, the user may enter their username or email and set a new password. The application checks if an account exists for the entered email or password, and if so, the user's password may be reset and can continue to log in with their new password.

View Model LiveData(DashboardFragment.kt, DashboardViewModel.kt):

We decided to implement a View Model to display a welcome message to the user upon successfully logging into their account. The message stays persistent throughout the entire user session.

Unit Tests (MainActivityTest.kt):

Ran unit tests to ensure correct and reliable functionality for our application. In the MainActivityTest.kt, the schedule for the alarm was tested and well as some navigation methods. Especially since we had issues with navigation, we thought it would help to run unit tests to ensure correct navigation between fragments.

Take Quiz (TakeQuiz.kt):

This file is the main feature of our app, what allows you to take the quiz. The activity receives the quiz ID as an extra from the intent, retrieves the currently logged-in user's username, and initializes variables for managing quiz questions and answers. The Firestore database is queries for the specified quiz, populates lists with questions and answers, and sets up a listener for the "Next" button. As the user progresses through the quiz, questions are displayed sequentially, and their answers are checked for correctness. The total number of correct answers is tracked, and upon completion of the quiz, a score is calculated and stored in the Firestore database under the user's scores.

Scoreboard (ScoreboardAdapter.kt, ScoreboardFragment.kt):

Upon completion of a quiz, the user's score is calculated based on the number of correct answers relative to the total number of questions. This score is then stored in the Firestore database along with the associated quiz ID and the user's username. The scoreboard is designed to display users' average scores, providing a comparative overview of their quiz performance. We also added a top five to our scorboard to display the top five users. The app retrieves scores from the Firestore database, organizes them by username, calculates the average score for each user, and sorts the results in descending order. The information is then presented in a RecyclerView, displaying usernames, average scores, and a default user image.

Limitations:

Scoreboard Functionality:
There was an issue with ImageAsset and an error that said “linking failed.” Once that issue was resolved, there was an issue with the algorithm that sorts the RecyclerView so that our scoreboard was actually accurate. This was solved by changing the activity to a fragment. There was an issue with the firestore having every field say “test” when we were trying to add scores. This was resolved by the take quiz functionality and once those scores were saved to the database using TakeQuiz.kt, we could then ensure the scoreboard was accurate.

Unit Tests:
We generated unit tests for the main activity, create quiz class, login activity, and fragment, as well as the notification class. However, the functions in these components didn't align seamlessly with the typical structure found in our class activity for unit tests. Consequently, using assertEquals posed a challenge because the user defines both the input and output, rendering it flexible rather than fixed. Attempting to address this, I explored the use of mock or espresso, but encountered dependency issues during installation. Even after successfully installing them, the project failed to recognize their imports. We opted to focus on unit testing more straightforward functions, particularly those within the Main Activity.

Navigation:
When pressing the back button on the quiz detail page, the app would crash. The finish() method is used to close the current activity and return to the previous one in the activity stack. This was put inside the onClickListener for the back button.

Name	**Contribution**
William Cassel	• Solely responsible for Create Quiz functionality, including database implementation for storing quizzes. • Implemented recycler view for viewing quiz questions. • Implemented notifications that occur every 24 hours if the user has not opened the app. • Implemented database collection for storing quiz scores and made it so user scores are automatically saved to database upon finishing a quiz. • Assisted with building the general skeleton of the app. • Made it so the dashboard displays the currently logged in users top 5 quiz scores • Added app logo

Maksym Bondarenko	• Implemented a "Storyboard" fragment transition template that allows for the cycling between fragments • Created initial core functionality of Edit quiz via the mentioned Storyboard functionality • Redefined Theme.xml to allow for custom color themes • Assited with UI re-design to be more consistent within the brand's theme • Defined explicit button and header theming
Olivia Sheehan	• Ensured that the quiz id matched in our program and the firebase to ensure consistency • Created unit tests • Conclusion of the Final Report • Appendix for the Final Report • Summary/Abstract for the Final Report • Limitations for the Final Report • Some of the concepts in "List of Implemented Concepts" for the Final Report
Jared Ryan	• Created Navigation drawer to switch between main fragments and skeleton of application. • Implemented Login and Signup Fragment and functionalities. • When signing up, the users data is stored into the "users" firebase collection • When logging in, the user data is retrieved from the firebase. • Created Forgot password functionality so user can create a new password if forgotten • Created Dashboard Fragment. Upon logging in, the user is sent here. • Implemented a view model live data to display a welcome message for the current user • Created a spinner that pulls quiz names from database to easily select the quiz you want to take • Implemented user sessions. Session starts when logged in • Implemented Logout feature. Logging out ends the user session so another user can sign into their account

Casey McGuan	• Designed the core feature of the app the “take quiz” function • Debugging • Created the system to store the users scores for other app features • Cleaned up unused and excess code from different functions throughout
Jacob Irwin	• Implemented the Scoreboard Feature • Debugged the app • Implemented profile pictures, Usernames, and score within scoreboard app • Created scoreboard fragment
Simon Hausmaninger	• Implemented Edit Quiz functionality • Edit Quiz retrieves quiz data, presents it in a user interface for editing (using Recycler View), and allows users to update and save changes to the document. • Implemented scoreboard fragment connecting it to the Firebase • The Scoreboard Fragment manages a Fragment within an Android app, presenting a Recycler View populated with user scores retrieved from Firebase Firestore • The scoreboard fragment fetches data from the "quiz Scores" collection, calculates average scores per user, and dynamically updates the Recycler View to display usernames alongside their corresponding average scores in descending order. • Assisted in the passing and handling of information between fragments and activities

Conclusion:

In conclusion, we were able to successfully reach our goal of a functioning quiz taking app with an accurate scoreboard and database. Users are encouraged to study daily through a 24 hour notification system (once the user gives the app permission to do so). The user can create quizzes, edit quizzes, and most importantly take them and receive a score that will be stored in the database. This keeps the app fun while also being educational and helping the user track their learning progress. RecyclerView was used to view quiz questions and the scoreboard by retrieving the information from the database and storing it in a RecyclerView to be able to view. The Firestore databases were leveraged to store quizzes (questions and answers), scores, and user data

securely. ViewModel was used to manage data in the dashboard and scoreboard as well as display the welcome message properly. We implemented a Navigation Drawer to facilitate smooth transitions between different fragments so that our app would be simple to navigate. We encountered issues with dependencies and versioning which led us to find alternatives for unit tests and a storyboard. The scoreboard gave us trouble because of database issues since we were basing our scoreboard class off of the results in the database. We switched from an activity to a fragment for the scoreboard which helped significantly. There were also some navigation issues so we added methods like finish() within the onClickListener to be as clear as possible.

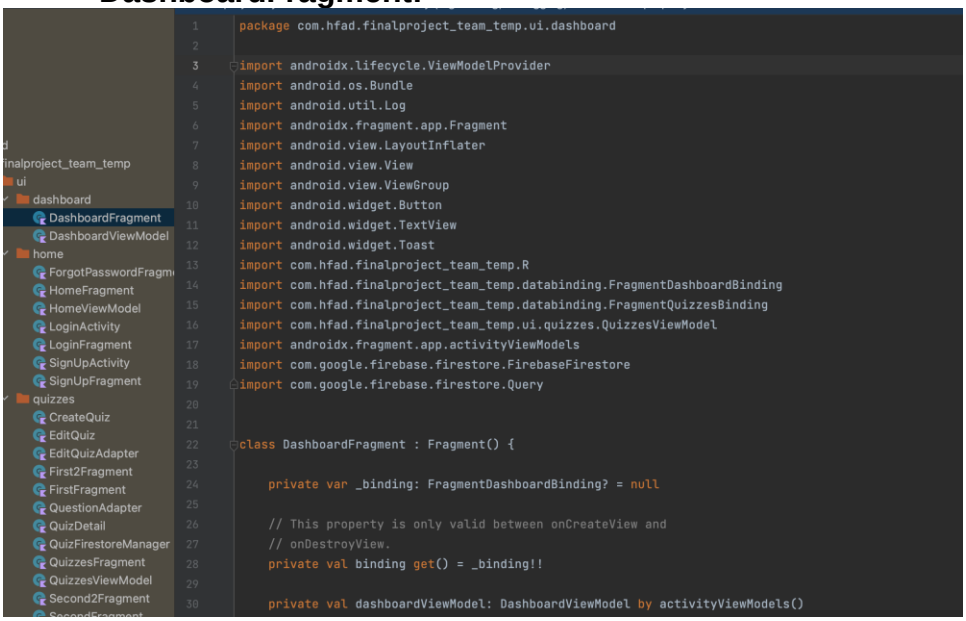
References:

Appendix:

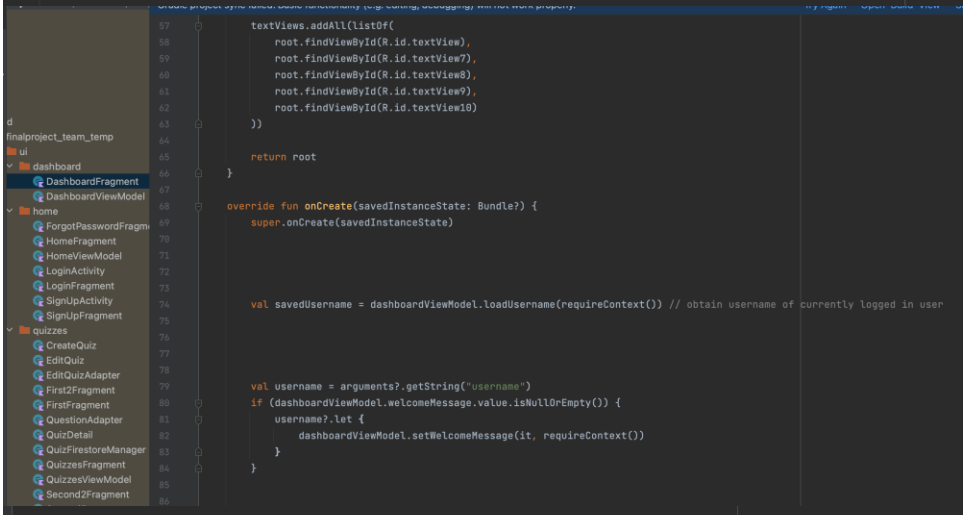
CODE
UI

Dashboard

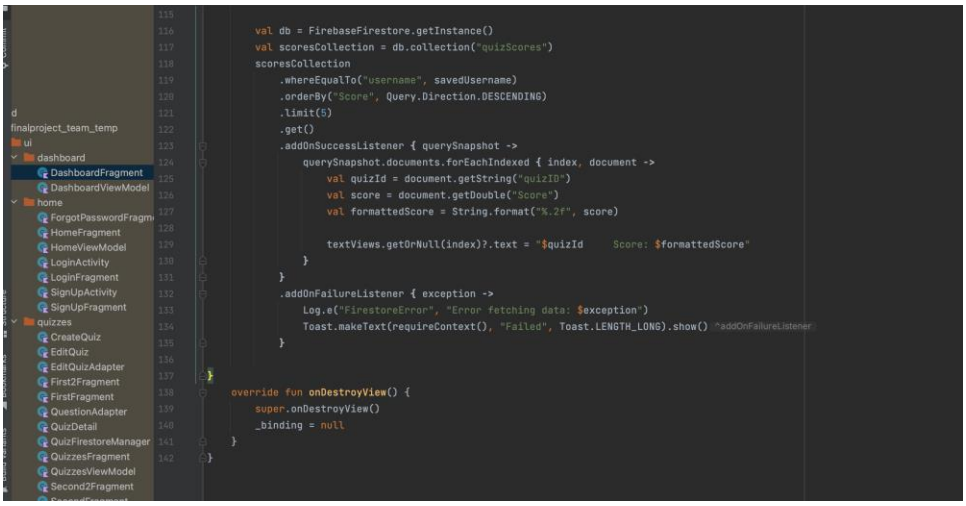
- **DashboardFragment:**



```
private val dashboardViewModel: DashboardViewModel by activityViewModels()

private val textViews = mutableListOf<TextView>()

override fun onCreateView(
    inflater: LayoutInflater,
    container: ViewGroup?,
    savedInstanceState: Bundle?
): View {
    //val galleryViewModel =
        //ViewModelProvider(this).get(DashboardViewModel::class.java)

    _binding = FragmentDashboardBinding.inflate(inflater, container, false)
    val root: View = binding.root
/*
    val textView: TextView = binding.textGallery
    galleryViewModel.text.observe(viewLifecycleOwner) {
        textView.text = it
    }

 */

    //val textView: TextView = binding.textGallery

    //dashboardViewModel.welcomeMessage.observe(viewLifecycleOwner) { message ->
       // textView.text = message
    //}
```

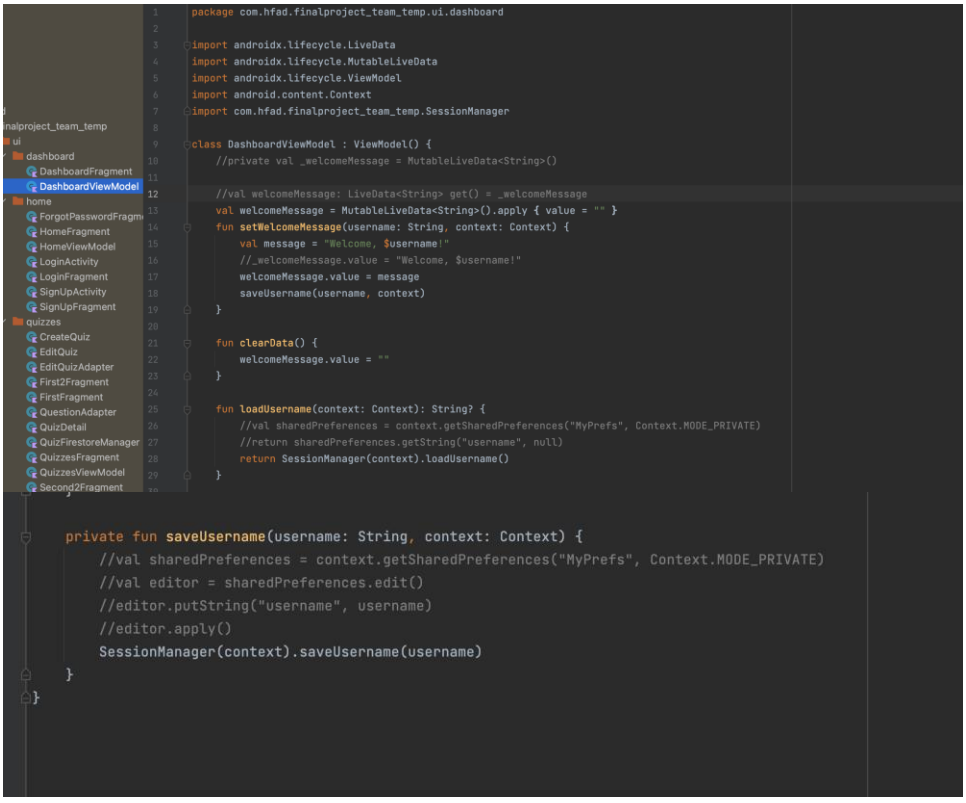



```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    /*
    val textViews = listOf<TextView>(
        view.findViewById(R.id.textView),
        view.findViewById(R.id.textView7),
        view.findViewById(R.id.textView8),
        view.findViewById(R.id.textView9),
        view.findViewById(R.id.textView10)
    )
     */

    val textView: TextView = binding.textGallery
    dashboardViewModel.welcomeMessage.observe(viewLifecycleOwner) { message ->
        textView.text = message
    }
    val savedUsername = dashboardViewModel.loadUsername(requireContext()) // obtain username of currently logged in user

    val username = arguments?.getString("username")

    if (dashboardViewModel.welcomeMessage.value.isNullOrEmpty()) {
        username?.let {
            dashboardViewModel.setWelcomeMessage(it, requireContext())
        }
    }
```



```
        val db = FirebaseFirestore.getInstance()
        val scoresCollection = db.collection("quizScores")
        scoresCollection
            .whereEqualTo("username", savedUsername)
            .orderBy("Score", Query.Direction.DESCENDING)
            .limit(5)
            .get()
            .addOnSuccessListener { querySnapshot ->
                querySnapshot.documents.forEachIndexed { index, document ->
                    val quizId = document.getString("quizID")
                    val score = document.getDouble("Score")
                    val formattedScore = String.format("%.2f", score)

                    textViews.getOrNull(index)?.text = "$quizId     Score: $formattedScore"
                }
            }
            .addOnFailureListener { exception ->
                Log.e("FirestoreError", "Error fetching data: $exception")
                Toast.makeText(requireContext(), "Failed", Toast.LENGTH_LONG).show()
            }

    }
    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

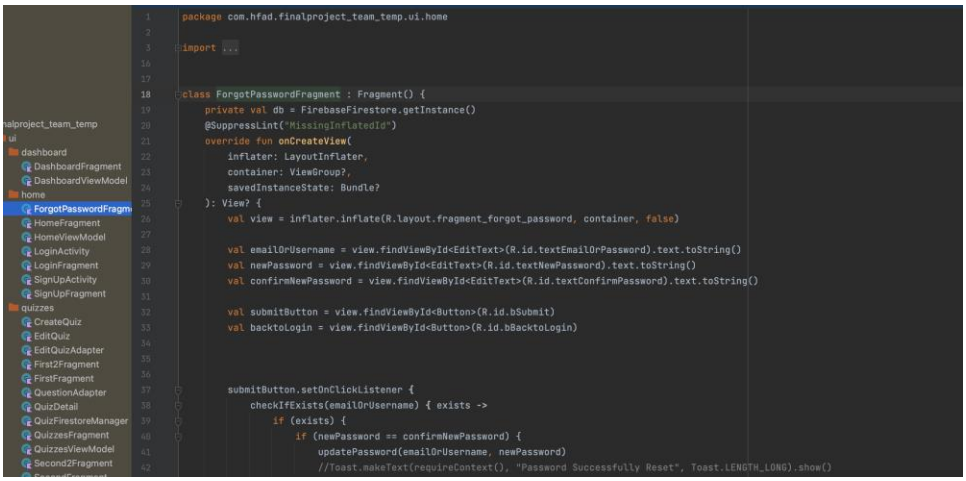
- DashboardViewModel



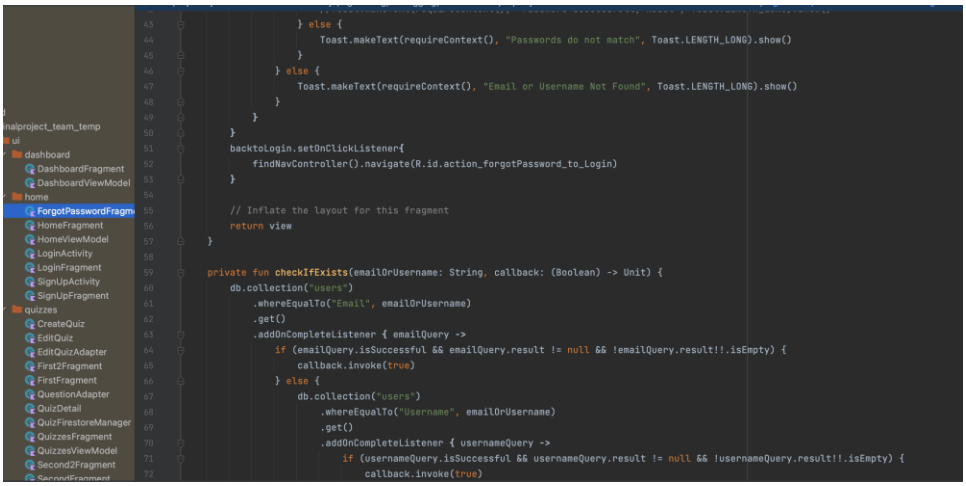
```
package com.hfad.finalproject_team_temp.ui.dashboard

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import android.content.Context
import com.hfad.finalproject_team_temp.SessionManager

class DashboardViewModel : ViewModel() {
    //private val _welcomeMessage = MutableLiveData<String>()

    //val welcomeMessage: LiveData<String> get() = _welcomeMessage
    val welcomeMessage = MutableLiveData<String>().apply { value = "" }
    fun setWelcomeMessage(username: String, context: Context) {
        val message = "Welcome, $username!"
        //_welcomeMessage.value = "Welcome, $username!"
        welcomeMessage.value = message
        saveUsername(username, context)
    }

    fun clearData() {
        welcomeMessage.value = ""
    }

    fun loadUsername(context: Context): String? {
        //val sharedPreferences = context.getSharedPreferences("MyPrefs", Context.MODE_PRIVATE)
        //return sharedPreferences.getString("username", null)
        return SessionManager(context).loadUsername()
    }

    private fun saveUsername(username: String, context: Context) {
        //val sharedPreferences = context.getSharedPreferences("MyPrefs", Context.MODE_PRIVATE)
        //val editor = sharedPreferences.edit()
        //editor.putString("username", username)
        //editor.apply()
        SessionManager(context).saveUsername(username)
    }
}
```

Home

- ForgotPasswordFragment



```
package com.hfad.finalproject_team_temp.ui.home

import ...


class ForgotPasswordFragment : Fragment() {
    private val db = FirebaseFirestore.getInstance()
    @SuppressLint("MissingInflatedId")
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_forgot_password, container, false)

        val emailOrUsername = view.findViewById<EditText>(R.id.textEmailOrPassword).text.toString()
        val newPassword = view.findViewById<EditText>(R.id.textNewPassword).text.toString()
        val confirmNewPassword = view.findViewById<EditText>(R.id.textConfirmPassword).text.toString()

        val submitButton = view.findViewById<Button>(R.id.bSubmit)
        val backtoLogin = view.findViewById<Button>(R.id.bBacktoLogin)



        submitButton.setOnClickListener {
            checkIfExists(emailOrUsername) { exists ->
                if (exists) {
                    if (newPassword == confirmNewPassword) {
                        updatePassword(emailOrUsername, newPassword)
                        //Toast.makeText(requireContext(), "Password Successfully Reset", Toast.LENGTH_LONG).show()
```



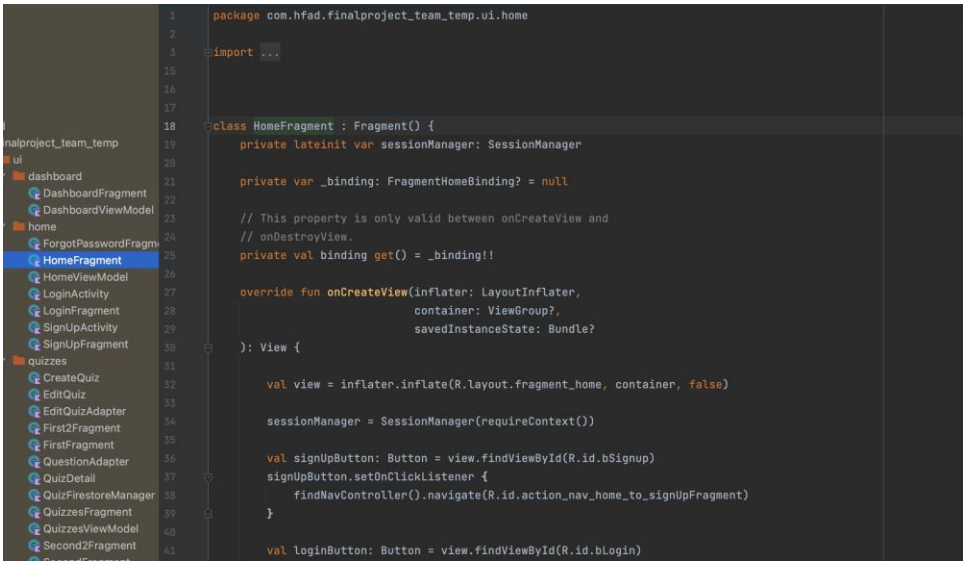
```
                    } else {
                        Toast.makeText(requireContext(), "Passwords do not match", Toast.LENGTH_LONG).show()
                    }
                } else {
                    Toast.makeText(requireContext(), "Email or Username Not Found", Toast.LENGTH_LONG).show()
                }
            }
        }
        backtoLogin.setOnClickListener{
            findNavController().navigate(R.id.action_forgotPassword_to_Login)
        }

        // Inflate the layout for this fragment
        return view
    }

    private fun checkIfExists(emailOrUsername: String, callback: (Boolean) -> Unit) {
        db.collection("users")
            .whereEqualTo("Email", emailOrUsername)
            .get()
            .addOnCompleteListener { emailQuery ->
                if (emailQuery.isSuccessful && emailQuery.result != null && !emailQuery.result!!.isEmpty) {
                    callback.invoke(true)
                } else {
                    db.collection("users")
                        .whereEqualTo("Username", emailOrUsername)
                        .get()
                        .addOnCompleteListener { usernameQuery ->
                            if (usernameQuery.isSuccessful && usernameQuery.result != null && !usernameQuery.result!!.isEmpty) {
                                callback.invoke(true)
```

```
                                callback.invoke(true)
                            } else {
                                callback.invoke(false)
                            }
                        }
                }
            }
    }

    private fun updatePassword(emailOrUsername: String, newPassword: String) {
       // val userId = "wcLM9HMaxpC9OwgSHSS2"
        db.collection("users")
            /*
            .document(userId)
            .update("Password", newPassword)
            .addOnSuccessListener {
                Log.d("UpdatePassword", "Password Updated successfully")
            }
            .addOnFailureListener{ e ->
                Log.e("UpdatePassword", "Error updating password", e)
            }

             */

            .whereEqualTo("Email", emailOrUsername)
            .get()
            .addOnSuccessListener { emailQuery ->
                if (emailQuery.documents.isNotEmpty()) {
                    val doc = emailQuery.documents[0]
                    val userId = doc.id
```

```
                Log.d("updatePassword", "Found user by email")

                db.collection("users")
                    .document(userId)
                    .update("Password", newPassword)
                    .addOnSuccessListener {
                        Log.d("UpdatePassword", "password updated successfully")
                        Toast.makeText(requireContext(), "Password Successfully Reset", Toast.LENGTH_LONG).show() ^addOnSuccessListener
                    }
                    .addOnFailureListener { e ->
                        Log.e("updatePassword", "error updating password", e)
                    } ^addOnSuccessListener
            } else {
                db.collection("users")
                    .whereEqualTo("Username", emailOrUsername)
                    .get()
                    .addOnSuccessListener { usernameQuery ->
                        if (usernameQuery.documents.isNotEmpty()) {
                            val doc = usernameQuery.documents[0]
                            val userId = doc.id

                            Log.d("updatePassword", "Found user by username")
                            db.collection("users")
                                .document(userId)
                                .update("Password", newPassword)
                                .addOnSuccessListener {
                                    Log.d("UpdatePassword", "password updated successfully")
                                    Toast.makeText(requireContext(), "Password Successfully Reset", Toast.LENGTH_LONG).show() ^addOnSuccess
                                }
                                .addOnFailureListener { e ->
```

```
                    .addOnFailureListener { e ->
                        Log.e("updatePassword", "error updating password", e)
                    }
                }
            } ^addOnSuccessListener
        }
    }

}
```

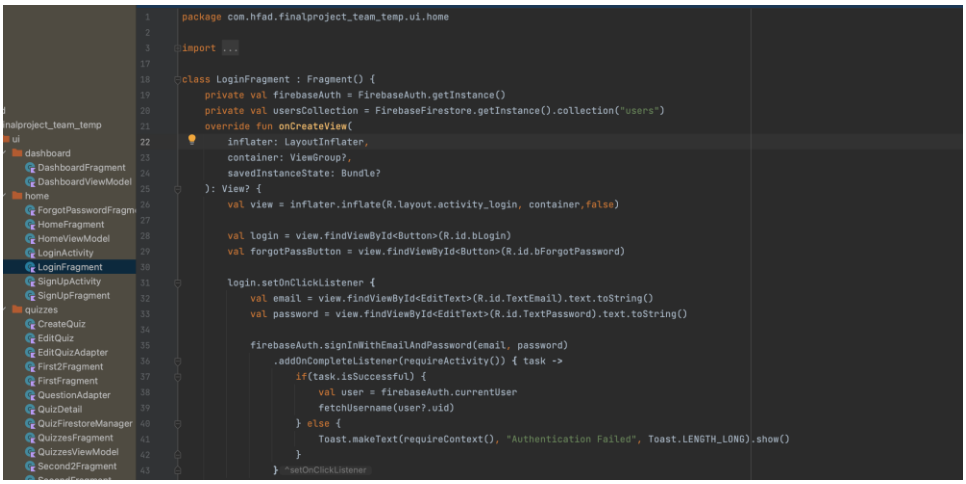
- HomeFragment

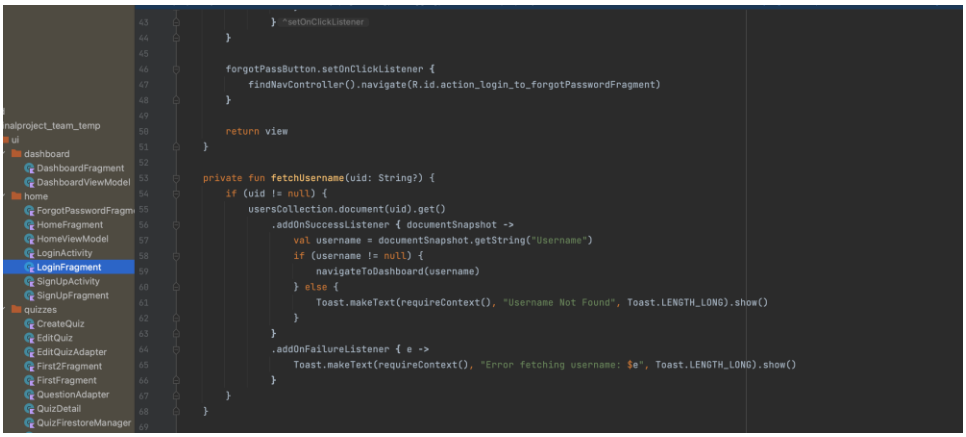


```
package com.hfad.finalproject_team_temp.ui.home

import ...



class HomeFragment : Fragment() {
    private lateinit var sessionManager: SessionManager

    private var _binding: FragmentHomeBinding? = null

    // This property is only valid between onCreateView and
    // onDestroyView.
    private val binding get() = _binding!!

    override fun onCreateView(inflater: LayoutInflater,
                              container: ViewGroup?,
                              savedInstanceState: Bundle?
    ): View {

        val view = inflater.inflate(R.layout.fragment_home, container, false)

        sessionManager = SessionManager(requireContext())

        val signUpButton: Button = view.findViewById(R.id.bSignup)
        signUpButton.setOnClickListener {
            findNavController().navigate(R.id.action_nav_home_to_signUpFragment)
        }

        val loginButton: Button = view.findViewById(R.id.bLogin)
```

- HomeViewModel

- LoginFragment

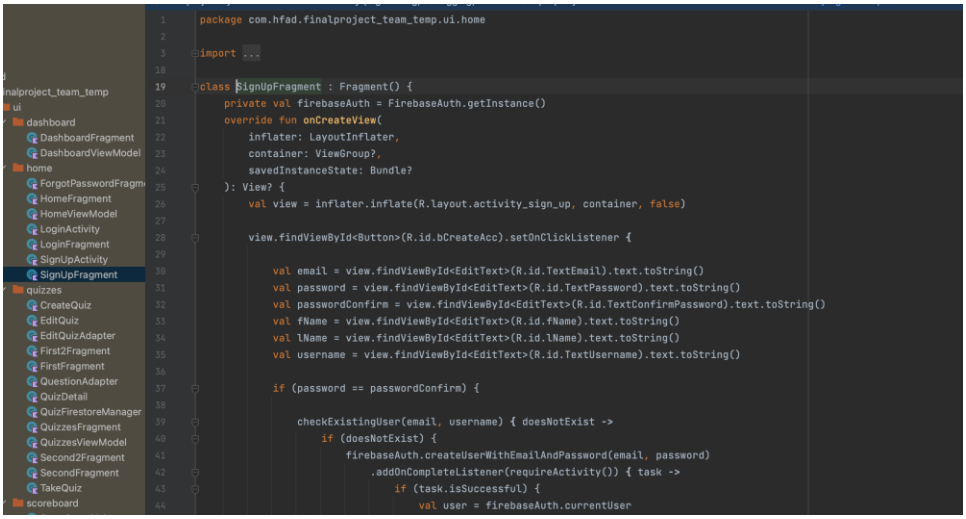




```
        }

        forgotPassButton.setOnClickListener {
            findNavController().navigate(R.id.action_login_to_forgotPasswordFragment)
        }

        return view
    }

    private fun fetchUsername(uid: String?) {
        if (uid != null) {
            usersCollection.document(uid).get()
                .addOnSuccessListener { documentSnapshot ->
                    val username = documentSnapshot.getString("Username")
                    if (username != null) {
                        navigateToDashboard(username)
                    } else {
                        Toast.makeText(requireContext(), "Username Not Found", Toast.LENGTH_LONG).show()
                    }
                }
                .addOnFailureListener { e ->
                    Toast.makeText(requireContext(), "Error fetching username: $e", Toast.LENGTH_LONG).show()
                }
        }
    }
```

```
    private fun navigateToDashboard(username: String) {
        Toast.makeText(requireContext(), "Login Successful!", Toast.LENGTH_LONG).show()
        val fragmentManager = parentFragmentManager
        val bundle = Bundle().apply {
            putString("username", username)
        }
        findNavController().navigate(R.id.action_loginFragment_to_dashboardFragment, bundle)
    }
}
```

- SignUpFragment



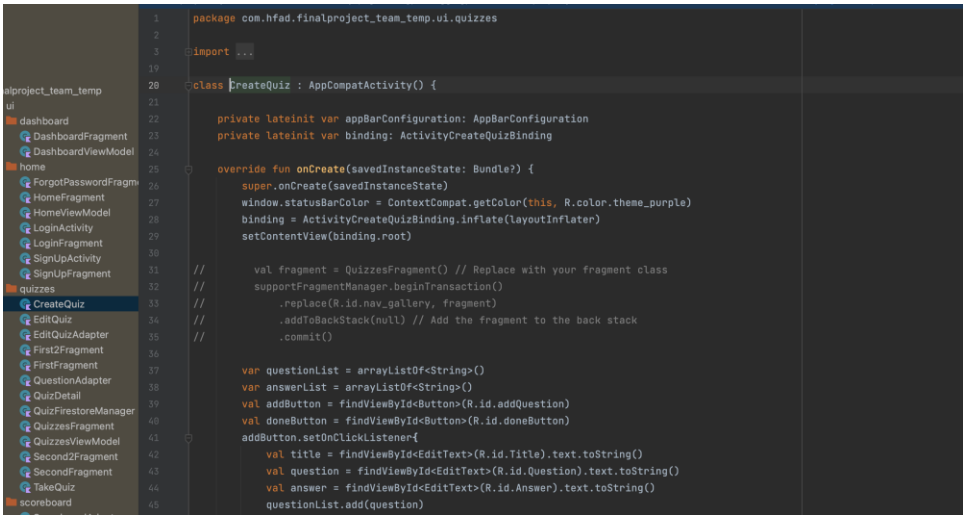
```
package com.hfad.finalproject_team_temp.ui.home

import ...

class SignUpFragment : Fragment() {
    private val firebaseAuth = FirebaseAuth.getInstance()
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.activity_sign_up, container, false)

        view.findViewById<Button>(R.id.bCreateAcc).setOnClickListener {

            val email = view.findViewById<EditText>(R.id.TextEmail).text.toString()
            val password = view.findViewById<EditText>(R.id.TextPassword).text.toString()
            val passwordConfirm = view.findViewById<EditText>(R.id.TextConfirmPassword).text.toString()
            val fName = view.findViewById<EditText>(R.id.fName).text.toString()
            val lName = view.findViewById<EditText>(R.id.lName).text.toString()
            val username = view.findViewById<EditText>(R.id.TextUsername).text.toString()

            if (password == passwordConfirm) {

                checkExistingUser(email, username) { doesNotExist ->
                    if (doesNotExist) {
                        firebaseAuth.createUserWithEmailAndPassword(email, password)
                            .addOnCompleteListener(requireActivity()) { task ->
                                if (task.isSuccessful) {
                                    val user = firebaseAuth.currentUser
```

```
                                val db = FirebaseFirestore.getInstance()
                                val userData: MutableMap<String, Any> = HashMap()

                                userData["Email"] = email
                                userData["First Name"] = fName
                                userData["Last Name"] = lName
                                userData["Username"] = username

                                db.collection("users")
                                    .document(user!!.uid)
                                    .set(userData)
                                    .addOnSuccessListener {
                                        Log.d("dbfirebase", "save: ${user}")
                                        findNavController().navigate(R.id.nav_home)
                                        Toast.makeText(requireContext(), "Account created successfully", Toast.LENGTH_LONG).show()
                                    }
                                    .addOnFailureListener{
                                        Log.d("dbfirebase Failed", "${user}")
                                    }
                            } else {
                                Toast.makeText(requireContext(), "Authentication Failed", Toast.LENGTH_LONG).show()
                            }
                        }
                    } else {
                        Toast.makeText(requireContext(), "Username or Email already exist", Toast.LENGTH_LONG).show()
                    }
                }
            } else {
                Toast.makeText(requireContext(), "Passwords do not Match", Toast.LENGTH_LONG).show()
```

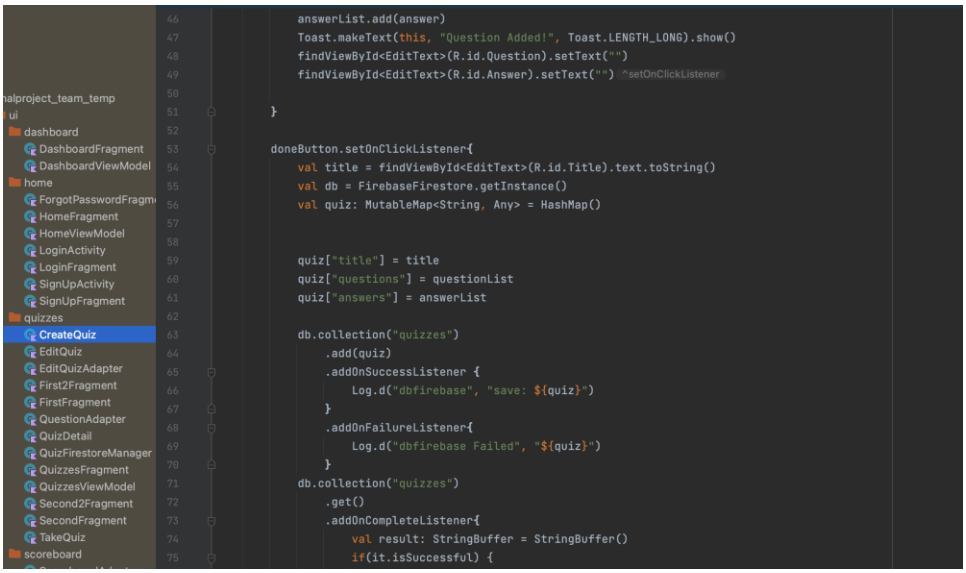
```
            }
        }
        return view
    }

    private fun checkExistingUser(email: String, username: String, callback: (Boolean) -> Unit) {
        val db = FirebaseFirestore.getInstance()

        db.collection("users")
            .whereEqualTo("Email", email)
            .get()
            .addOnCompleteListener { emailQuery ->
                if (emailQuery.isSuccessful && emailQuery.result != null && !emailQuery.result!!.isEmpty) {
                    callback.invoke(false)
                } else {
                    db.collection("users")
                        .whereEqualTo("Username", username)
                        .get()
                        .addOnCompleteListener { usernameQuery ->
                            if (usernameQuery.isSuccessful && usernameQuery.result != null && !usernameQuery.result!!.isEmpty) {
                                callback.invoke(false)
                            } else {
                                callback.invoke(true)
                            }
                        }
                }
            }
    }
}
```

Quizzes

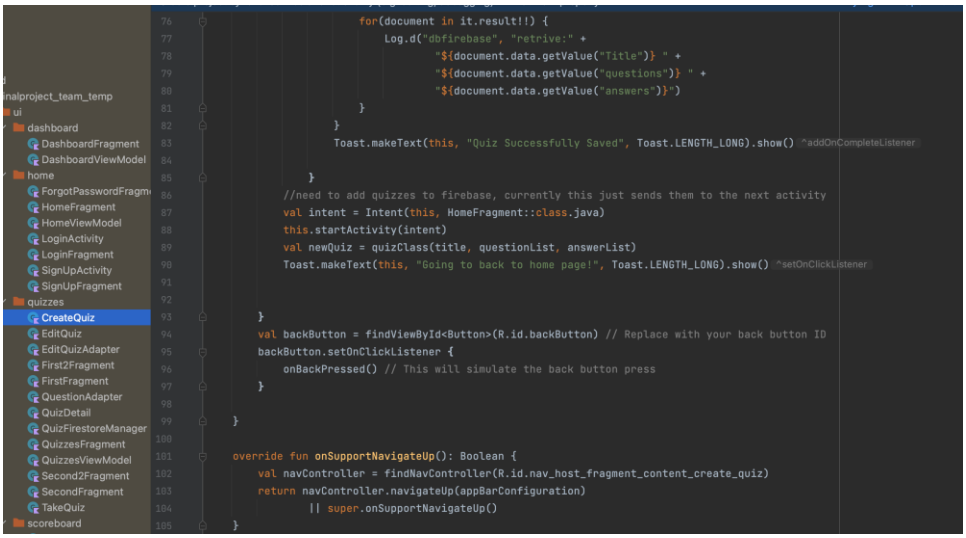
- CreateQuiz



```
package com.hfad.finalproject_team_temp.ui.quizzes

import ...

class CreateQuiz : AppCompatActivity() {

    private lateinit var appBarConfiguration: AppBarConfiguration
    private lateinit var binding: ActivityCreateQuizBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        window.statusBarColor = ContextCompat.getColor(this, R.color.theme_purple)
        binding = ActivityCreateQuizBinding.inflate(layoutInflater)
        setContentView(binding.root)

//        val fragment = QuizzesFragment() // Replace with your fragment class
//        supportFragmentManager.beginTransaction()
//            .replace(R.id.nav_gallery, fragment)
//            .addToBackStack(null) // Add the fragment to the back stack
//            .commit()

        var questionList = arrayListOf<String>()
        var answerList = arrayListOf<String>()
        val addButton = findViewById<Button>(R.id.addQuestion)
        val doneButton = findViewById<Button>(R.id.doneButton)
        addButton.setOnClickListener{
            val title = findViewById<EditText>(R.id.Title).text.toString()
            val question = findViewById<EditText>(R.id.Question).text.toString()
            val answer = findViewById<EditText>(R.id.Answer).text.toString()
            questionList.add(question)
```

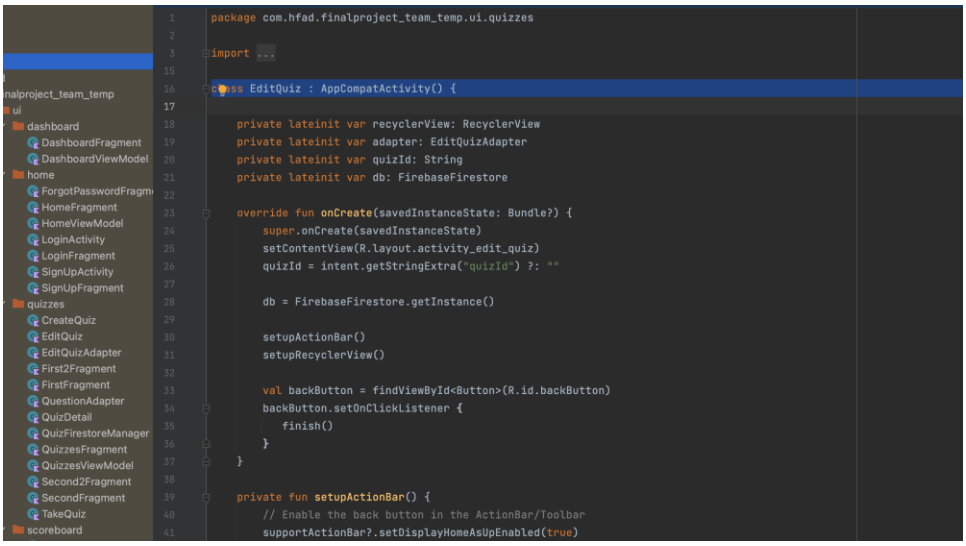


```
            answerList.add(answer)
            Toast.makeText(this, "Question Added!", Toast.LENGTH_LONG).show()
            findViewById<EditText>(R.id.Question).setText("")
            findViewById<EditText>(R.id.Answer).setText("") ^setOnClickListener

        }

        doneButton.setOnClickListener{
            val title = findViewById<EditText>(R.id.Title).text.toString()
            val db = FirebaseFirestore.getInstance()
            val quiz: MutableMap<String, Any> = HashMap()


            quiz["title"] = title
            quiz["questions"] = questionList
            quiz["answers"] = answerList

            db.collection("quizzes")
                .add(quiz)
                .addOnSuccessListener {
                    Log.d("dbfirebase", "save: ${quiz}")
                }
                .addOnFailureListener{
                    Log.d("dbfirebase Failed", "${quiz}")
                }
            db.collection("quizzes")
                .get()
                .addOnCompleteListener{
                    val result: StringBuffer = StringBuffer()
                    if(it.isSuccessful) {
```

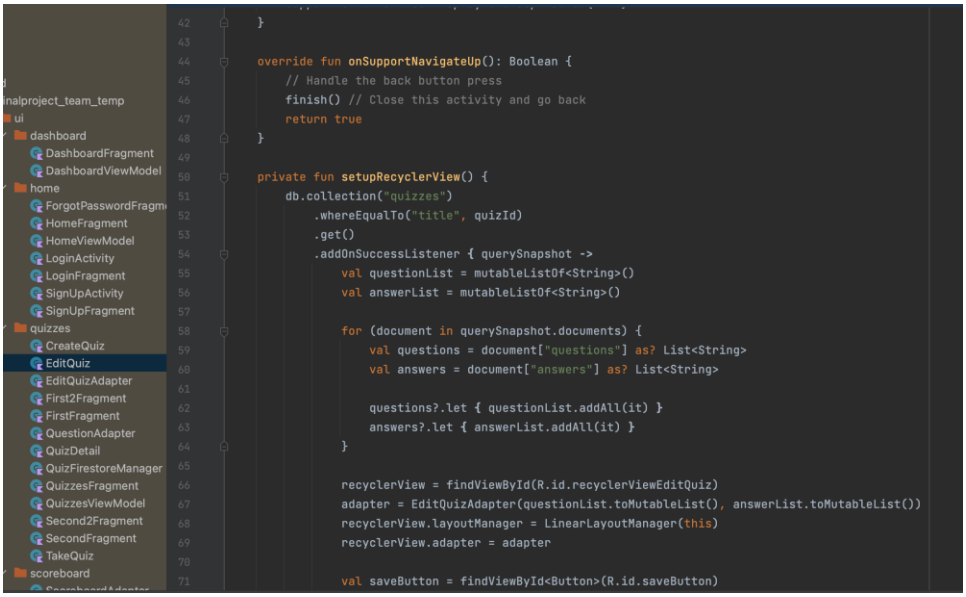


```
                    for(document in it.result!!) {
                        Log.d("dbfirebase", "retrive:" +
                                "${document.data.getValue("Title")} " +
                                "${document.data.getValue("questions")} " +
                                "${document.data.getValue("answers")}")
                    }
                }
                Toast.makeText(this, "Quiz Successfully Saved", Toast.LENGTH_LONG).show() ^addOnCompleteListener

            }
        //need to add quizzes to firebase, currently this just sends them to the next activity
        val intent = Intent(this, HomeFragment::class.java)
        this.startActivity(intent)
        val newQuiz = quizClass(title, questionList, answerList)
        Toast.makeText(this, "Going to back to home page!", Toast.LENGTH_LONG).show() ^setOnClickListener


    }
    val backButton = findViewById<Button>(R.id.backButton) // Replace with your back button ID
    backButton.setOnClickListener {
        onBackPressed() // This will simulate the back button press
    }

}

override fun onSupportNavigateUp(): Boolean {
    val navController = findNavController(R.id.nav_host_fragment_content_create_quiz)
    return navController.navigateUp(appBarConfiguration)
            || super.onSupportNavigateUp()
}
```

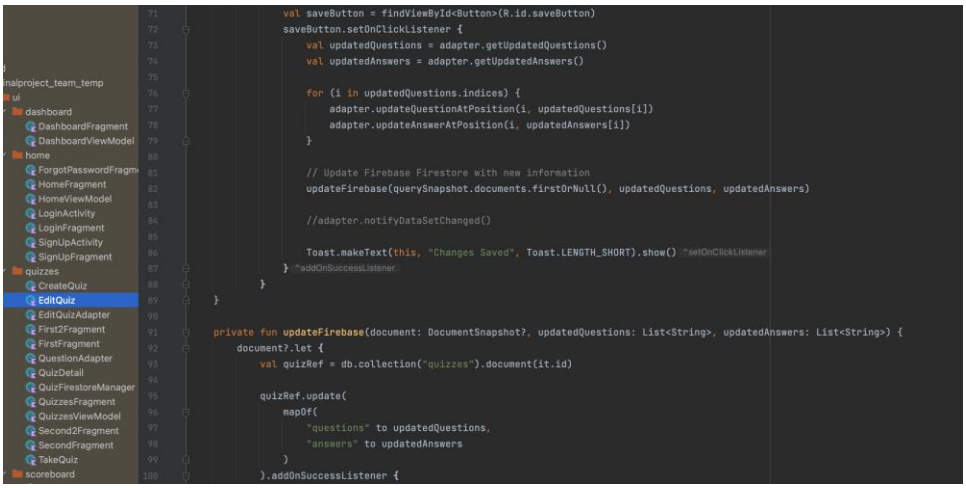
- EditQuiz



```
package com.hfad.finalproject_team_temp.ui.quizzes

import ...

class EditQuiz : AppCompatActivity() {

    private lateinit var recyclerView: RecyclerView
    private lateinit var adapter: EditQuizAdapter
    private lateinit var quizId: String
    private lateinit var db: FirebaseFirestore

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_edit_quiz)
        quizId = intent.getStringExtra("quizId") ?: ""

        db = FirebaseFirestore.getInstance()

        setupActionBar()
        setupRecyclerView()

        val backButton = findViewById<Button>(R.id.backButton)
        backButton.setOnClickListener {
            finish()
        }
    }

    private fun setupActionBar() {
        // Enable the back button in the ActionBar/Toolbar
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
```



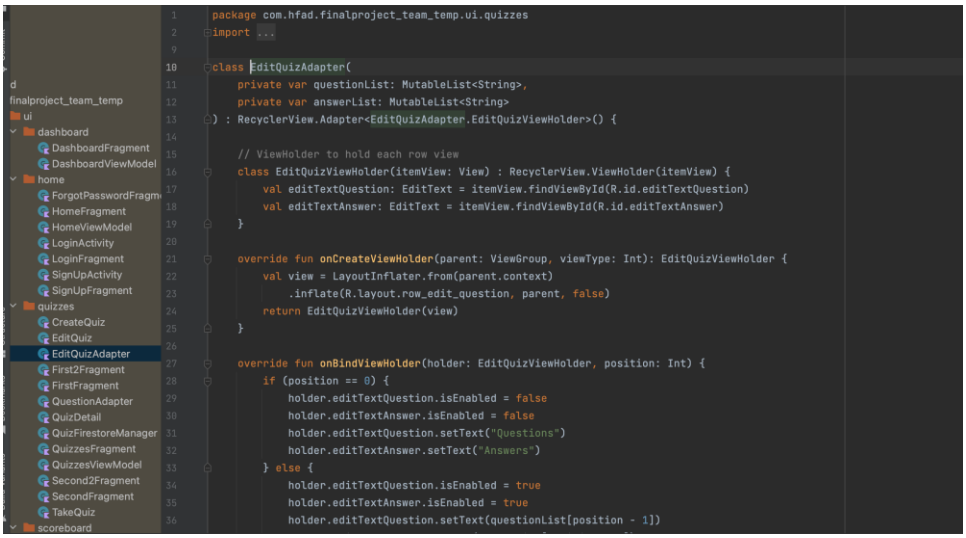
```
    }

    override fun onSupportNavigateUp(): Boolean {
        // Handle the back button press
        finish() // Close this activity and go back
        return true
    }

    private fun setupRecyclerView() {
        db.collection("quizzes")
            .whereEqualTo("title", quizId)
            .get()
            .addOnSuccessListener { querySnapshot ->
                val questionList = mutableListOf<String>()
                val answerList = mutableListOf<String>()

                for (document in querySnapshot.documents) {
                    val questions = document["questions"] as? List<String>
                    val answers = document["answers"] as? List<String>

                    questions?.let { questionList.addAll(it) }
                    answers?.let { answerList.addAll(it) }
                }

                recyclerView = findViewById(R.id.recyclerViewEditQuiz)
                adapter = EditQuizAdapter(questionList.toMutableList(), answerList.toMutableList())
                recyclerView.layoutManager = LinearLayoutManager(this)
                recyclerView.adapter = adapter

                val saveButton = findViewById<Button>(R.id.saveButton)
```



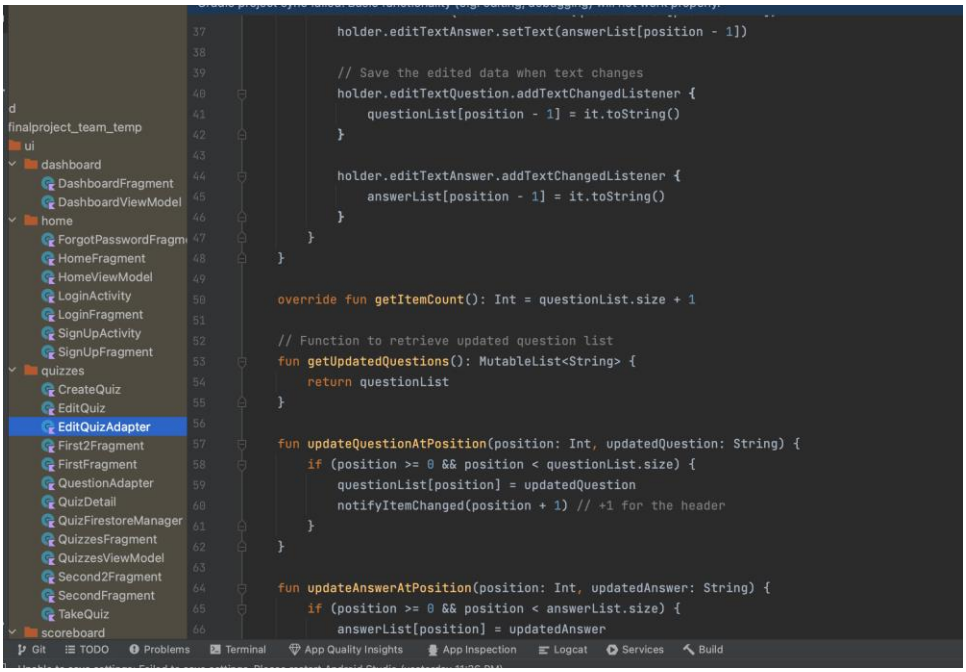
```
                val saveButton = findViewById<Button>(R.id.saveButton)
                saveButton.setOnClickListener {
                    val updatedQuestions = adapter.getUpdatedQuestions()
                    val updatedAnswers = adapter.getUpdatedAnswers()

                    for (i in updatedQuestions.indices) {
                        adapter.updateQuestionAtPosition(i, updatedQuestions[i])
                        adapter.updateAnswerAtPosition(i, updatedAnswers[i])
                    }

                    // Update Firebase Firestore with new information
                    updateFirebase(querySnapshot.documents.firstOrNull(), updatedQuestions, updatedAnswers)

                    //adapter.notifyDataSetChanged()

                    Toast.makeText(this, "Changes Saved", Toast.LENGTH_SHORT).show() ^setOnClickListener
                } ^addOnSuccessListener
            }
    }

    private fun updateFirebase(document: DocumentSnapshot?, updatedQuestions: List<String>, updatedAnswers: List<String>) {
        document?.let {
            val quizRef = db.collection("quizzes").document(it.id)

            quizRef.update(
                mapOf(
                    "questions" to updatedQuestions,
                    "answers" to updatedAnswers
                )
            ).addOnSuccessListener {
```

```
        }
    }

    private fun updateFirebase(document: DocumentSnapshot?, updatedQuestions: List<String>, updatedAnswers: List<String>) {
        document?.let {
            val quizRef = db.collection("quizzes").document(it.id)

            quizRef.update(
                mapOf(
                    "questions" to updatedQuestions,
                    "answers" to updatedAnswers
                )
            ).addOnSuccessListener {
                Log.d("EditQuiz", "Document successfully updated in Firestore!")
            }.addOnFailureListener { e ->
                Log.w("EditQuiz", "Error updating document", e)
            }
        }
    }
}
```

- EditQuizAdapter

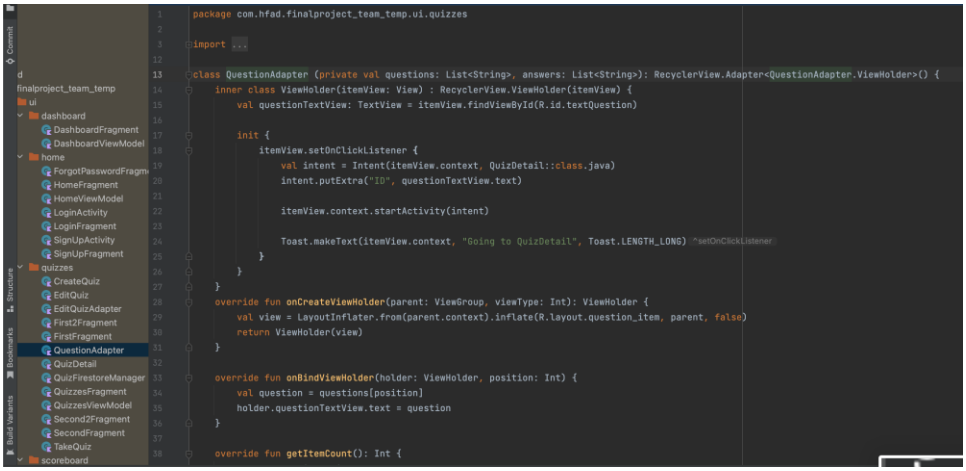


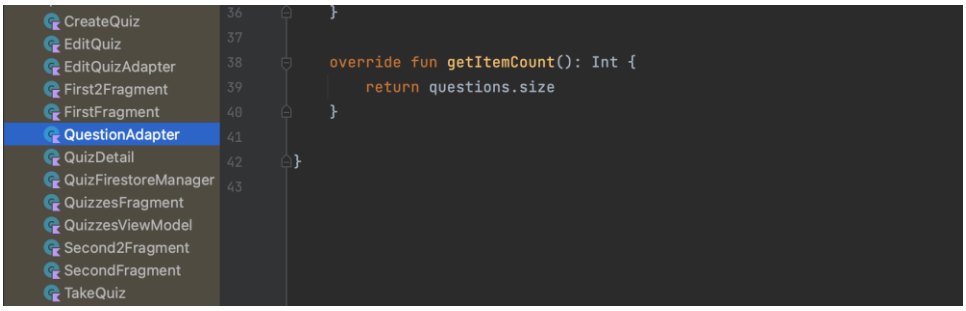
```
package com.hfad.finalproject_team_temp.ui.quizzes
import ...

class EditQuizAdapter(
    private var questionList: MutableList<String>,
    private var answerList: MutableList<String>
) : RecyclerView.Adapter<EditQuizAdapter.EditQuizViewHolder>() {

    // ViewHolder to hold each row view
    class EditQuizViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val editTextQuestion: EditText = itemView.findViewById(R.id.editTextQuestion)
        val editTextAnswer: EditText = itemView.findViewById(R.id.editTextAnswer)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): EditQuizViewHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.row_edit_question, parent, false)
        return EditQuizViewHolder(view)
    }

    override fun onBindViewHolder(holder: EditQuizViewHolder, position: Int) {
        if (position == 0) {
            holder.editTextQuestion.isEnabled = false
            holder.editTextAnswer.isEnabled = false
            holder.editTextQuestion.setText("Questions")
            holder.editTextAnswer.setText("Answers")
        } else {
            holder.editTextQuestion.isEnabled = true
            holder.editTextAnswer.isEnabled = true
            holder.editTextQuestion.setText(questionList[position - 1])
```



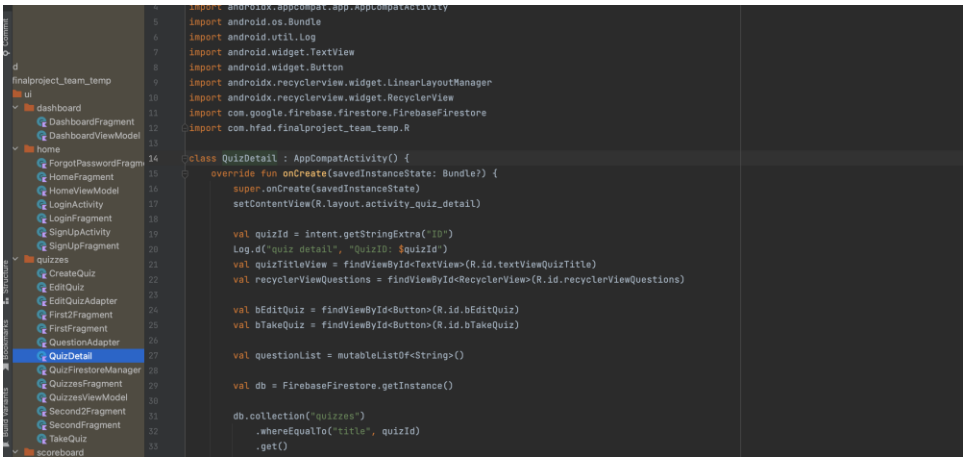
```
            holder.editTextAnswer.setText(answerList[position - 1])

            // Save the edited data when text changes
            holder.editTextQuestion.addTextChangedListener {
                questionList[position - 1] = it.toString()
            }

            holder.editTextAnswer.addTextChangedListener {
                answerList[position - 1] = it.toString()
            }
        }
    }

    override fun getItemCount(): Int = questionList.size + 1

    // Function to retrieve updated question list
    fun getUpdatedQuestions(): MutableList<String> {
        return questionList
    }

    fun updateQuestionAtPosition(position: Int, updatedQuestion: String) {
        if (position >= 0 && position < questionList.size) {
            questionList[position] = updatedQuestion
            notifyItemChanged(position + 1) // +1 for the header
        }
    }

    fun updateAnswerAtPosition(position: Int, updatedAnswer: String) {
        if (position >= 0 && position < answerList.size) {
            answerList[position] = updatedAnswer
```

- QuestionAdapter





- QuizDetail

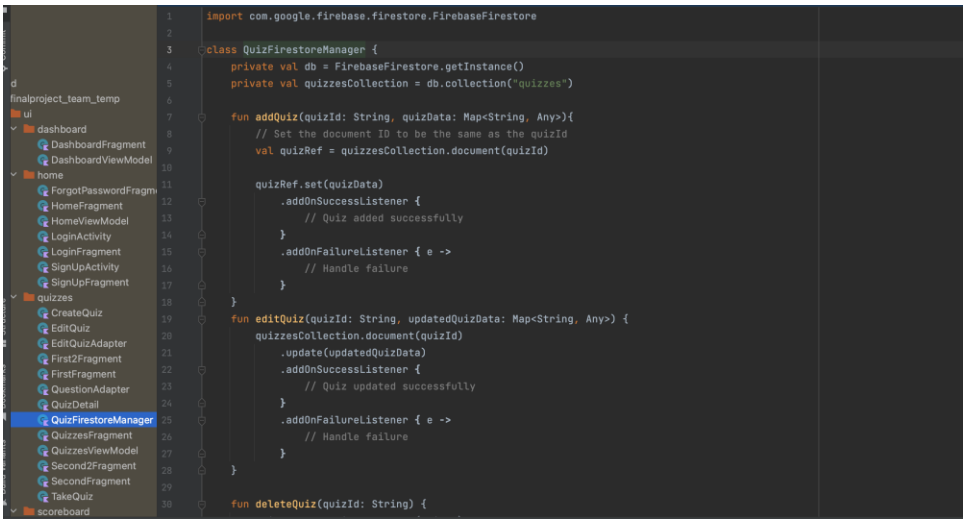


```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.TextView
import android.widget.Button
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.firebase.firestore.FirebaseFirestore
import com.hfad.finalproject_team_temp.R

class QuizDetail : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_quiz_detail)

        val quizId = intent.getStringExtra("ID")
        Log.d("quiz detail", "QuizID: $quizId")
        val quizTitleView = findViewById<TextView>(R.id.textViewQuizTitle)
        val recyclerViewQuestions = findViewById<RecyclerView>(R.id.recyclerViewQuestions)

        val bEditQuiz = findViewById<Button>(R.id.bEditQuiz)
        val bTakeQuiz = findViewById<Button>(R.id.bTakeQuiz)

        val questionList = mutableListOf<String>()

        val db = FirebaseFirestore.getInstance()

        db.collection("quizzes")
            .whereEqualTo("title", quizId)
            .get()
```

```
            .addOnSuccessListener { querySnapshot ->
                for (document in querySnapshot.documents) {
                    val questions = document["questions"] as? List<String>
                    if (questions != null) {
                        questionList.addAll(questions)
                    }
                }

                recyclerViewQuestions.layoutManager = LinearLayoutManager(this)
                val answerList = listOf(
                    "Answer 1",
                    "Answer 2",
                    "Answer 3"
                )

                quizTitleView.text = quizId.toString()
                var questionAdapter = QuestionAdapter(questionList, answerList)
                recyclerViewQuestions.adapter = questionAdapter
                val quizIdList = listOf(quizId)

                val bBack = findViewById<Button>(R.id.bBack)
                bBack.setOnClickListener {
                    //startActivity(Intent(this, QuizzesFragment::class.java))
                    finish()
                }
                bEditQuiz.setOnClickListener {
                    val intent = Intent(this, EditQuiz::class.java)
                    intent.putExtra("quizId", quizId)
                    startActivity(intent)
                }
```

```
                    startActivity(intent)
                }
                bTakeQuiz.setOnClickListener {
                    val intent = Intent(this, TakeQuiz::class.java)
                    intent.putExtra("quizId", quizId)
                    startActivity(intent)
                } ^addOnSuccessListener
            }
    }
}
```

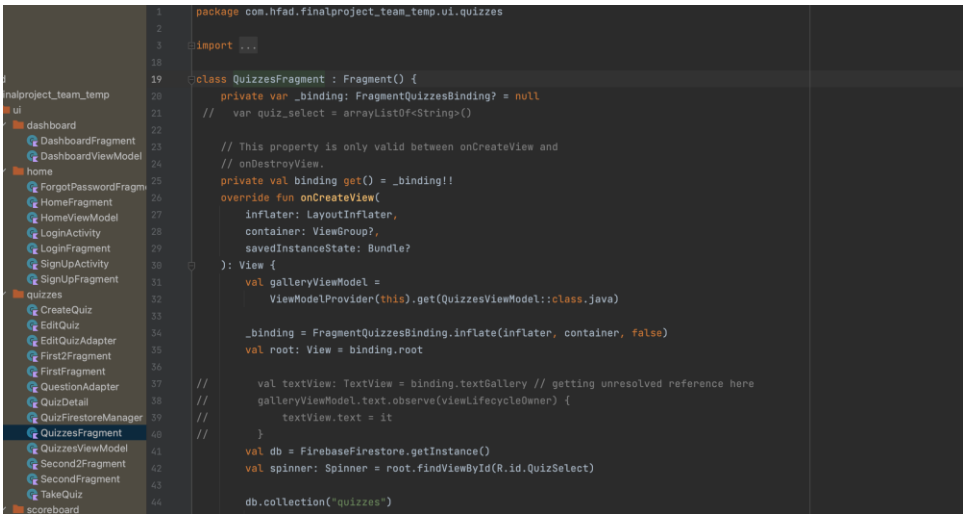
- QuizFirestoreManager



```
import com.google.firebase.firestore.FirebaseFirestore

class QuizFirestoreManager {
    private val db = FirebaseFirestore.getInstance()
    private val quizzesCollection = db.collection("quizzes")

    fun addQuiz(quizId: String, quizData: Map<String, Any>){
        // Set the document ID to be the same as the quizId
        val quizRef = quizzesCollection.document(quizId)

        quizRef.set(quizData)
            .addOnSuccessListener {
                // Quiz added successfully
            }
            .addOnFailureListener { e ->
                // Handle failure
            }
    }
    fun editQuiz(quizId: String, updatedQuizData: Map<String, Any>) {
        quizzesCollection.document(quizId)
            .update(updatedQuizData)
            .addOnSuccessListener {
                // Quiz updated successfully
            }
            .addOnFailureListener { e ->
                // Handle failure
            }
    }

    fun deleteQuiz(quizId: String) {
```

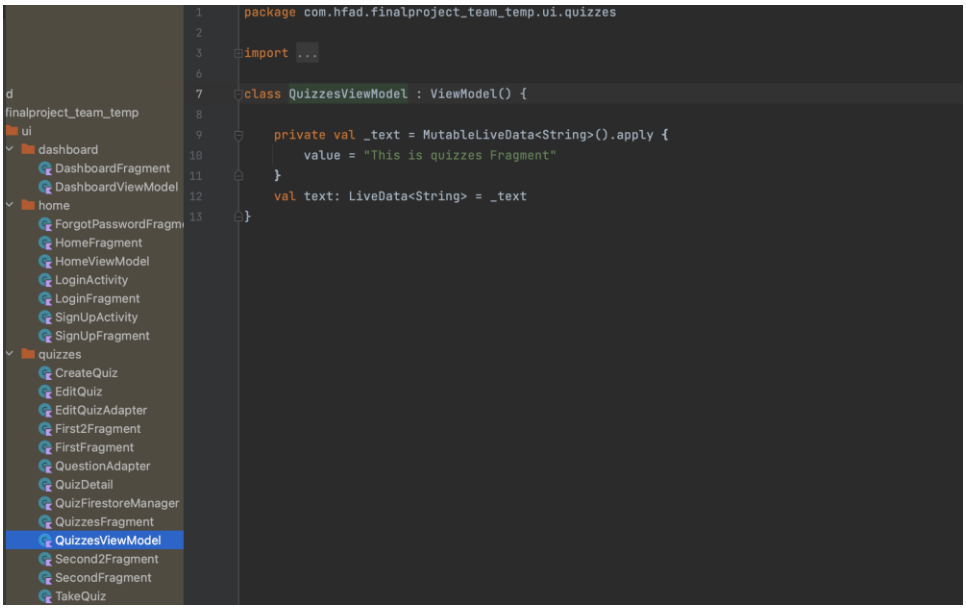
```
    }

    fun deleteQuiz(quizId: String) {
        quizzesCollection.document(quizId)
            .delete()
            .addOnSuccessListener {
                // Quiz deleted successfully
            }
            .addOnFailureListener { e ->
                // Handle failure
            }
    }
```

- QuizzesFragment

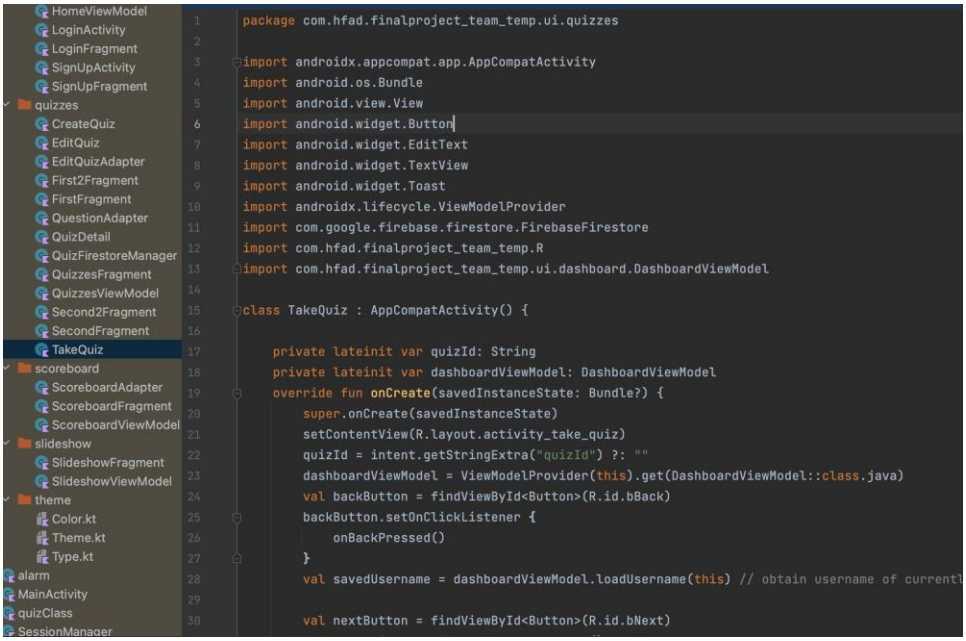


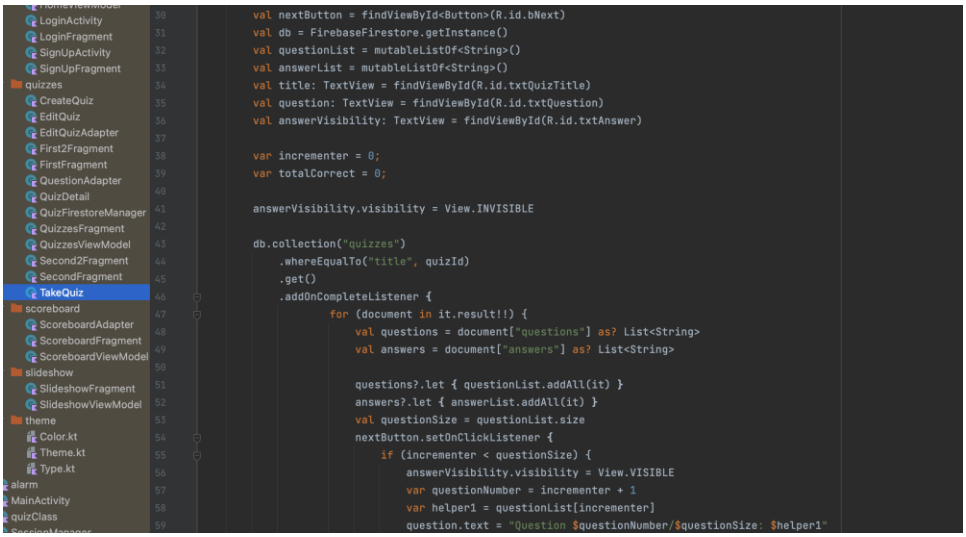
```
package com.hfad.finalproject_team_temp.ui.quizzes

import ...

class QuizzesFragment : Fragment() {
    private var _binding: FragmentQuizzesBinding? = null
//   var quiz_select = arrayListOf<String>()

    // This property is only valid between onCreateView and
    // onDestroyView.
    private val binding get() = _binding!!
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        val galleryViewModel =
            ViewModelProvider(this).get(QuizzesViewModel::class.java)

        _binding = FragmentQuizzesBinding.inflate(inflater, container, false)
        val root: View = binding.root

//        val textView: TextView = binding.textGallery // getting unresolved reference here
//        galleryViewModel.text.observe(viewLifecycleOwner) {
//            textView.text = it
//        }
        val db = FirebaseFirestore.getInstance()
        val spinner: Spinner = root.findViewById(R.id.QuizSelect)

        db.collection("quizzes")
```

- QuizzesViewModel

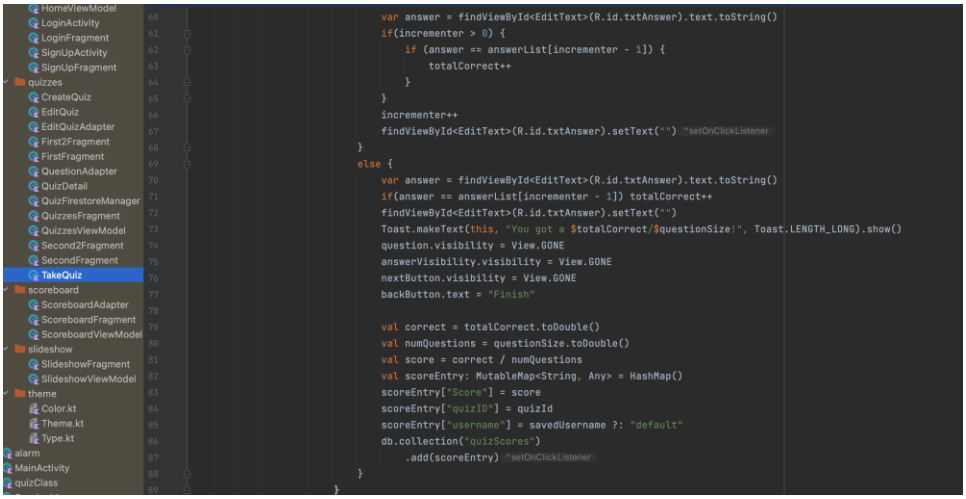

d
finalproject_team_temp
ui
dashboard
DashboardFragment
DashboardViewModel
home
ForgotPasswordFragm
HomeFragment
HomeViewModel
LoginActivity
LoginFragment
SignUpActivity
SignUpFragment
quizzes
CreateQuiz
EditQuiz
EditQuizAdapter
First2Fragment
FirstFragment
QuestionAdapter
QuizDetail
QuizFirestoreManager
QuizzesFragment
QuizzesViewModel
Second2Fragment
SecondFragment
TakeQuiz
package com.hfad.finalproject_team_temp.ui.quizzes
import ...
class QuizzesViewModel : ViewModel() {
private val _text = MutableLiveData<String>().apply {
value = "This is quizzes Fragment"
}
val text: LiveData<String> = _text
}

- TakeQuiz


HomeViewModel
LoginActivity
LoginFragment
SignUpActivity
SignUpFragment
quizzes
CreateQuiz
EditQuiz
EditQuizAdapter
First2Fragment
FirstFragment
QuestionAdapter
QuizDetail
QuizFirestoreManager
QuizzesFragment
QuizzesViewModel
Second2Fragment
SecondFragment
TakeQuiz
scoreboard
ScoreboardAdapter
ScoreboardFragment
ScoreboardViewModel
slideshow
SlideshowFragment
SlideshowViewModel
theme
Color.kt
Theme.kt
Type.kt
alarm
MainActivity
quizClass
SessionManager
package com.hfad.finalproject_team_temp.ui.quizzes
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.lifecycle.ViewModelProvider
import com.google.firebase.firestore.FirebaseFirestore
import com.hfad.finalproject_team_temp.R
import com.hfad.finalproject_team_temp.ui.dashboard.DashboardViewModel
class TakeQuiz : AppCompatActivity() {
private lateinit var quizId: String
private lateinit var dashboardViewModel: DashboardViewModel
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_take_quiz)
quizId = intent.getStringExtra("quizId") ?: ""
dashboardViewModel = ViewModelProvider(this).get(DashboardViewModel::class.java)
val backButton = findViewById<Button>(R.id.bBack)
backButton.setOnClickListener {
onBackPressed()
}
val savedUsername = dashboardViewModel.loadUsername(this) // obtain username of current
val nextButton = findViewById<Button>(R.id.bNext)



```
        val nextButton = findViewById<Button>(R.id.bNext)
        val db = FirebaseFirestore.getInstance()
        val questionList = mutableListOf<String>()
        val answerList = mutableListOf<String>()
        val title: TextView = findViewById(R.id.txtQuizTitle)
        val question: TextView = findViewById(R.id.txtQuestion)
        val answerVisibility: TextView = findViewById(R.id.txtAnswer)

        var incrementer = 0;
        var totalCorrect = 0;

        answerVisibility.visibility = View.INVISIBLE

        db.collection("quizzes")
            .whereEqualTo("title", quizId)
            .get()
            .addOnCompleteListener {
                    for (document in it.result!!) {
                        val questions = document["questions"] as? List<String>
                        val answers = document["answers"] as? List<String>

                        questions?.let { questionList.addAll(it) }
                        answers?.let { answerList.addAll(it) }
                        val questionSize = questionList.size
                        nextButton.setOnClickListener {
                            if (incrementer < questionSize) {
                                answerVisibility.visibility = View.VISIBLE
                                var questionNumber = incrementer + 1
                                var helper1 = questionList[incrementer]
                                question.text = "Question $questionNumber/$questionSize: $helper1"
```

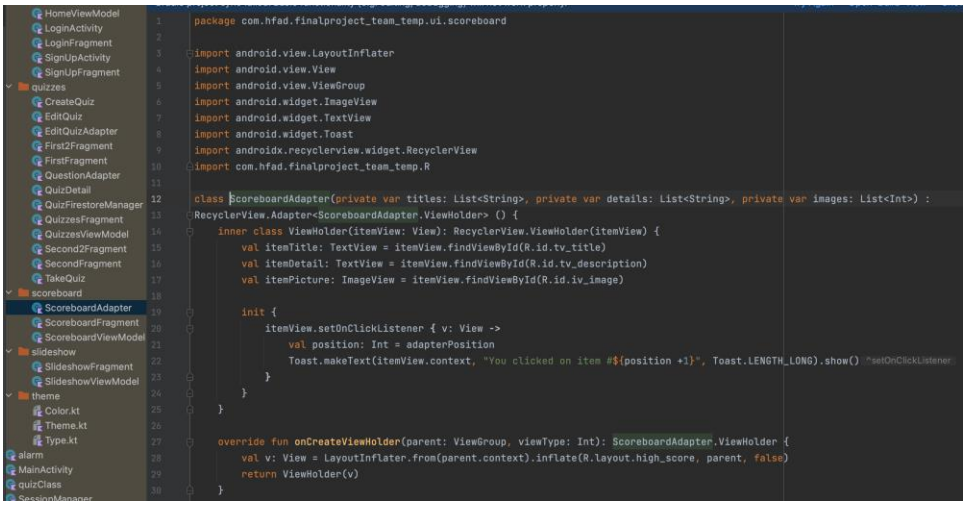


```
                                var answer = findViewById<EditText>(R.id.txtAnswer).text.toString()
                                if(incrementer > 0) {
                                    if (answer == answerList[incrementer - 1]) {
                                        totalCorrect++
                                    }
                                }
                                incrementer++
                                findViewById<EditText>(R.id.txtAnswer).setText("") ^setOnClickListener
                            }
                            else {
                                var answer = findViewById<EditText>(R.id.txtAnswer).text.toString()
                                if(answer == answerList[incrementer - 1]) totalCorrect++
                                findViewById<EditText>(R.id.txtAnswer).setText("")
                                Toast.makeText(this, "You got a $totalCorrect/$questionSize!", Toast.LENGTH_LONG).show()
                                question.visibility = View.GONE
                                answerVisibility.visibility = View.GONE
                                nextButton.visibility = View.GONE
                                backButton.text = "Finish"

                                val correct = totalCorrect.toDouble()
                                val numQuestions = questionSize.toDouble()
                                val score = correct / numQuestions
                                val scoreEntry: MutableMap<String, Any> = HashMap()
                                scoreEntry["Score"] = score
                                scoreEntry["quizID"] = quizId
                                scoreEntry["username"] = savedUsername ?: "default"
                                db.collection("quizScores")
                                    .add(scoreEntry) ^setOnClickListener
                            }
                        }
```

```
                                val score = correct / numQuestions
                                val scoreEntry: MutableMap<String, Any> = HashMap()
                                scoreEntry["Score"] = score
                                scoreEntry["quizID"] = quizId
                                scoreEntry["username"] = savedUsername ?: "default"
                                db.collection("quizScores")
                                    .add(scoreEntry) ^setOnClickListener
                            }
                        }
                    }
                }
            title.text = quizId

    }
}
```

Scoreboard

- ScoreboardAdapter



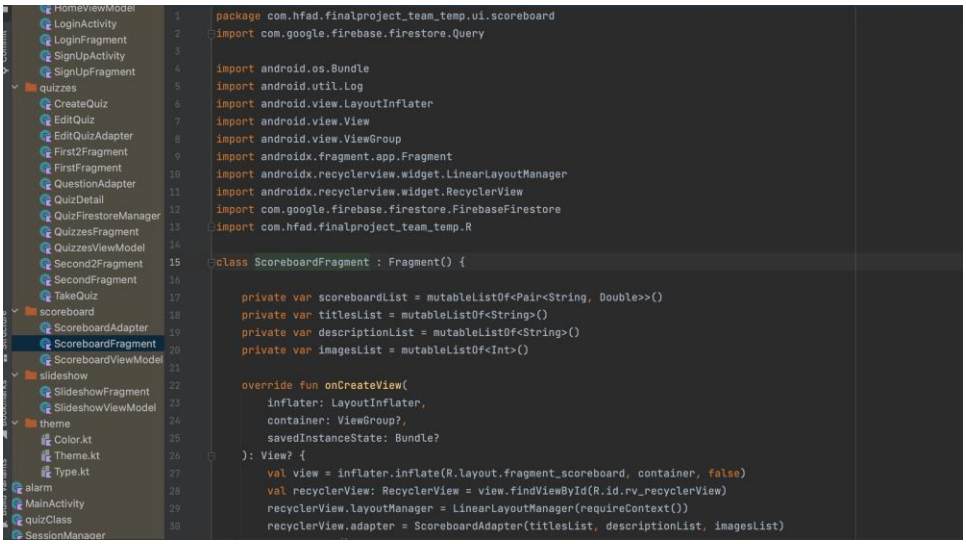
```
HomeViewModel
LoginActivity
LoginFragment
SignUpActivity
SignUpFragment
quizzes
CreateQuiz
EditQuiz
EditQuizAdapter
First2Fragment
FirstFragment
QuestionAdapter
QuizDetail
QuizFirestoreManager
QuizzesFragment
QuizzesViewModel
Second2Fragment
SecondFragment
TakeQuiz
scoreboard
ScoreboardAdapter
ScoreboardFragment
ScoreboardViewModel
slideshow
SlideshowFragment
SlideshowViewModel
theme
Color.kt
Theme.kt
Type.kt
alarm
MainActivity
quizClass
SessionManager
```

```
package com.hfad.finalproject_team_temp.ui.scoreboard

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.recyclerview.widget.RecyclerView
import com.hfad.finalproject_team_temp.R

class ScoreboardAdapter(private var titles: List<String>, private var details: List<String>, private var images: List<Int>) :
RecyclerView.Adapter<ScoreboardAdapter.ViewHolder> () {
    inner class ViewHolder(itemView: View): RecyclerView.ViewHolder(itemView) {
        val itemTitle: TextView = itemView.findViewById(R.id.tv_title)
        val itemDetail: TextView = itemView.findViewById(R.id.tv_description)
        val itemPicture: ImageView = itemView.findViewById(R.id.iv_image)

        init {
            itemView.setOnClickListener { v: View ->
                val position: Int = adapterPosition
                Toast.makeText(itemView.context, "You clicked on item #${position +1}", Toast.LENGTH_LONG).show() ^setOnClickListener
            }
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ScoreboardAdapter.ViewHolder {
        val v: View = LayoutInflater.from(parent.context).inflate(R.layout.high_score, parent, false)
        return ViewHolder(v)
    }
```

```
SecondFragment
TakeQuiz
scoreboard
ScoreboardAdapter
ScoreboardFragment
ScoreboardViewModel
slideshow
SlideshowFragment
SlideshowViewModel
theme
Color.kt
Theme.kt
```

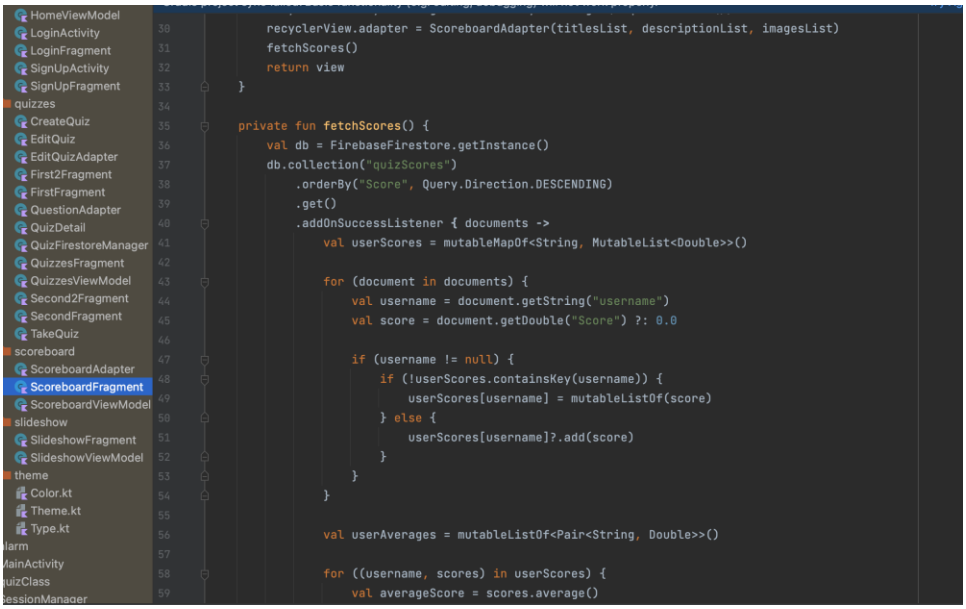
```
    override fun onBindViewHolder(holder: ScoreboardAdapter.ViewHolder, position: Int) {
        holder.itemTitle.text = titles[position]
        holder.itemDetail.text = details[position]
        holder.itemPicture.setImageResource(images[position])
    }

    override fun getItemCount(): Int {
        return titles.size
    }
}
```

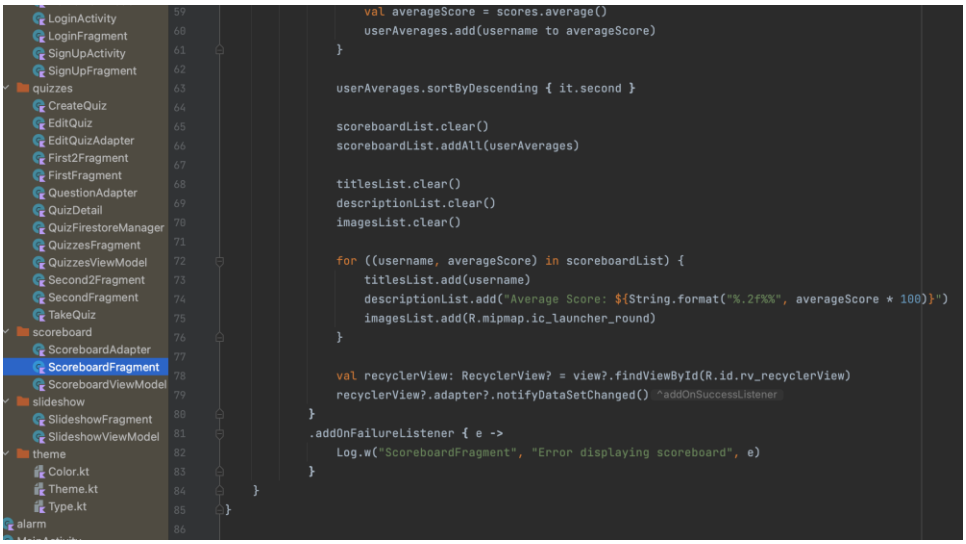
- ScoreboardFragment



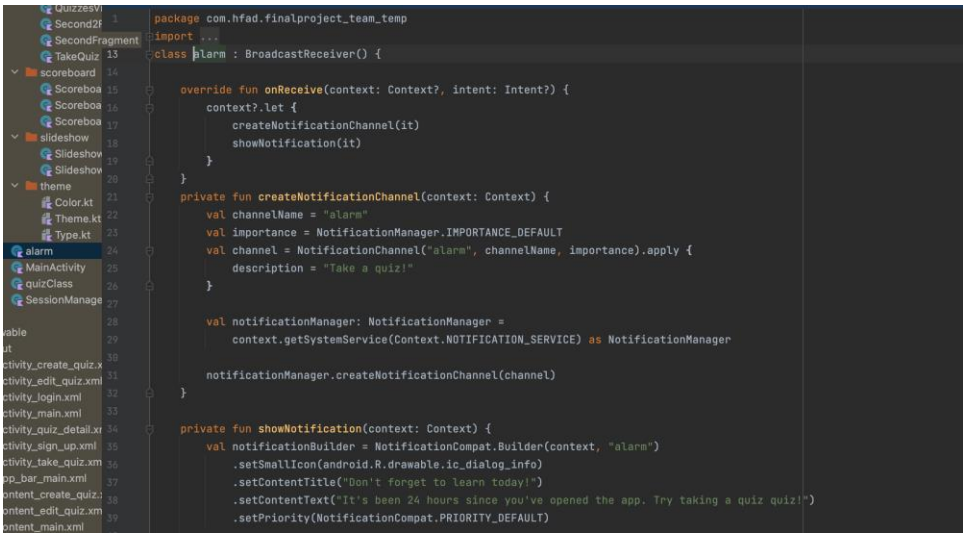
```
HomeViewModel
LoginActivity
LoginFragment
SignUpActivity
SignUpFragment
quizzes
CreateQuiz
EditQuiz
EditQuizAdapter
First2Fragment
FirstFragment
QuestionAdapter
QuizDetail
QuizFirestoreManager
QuizzesFragment
QuizzesViewModel
Second2Fragment
SecondFragment
TakeQuiz
scoreboard
ScoreboardAdapter
ScoreboardFragment
ScoreboardViewModel
slideshow
SlideshowFragment
SlideshowViewModel
theme
Color.kt
Theme.kt
Type.kt
alarm
MainActivity
quizClass
SessionManager
```

```
package com.hfad.finalproject_team_temp.ui.scoreboard
import com.google.firebase.firestore.Query

import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.firebase.firestore.FirebaseFirestore
import com.hfad.finalproject_team_temp.R

class ScoreboardFragment : Fragment() {

    private var scoreboardList = mutableListOf<Pair<String, Double>>()
    private var titlesList = mutableListOf<String>()
    private var descriptionList = mutableListOf<String>()
    private var imagesList = mutableListOf<Int>()

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_scoreboard, container, false)
        val recyclerView: RecyclerView = view.findViewById(R.id.rv_recyclerView)
        recyclerView.layoutManager = LinearLayoutManager(requireContext())
        recyclerView.adapter = ScoreboardAdapter(titlesList, descriptionList, imagesList)
```

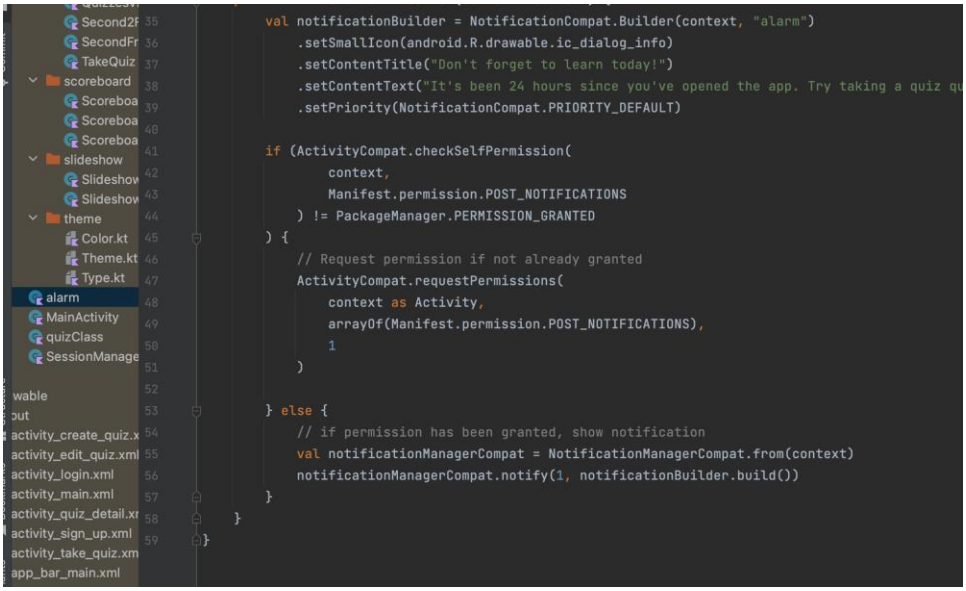

HomeViewModel
LoginActivity
LoginFragment
SignUpActivity
SignUpFragment
quizzes
CreateQuiz
EditQuiz
EditQuizAdapter
First2Fragment
FirstFragment
QuestionAdapter
QuizDetail
QuizFirestoreManager
QuizzesFragment
QuizzesViewModel
Second2Fragment
SecondFragment
TakeQuiz
scoreboard
ScoreboardAdapter
ScoreboardFragment
ScoreboardViewModel
slideshow
SlideshowFragment
SlideshowViewModel
theme
Color.kt
Theme.kt
Type.kt
alarm
MainActivity
QuizClass
SessionManager
recyclerView.adapter = ScoreboardAdapter(titlesList, descriptionList, imagesList)
fetchScores()
return view
}
private fun fetchScores() {
val db = FirebaseFirestore.getInstance()
db.collection("quizScores")
.orderBy("Score", Query.Direction.DESCENDING)
.get()
.addOnSuccessListener { documents ->
val userScores = mutableMapOf<String, MutableList<Double>>()
for (document in documents) {
val username = document.getString("username")
val score = document.getDouble("Score") ?: 0.0
if (username != null) {
if (!userScores.containsKey(username)) {
userScores[username] = mutableListOf(score)
} else {
userScores[username]?.add(score)
}
}
}
val userAverages = mutableListOf<Pair<String, Double>>()
for ((username, scores) in userScores) {
val averageScore = scores.average()


LoginActivity
LoginFragment
SignUpActivity
SignUpFragment
quizzes
CreateQuiz
EditQuiz
EditQuizAdapter
First2Fragment
FirstFragment
QuestionAdapter
QuizDetail
QuizFirestoreManager
QuizzesFragment
QuizzesViewModel
Second2Fragment
SecondFragment
TakeQuiz
scoreboard
ScoreboardAdapter
ScoreboardFragment
ScoreboardViewModel
slideshow
SlideshowFragment
SlideshowViewModel
theme
Color.kt
Theme.kt
Type.kt
alarm
val averageScore = scores.average()
userAverages.add(username to averageScore)
}
userAverages.sortByDescending { it.second }
scoreboardList.clear()
scoreboardList.addAll(userAverages)
titlesList.clear()
descriptionList.clear()
imagesList.clear()
for ((username, averageScore) in scoreboardList) {
titlesList.add(username)
descriptionList.add("Average Score: ${String.format("%.2f%%", averageScore * 100)}")
imagesList.add(R.mipmap.ic_launcher_round)
}
val recyclerView: RecyclerView? = view?.findViewById(R.id.rv_recyclerView)
recyclerView?.adapter?.notifyDataSetChanged()
}
.addOnFailureListener { e ->
Log.w("ScoreboardFragment", "Error displaying scoreboard", e)
}
}
}

- Alarm

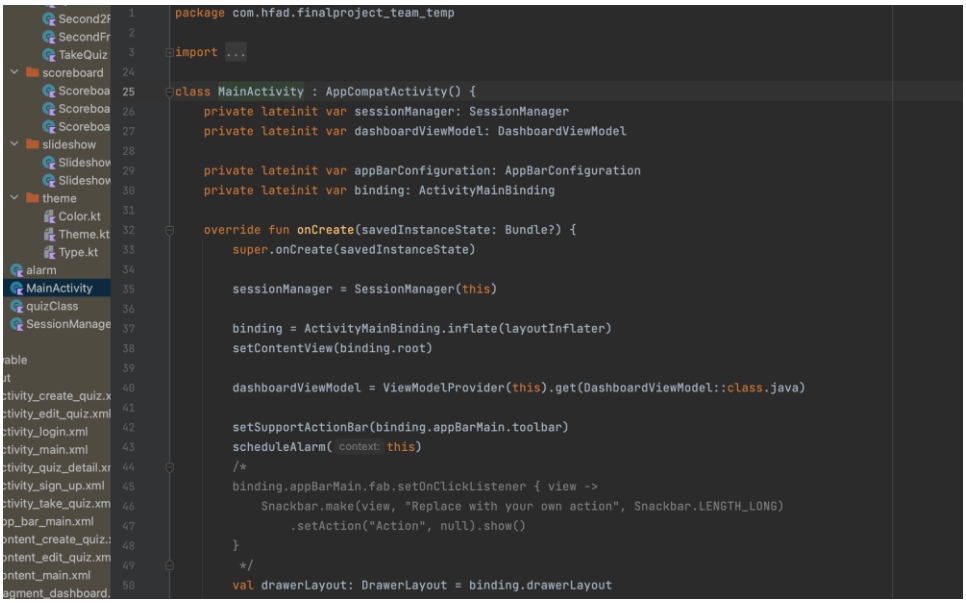


```kotlin
package com.hfad.finalproject_team_temp
import ...
class alarm : BroadcastReceiver() {

    override fun onReceive(context: Context?, intent: Intent?) {
        context?.let {
            createNotificationChannel(it)
            showNotification(it)
        }
    }
    private fun createNotificationChannel(context: Context) {
        val channelName = "alarm"
        val importance = NotificationManager.IMPORTANCE_DEFAULT
        val channel = NotificationChannel("alarm", channelName, importance).apply {
            description = "Take a quiz!"
        }

        val notificationManager: NotificationManager =
            context.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

        notificationManager.createNotificationChannel(channel)
    }

    private fun showNotification(context: Context) {
        val notificationBuilder = NotificationCompat.Builder(context, "alarm")
            .setSmallIcon(android.R.drawable.ic_dialog_info)
            .setContentTitle("Don't forget to learn today!")
            .setContentText("It's been 24 hours since you've opened the app. Try taking a quiz quiz!")
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)
```

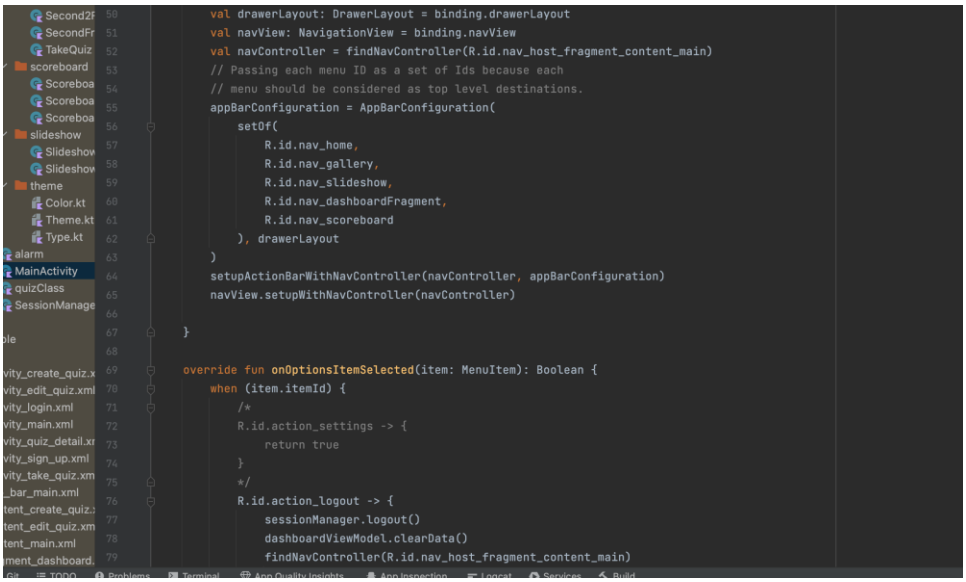


```kotlin
        val notificationBuilder = NotificationCompat.Builder(context, "alarm")
            .setSmallIcon(android.R.drawable.ic_dialog_info)
            .setContentTitle("Don't forget to learn today!")
            .setContentText("It's been 24 hours since you've opened the app. Try taking a quiz qu
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)

        if (ActivityCompat.checkSelfPermission(
                context,
                Manifest.permission.POST_NOTIFICATIONS
            ) != PackageManager.PERMISSION_GRANTED
        ) {
            // Request permission if not already granted
            ActivityCompat.requestPermissions(
                context as Activity,
                arrayOf(Manifest.permission.POST_NOTIFICATIONS),
                1
            )

        } else {
            // if permission has been granted, show notification
            val notificationManagerCompat = NotificationManagerCompat.from(context)
            notificationManagerCompat.notify(1, notificationBuilder.build())
        }
    }
}
```

MainActivity

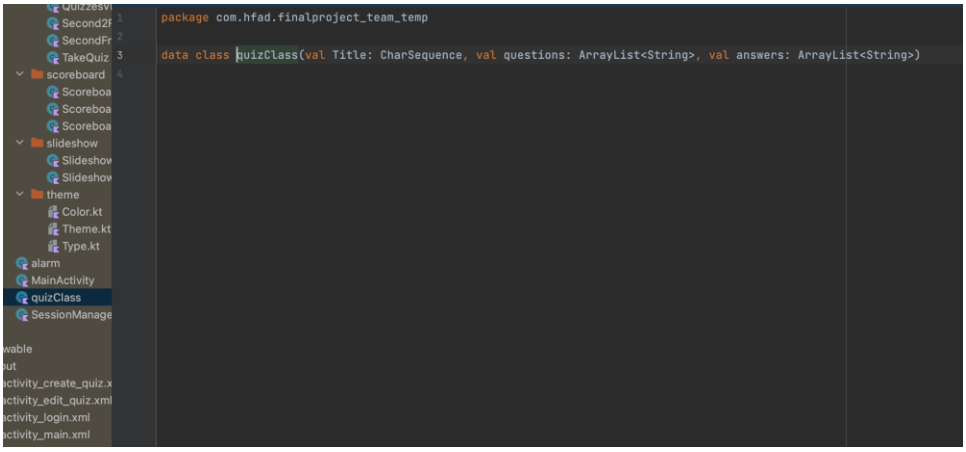


```
package com.hfad.finalproject_team_temp

import ...

class MainActivity : AppCompatActivity() {
    private lateinit var sessionManager: SessionManager
    private lateinit var dashboardViewModel: DashboardViewModel

    private lateinit var appBarConfiguration: AppBarConfiguration
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        sessionManager = SessionManager(this)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        dashboardViewModel = ViewModelProvider(this).get(DashboardViewModel::class.java)

        setSupportActionBar(binding.appBarMain.toolbar)
        scheduleAlarm( context: this)
        /*
        binding.appBarMain.fab.setOnClickListener { view ->
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction("Action", null).show()
        }
         */
        val drawerLayout: DrawerLayout = binding.drawerLayout
```

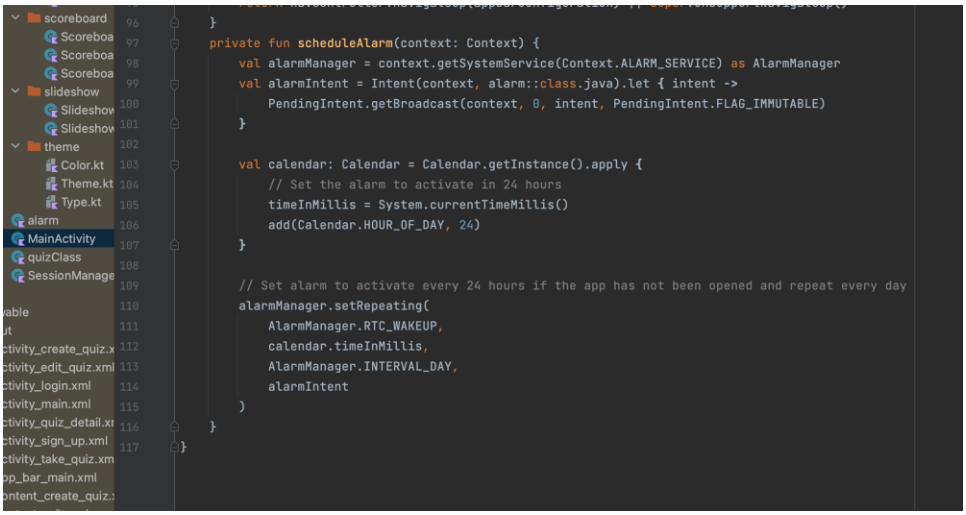


```
        val drawerLayout: DrawerLayout = binding.drawerLayout
        val navView: NavigationView = binding.navView
        val navController = findNavController(R.id.nav_host_fragment_content_main)
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        appBarConfiguration = AppBarConfiguration(
            setOf(
                R.id.nav_home,
                R.id.nav_gallery,
                R.id.nav_slideshow,
                R.id.nav_dashboardFragment,
                R.id.nav_scoreboard
            ), drawerLayout
        )
        setupActionBarWithNavController(navController, appBarConfiguration)
        navView.setupWithNavController(navController)

    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        when (item.itemId) {
            /*
            R.id.action_settings -> {
                return true
            }
            */
            R.id.action_logout -> {
                sessionManager.logout()
                dashboardViewModel.clearData()
                findNavController(R.id.nav_host_fragment_content_main)
```

QuizClass



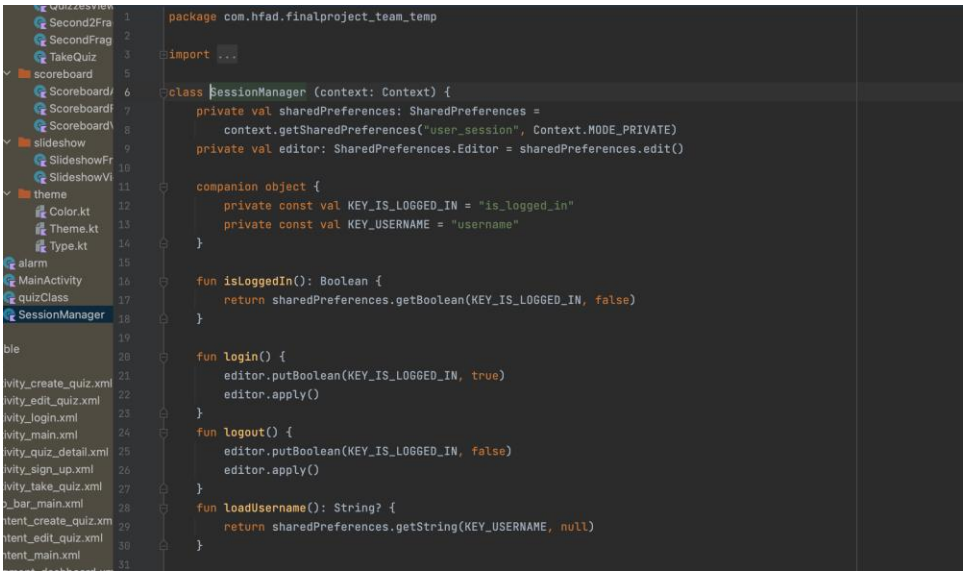
```
package com.hfad.finalproject_team_temp

data class quizClass(val Title: CharSequence, val questions: ArrayList<String>, val answers: ArrayList<String>)
```



```
    }
    private fun scheduleAlarm(context: Context) {
        val alarmManager = context.getSystemService(Context.ALARM_SERVICE) as AlarmManager
        val alarmIntent = Intent(context, alarm::class.java).let { intent ->
            PendingIntent.getBroadcast(context, 0, intent, PendingIntent.FLAG_IMMUTABLE)
        }

        val calendar: Calendar = Calendar.getInstance().apply {
            // Set the alarm to activate in 24 hours
            timeInMillis = System.currentTimeMillis()
            add(Calendar.HOUR_OF_DAY, 24)
        }

        // Set alarm to activate every 24 hours if the app has not been opened and repeat every day
        alarmManager.setRepeating(
            AlarmManager.RTC_WAKEUP,
            calendar.timeInMillis,
            AlarmManager.INTERVAL_DAY,
            alarmIntent
        )
    }
}
```

```
                findNavController(R.id.nav_host_fragment_content_main)
                    .navigate(R.id.action_nav_home_to_HomeFragment)
                return true
            }
            else -> return super.onOptionsItemSelected(item)
        }
    }

    override fun onCreateOptionsMenu(menu: Menu): Boolean {
        // Inflate the menu; this adds items to the action bar if it is present.
        menuInflater.inflate(R.menu.main, menu)
        return true
    }

    override fun onSupportNavigateUp(): Boolean {
        val navController = findNavController(R.id.nav_host_fragment_content_main)
        return navController.navigateUp(appBarConfiguration) || super.onSupportNavigateUp()
    }
    private fun scheduleAlarm(context: Context) {
        val alarmManager = context.getSystemService(Context.ALARM_SERVICE) as AlarmManager
        val alarmIntent = Intent(context, alarm::class.java).let { intent ->
            PendingIntent.getBroadcast(context, 0, intent, PendingIntent.FLAG_IMMUTABLE)
        }

        val calendar: Calendar = Calendar.getInstance().apply {
            // Set the alarm to activate in 24 hours
            timeInMillis = System.currentTimeMillis()
            add(Calendar.HOUR_OF_DAY, 24)
        }
```

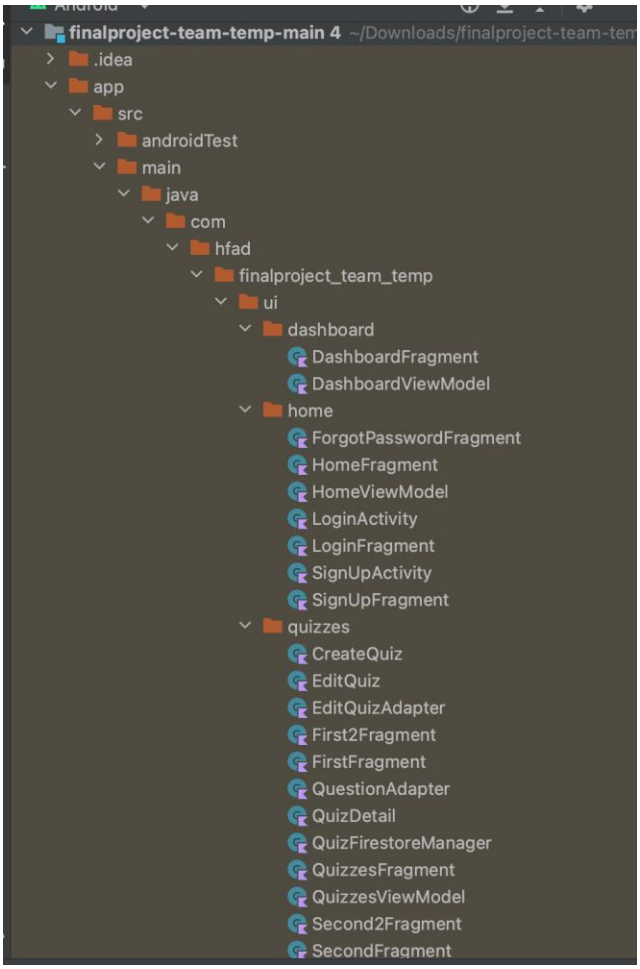
- SessionManager

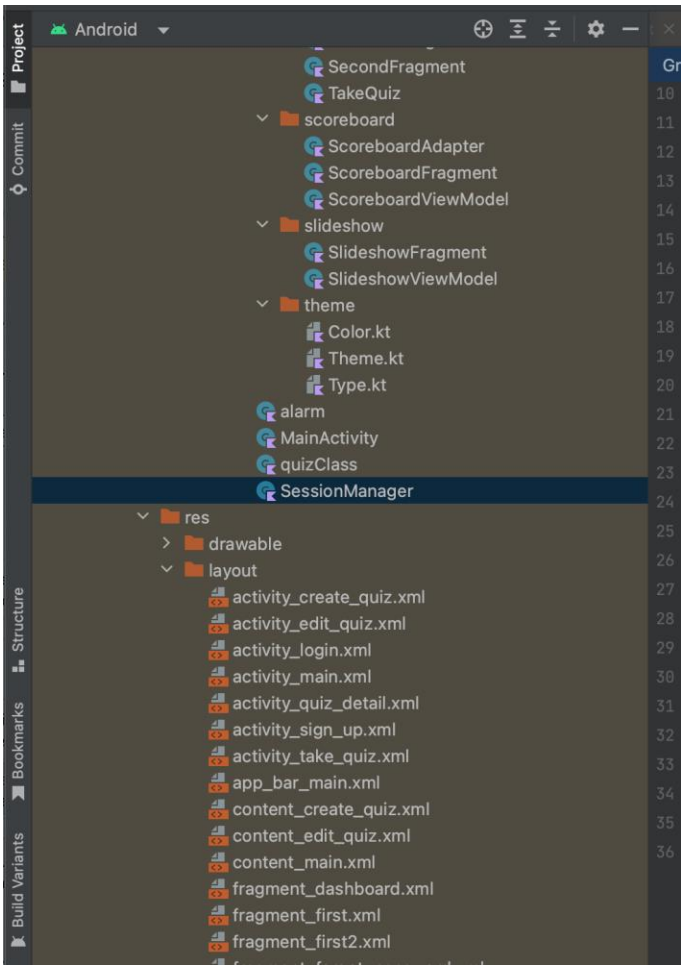


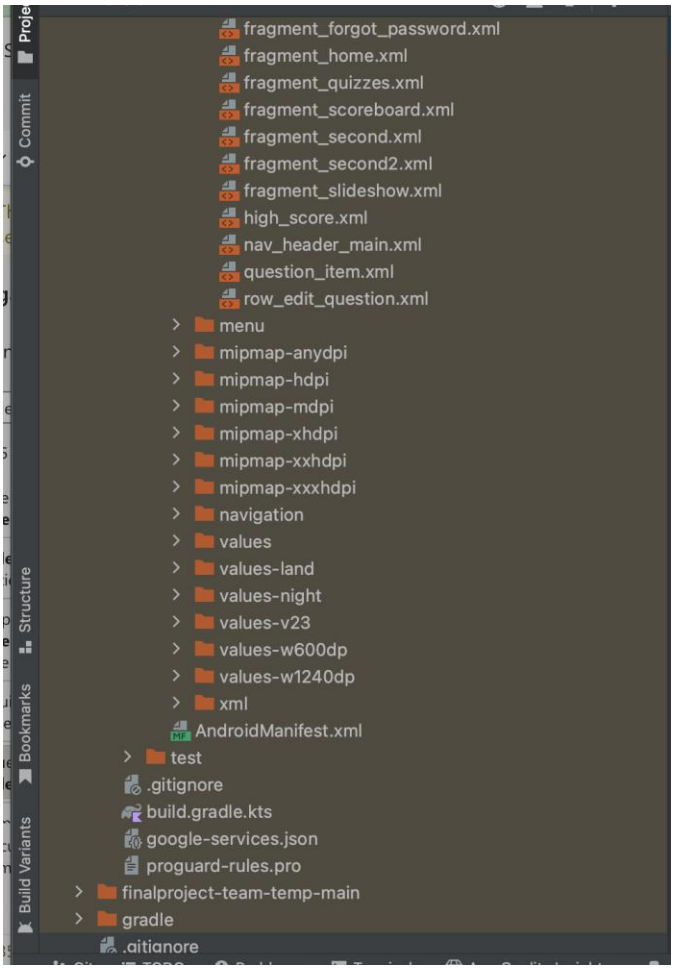
```
package com.hfad.finalproject_team_temp

import ...

class SessionManager (context: Context) {
    private val sharedPreferences: SharedPreferences =
        context.getSharedPreferences("user_session", Context.MODE_PRIVATE)
    private val editor: SharedPreferences.Editor = sharedPreferences.edit()

    companion object {
        private const val KEY_IS_LOGGED_IN = "is_logged_in"
        private const val KEY_USERNAME = "username"
    }

    fun isLoggedIn(): Boolean {
        return sharedPreferences.getBoolean(KEY_IS_LOGGED_IN, false)
    }

    fun login() {
        editor.putBoolean(KEY_IS_LOGGED_IN, true)
        editor.apply()
    }
    fun logout() {
        editor.putBoolean(KEY_IS_LOGGED_IN, false)
        editor.apply()
    }
    fun loadUsername(): String? {
        return sharedPreferences.getString(KEY_USERNAME, null)
    }
```

```
    }

    fun login() {
        editor.putBoolean(KEY_IS_LOGGED_IN, true)
        editor.apply()
    }
    fun logout() {
        editor.putBoolean(KEY_IS_LOGGED_IN, false)
        editor.apply()
    }
    fun loadUsername(): String? {
        return sharedPreferences.getString(KEY_USERNAME, null)
    }

    fun saveUsername(username: String) {
        editor.putString(KEY_USERNAME, username)
        editor.apply()
    }
}
```

Side Bar:


finalproject-team-temp-main 4 ~/Downloads/finalproject-team-tem
.idea
app
src
androidTest
main
java
com
hfad
finalproject_team_temp
ui
dashboard
DashboardFragment
DashboardViewModel
home
ForgotPasswordFragment
HomeFragment
HomeViewModel
LoginActivity
LoginFragment
SignUpActivity
SignUpFragment
quizzes
CreateQuiz
EditQuiz
EditQuizAdapter
First2Fragment
FirstFragment
QuestionAdapter
QuizDetail
QuizFirestoreManager
QuizzesFragment
QuizzesViewModel
Second2Fragment
SecondFragment

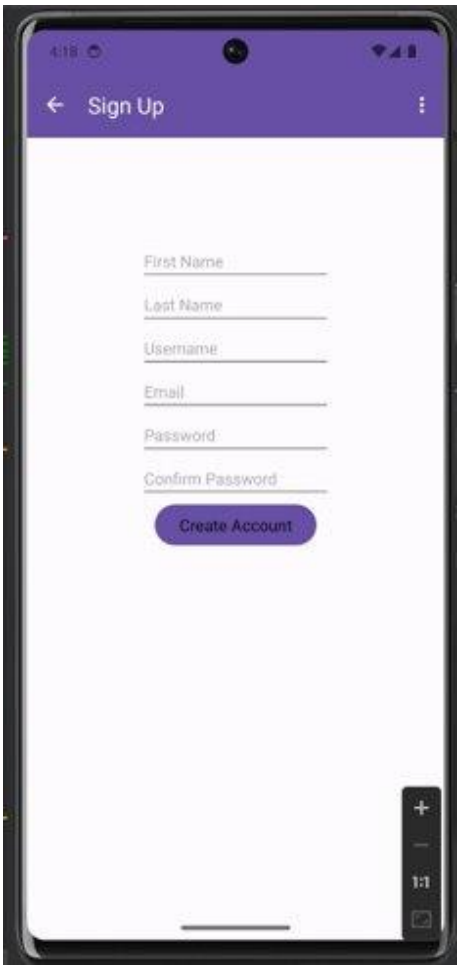

Android
Project
Commit
Structure
Bookmarks
Build Variants
SecondFragment
TakeQuiz
scoreboard
ScoreboardAdapter
ScoreboardFragment
ScoreboardViewModel
slideshow
SlideshowFragment
SlideshowViewModel
theme
Color.kt
Theme.kt
Type.kt
alarm
MainActivity
quizClass
SessionManager
res
drawable
layout
activity_create_quiz.xml
activity_edit_quiz.xml
activity_login.xml
activity_main.xml
activity_quiz_detail.xml
activity_sign_up.xml
activity_take_quiz.xml
app_bar_main.xml
content_create_quiz.xml
content_edit_quiz.xml
content_main.xml
fragment_dashboard.xml
fragment_first.xml
fragment_first2.xml

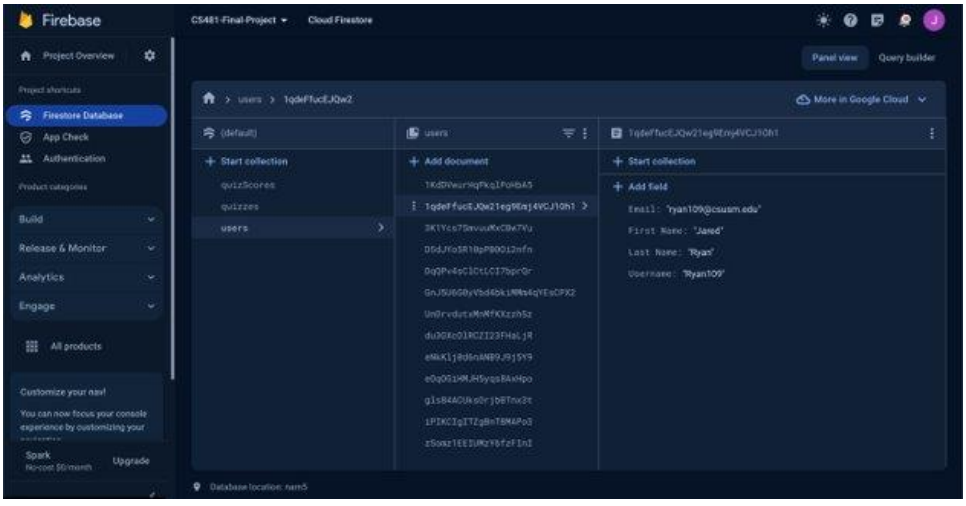


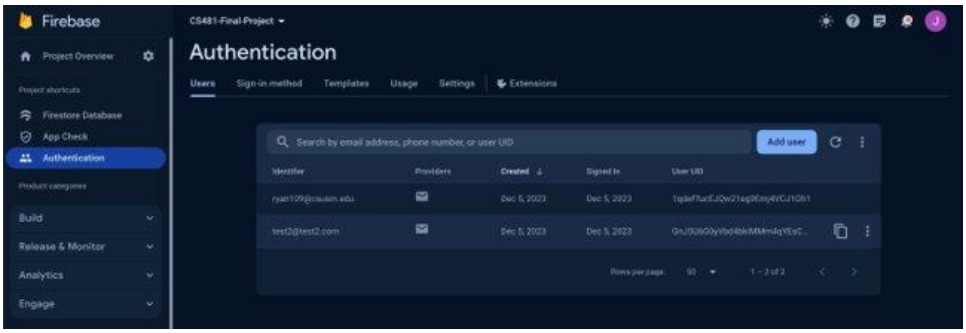
USER INTERFACE

- Login, Sign Up, and Authentication:

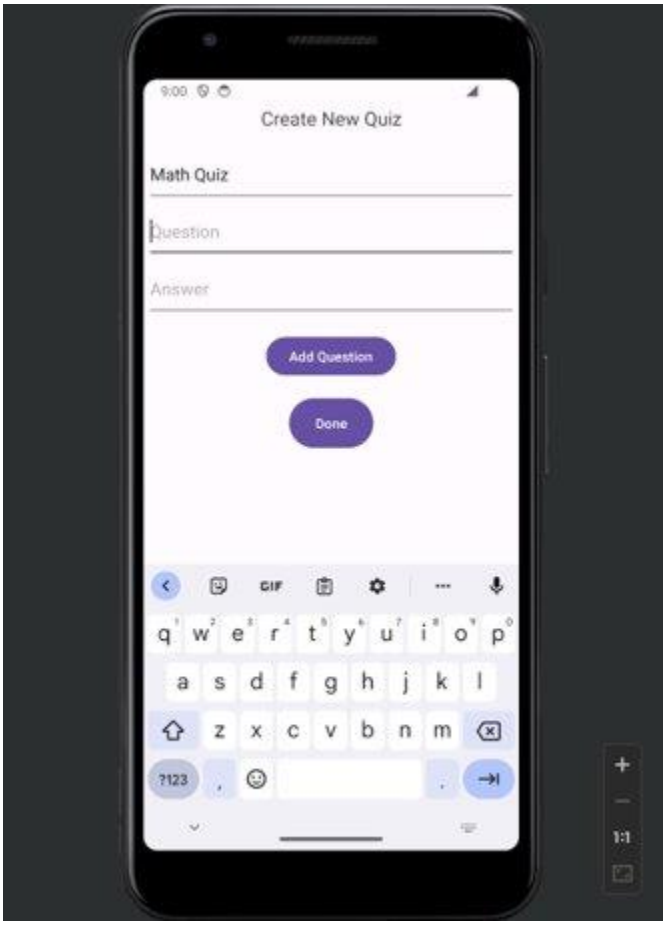

Login
Enter Email
Enter Password
LOGIN
FORGOT PASSWORD

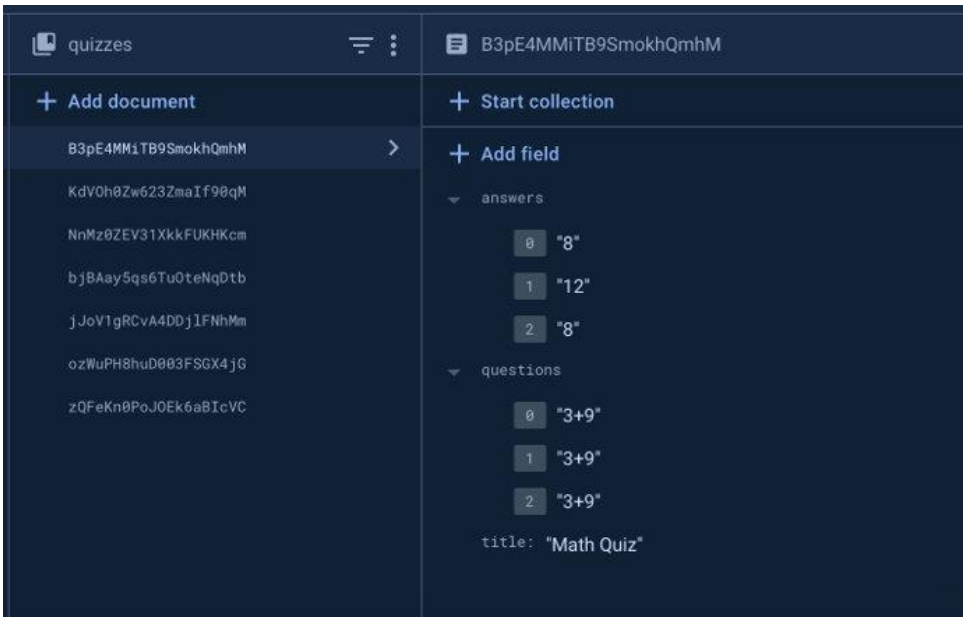

Sign Up
First Name
Last Name
Username
Email
Password
Confirm Password
Create Account


Firebase
CS481-Final-Project
Cloud Firestore
Project Overview
Firestore Database
App Check
Authentication
Build
Release & Monitor
Analytics
Engage
All products
Customize your nav!
Upgrade
Panel view
Query builder
users
1qdeFfucEJQw2
More in Google Cloud
(default)
Start collection
quizScores
quizzes
users
Add document
Add field
Email: "ryan109@csusm.edu"
First Name: "Jared"
Last Name: "Ryan"
Username: "Ryan109"

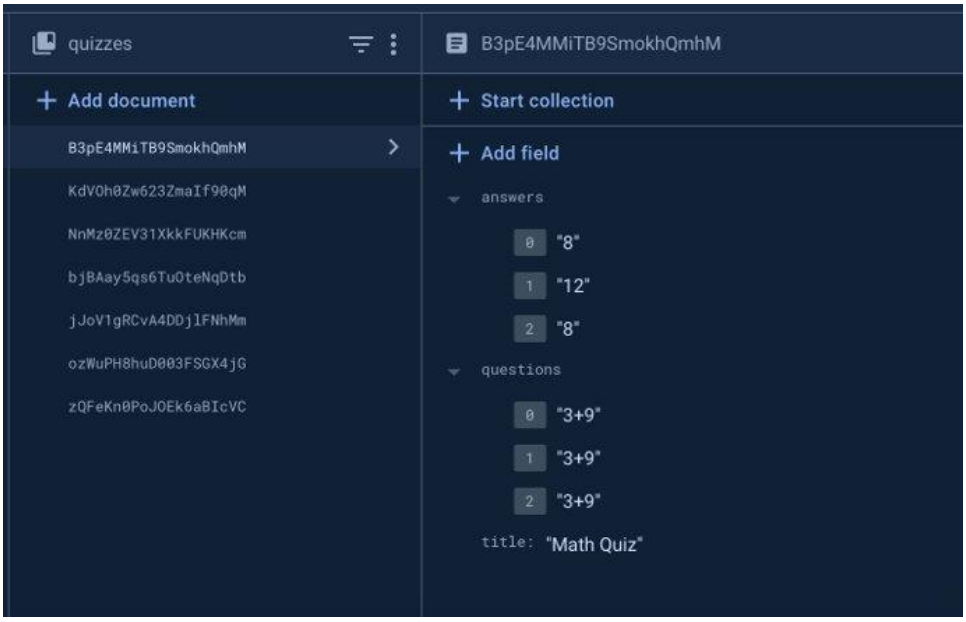

Firebase
CS481-Final-Project
Project Overview
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
Firestore Database
App Check
Authentication
Build
Release & Monitor
Analytics
Engage
Search by email address, phone number, or user UID
Add user
Identifier
Providers
Created
Signed In
User UID

- CreateQuiz

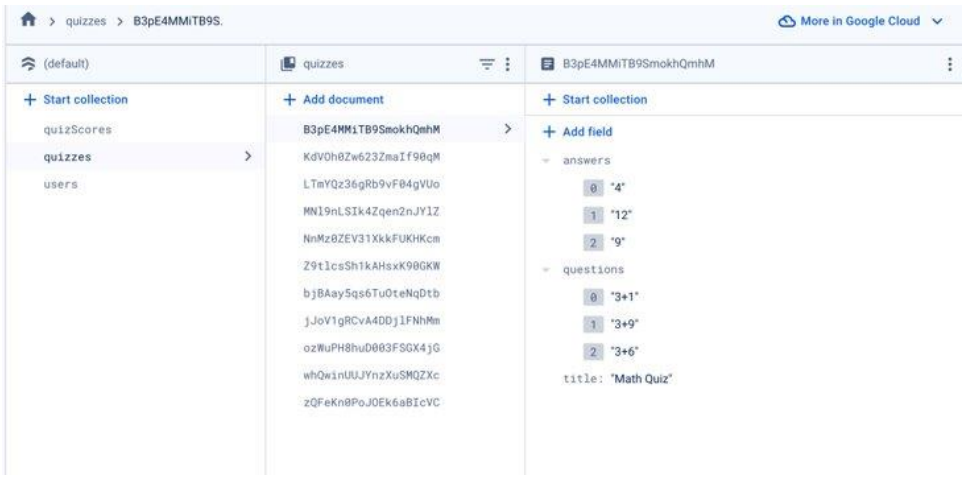

9:00
Create New Quiz
Math Quiz
Question
Answer
Add Question
Done
GIF
?123
1:1

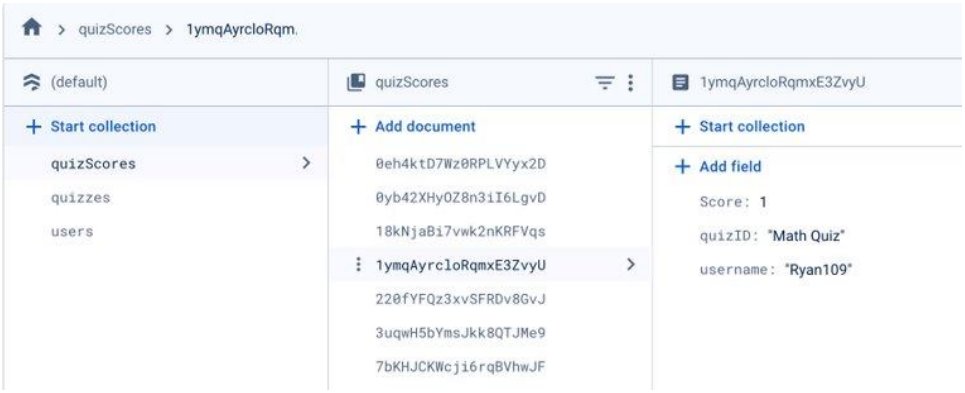


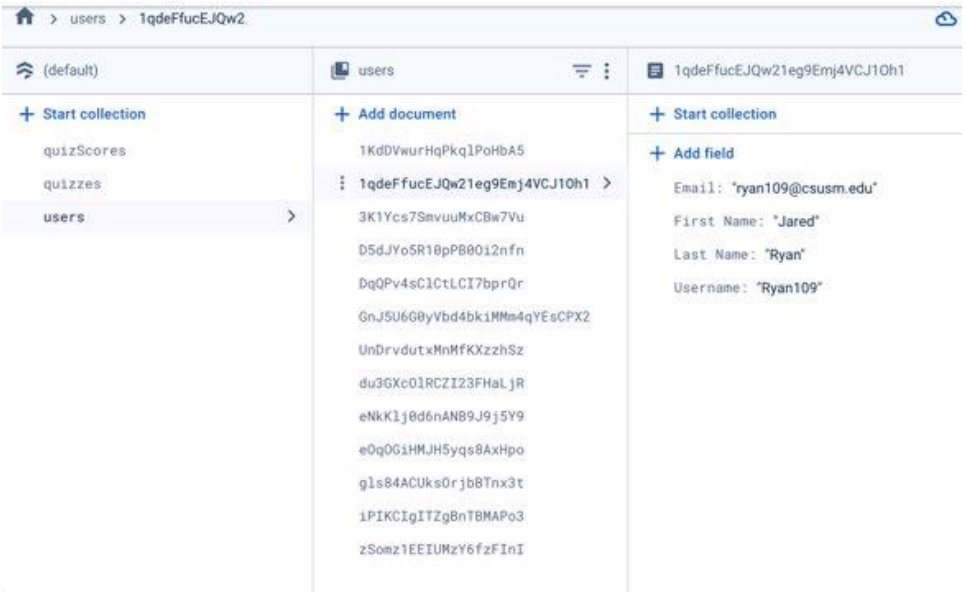
- EditQuiz



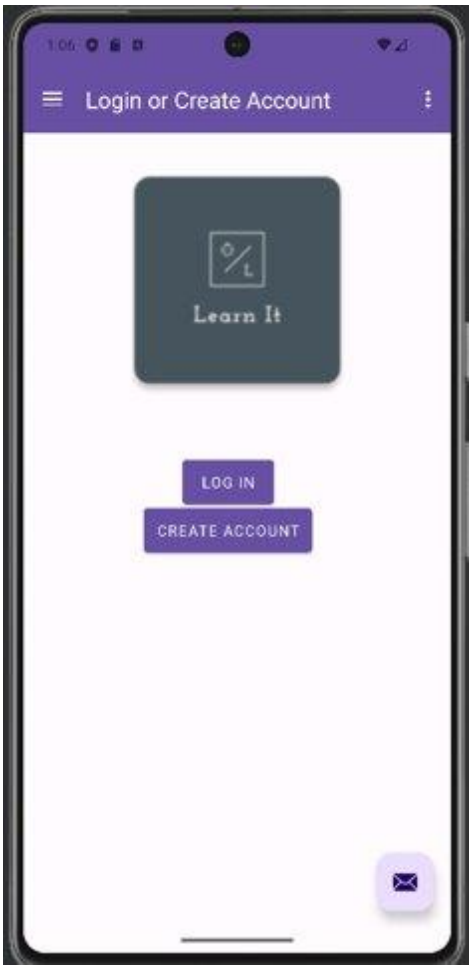
- TakeQuiz
- Scoreboard
- Firebase







- Notification
- UI Redesign


Login or Create Account
Learn It
LOG IN
CREATE ACCOUNT

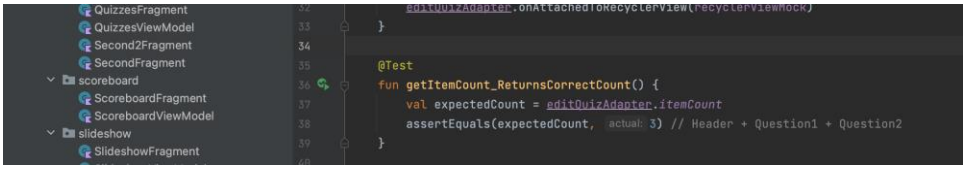

Login or Create Account
Learn It
LOG IN
CREATE ACCOUNT

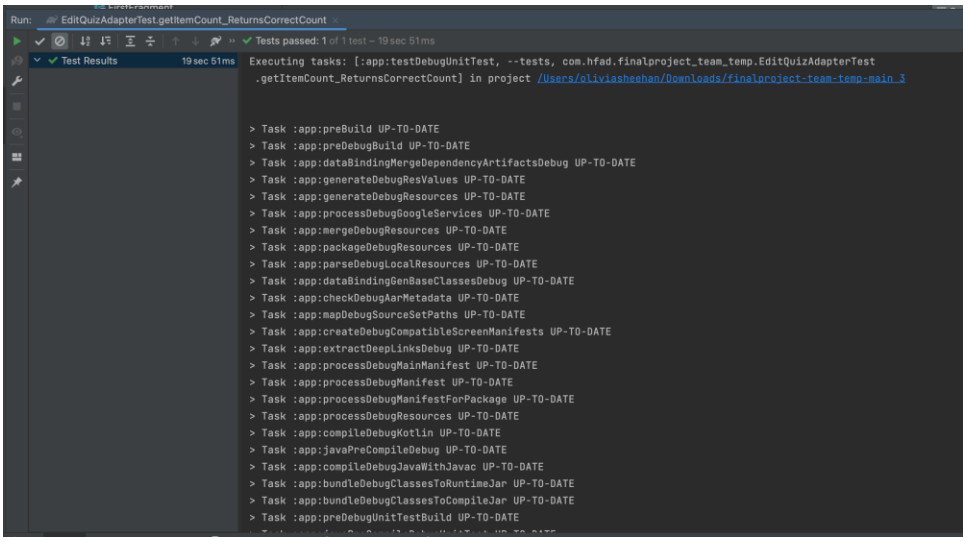
- Permission

UNIT TESTS

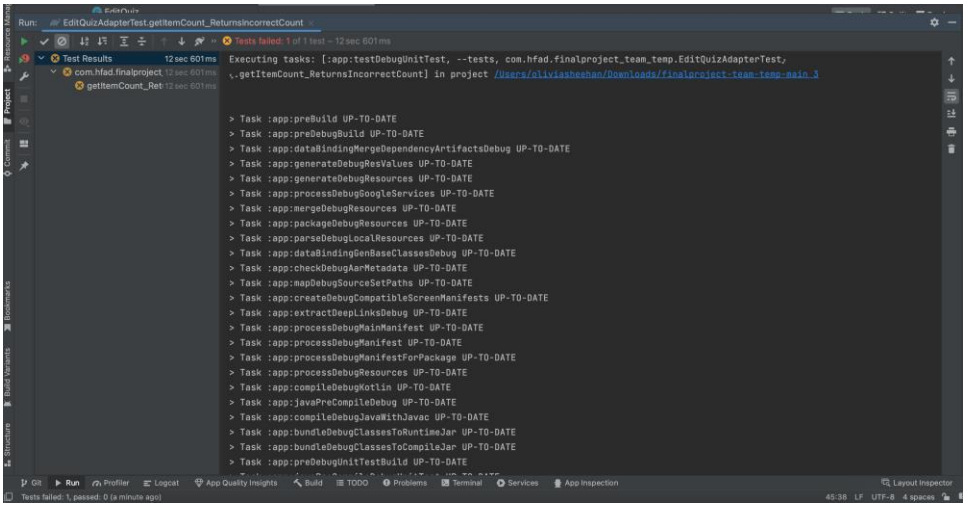
Edit Quiz Adapter Test

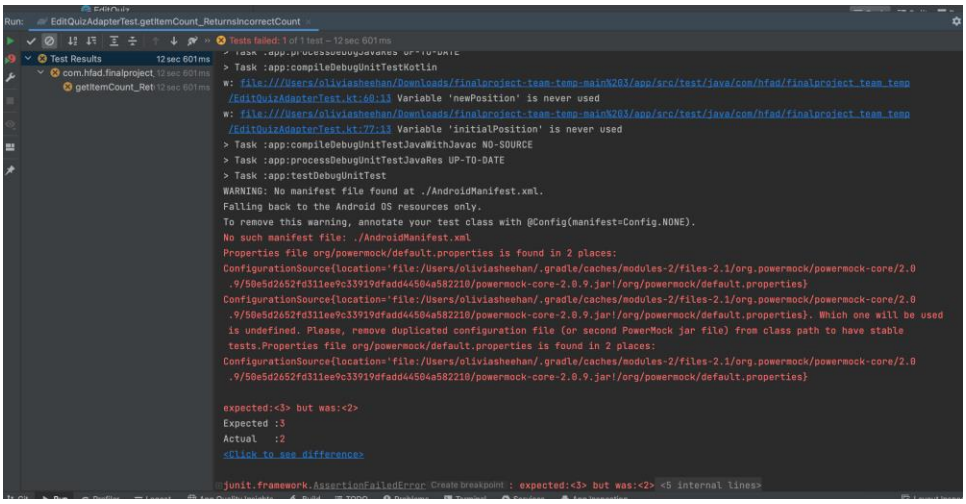
- getItemCount



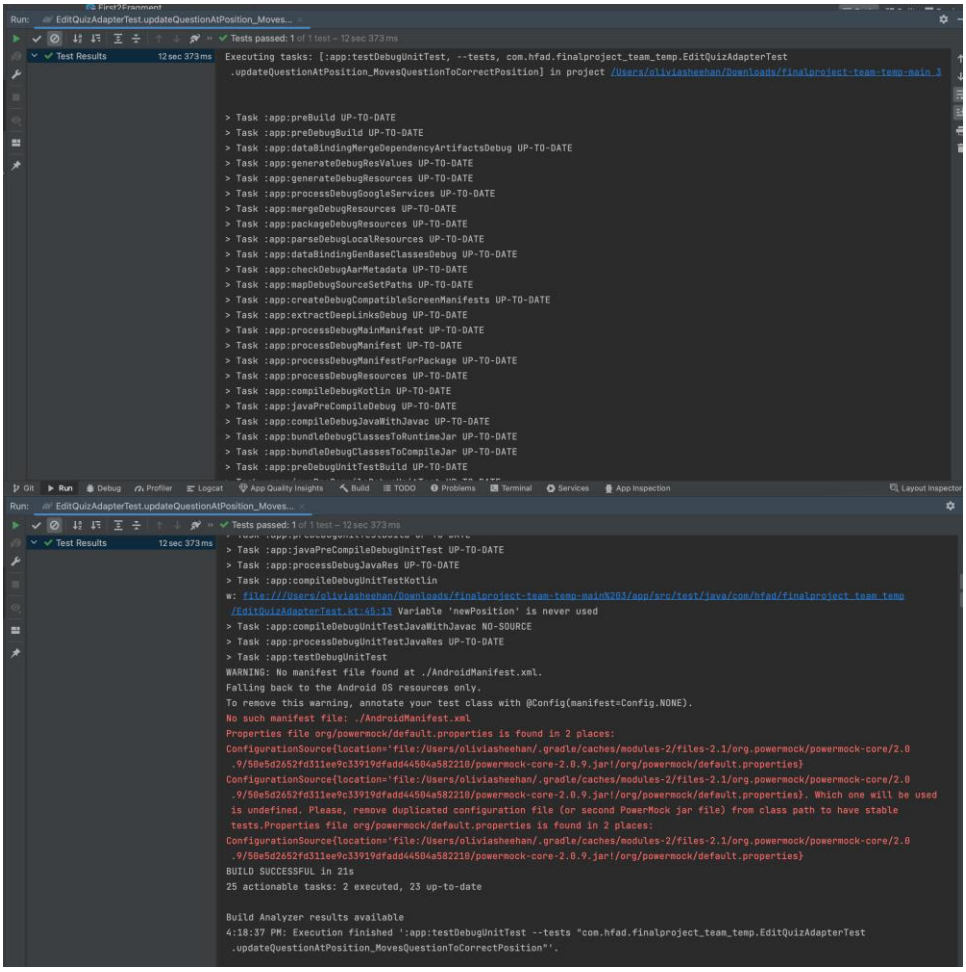


- Get Count Incorrect (Intended to Fail)

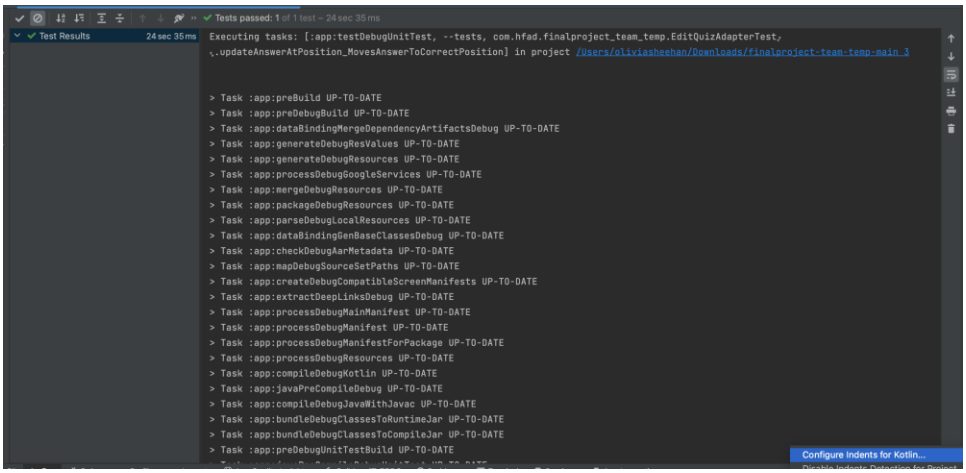


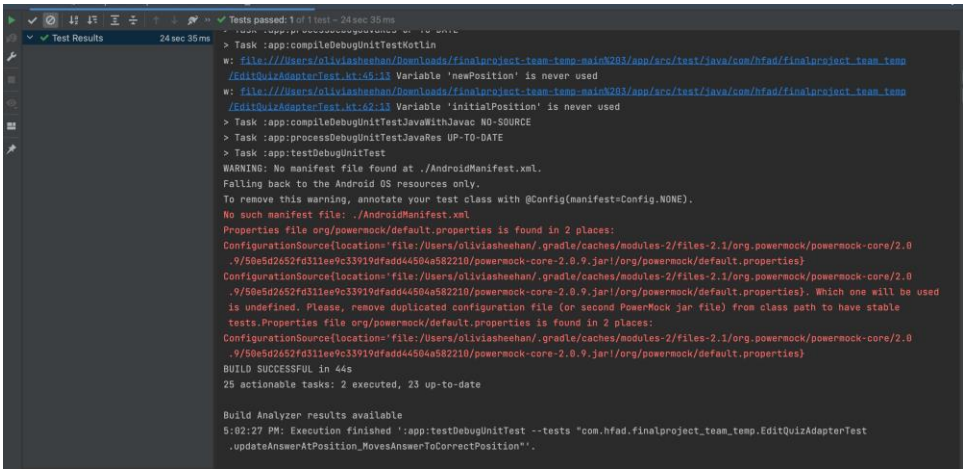


- Update Question



- Update Answer at Position

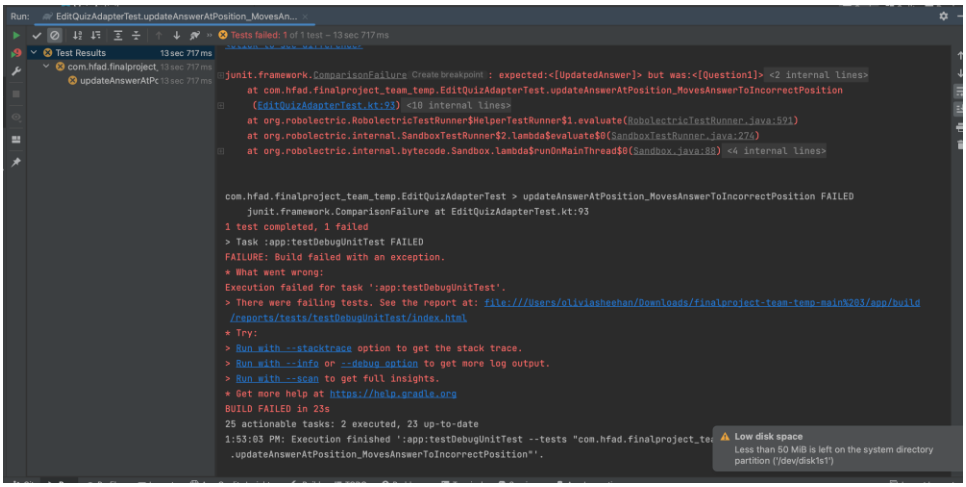


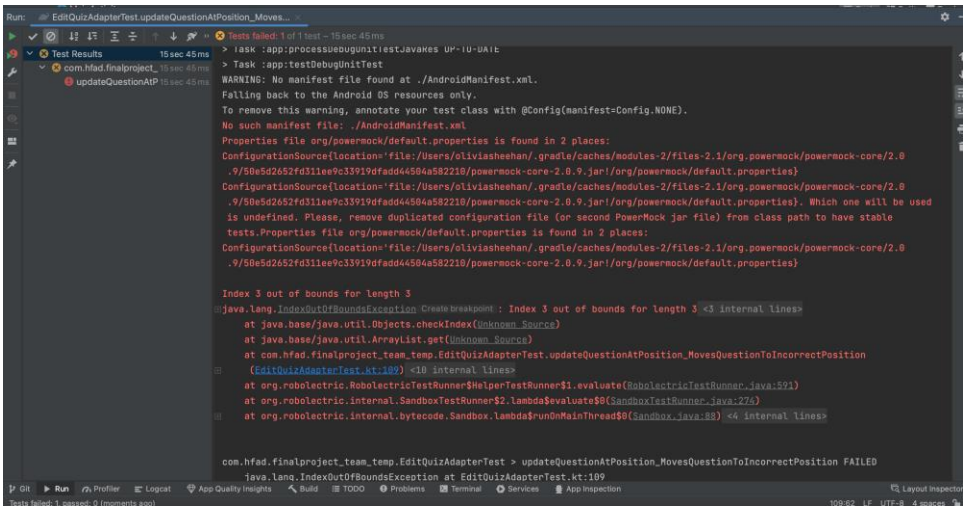


- Update Answer to Incorrect Position

```
78
79      @Test
80      fun updateAnswerAtPosition_MovesAnswerToIncorrectPosition() {
81          // Arrange
82          val newPos = 3;
83          val initialAnswer = "InitialAnswer"
84          val updatedAnswer = "UpdatedAnswer"
85
86          // Set up the initial state of the adapter
87          editQuizAdapter.questionList = mutableListOf("Question1", initialAnswer,"Question2")
88
89          //Update
90          editQuizAdapter.updateQuestionAtPosition(newPos, updatedAnswer)
91
92          // Assert
93          assertEquals(updatedAnswer, editQuizAdapter.questionList[0])
94      }
95
```

(unit test intended to fail when placing an answer beyond the scope)

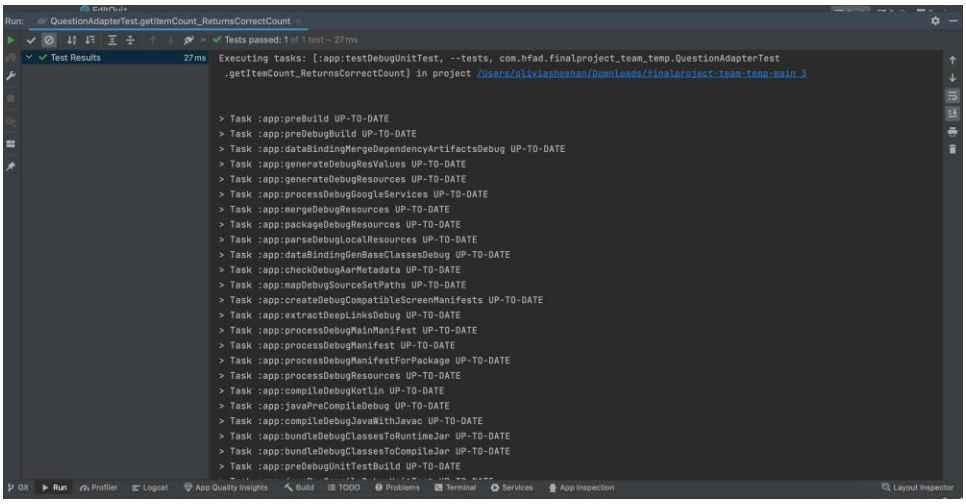


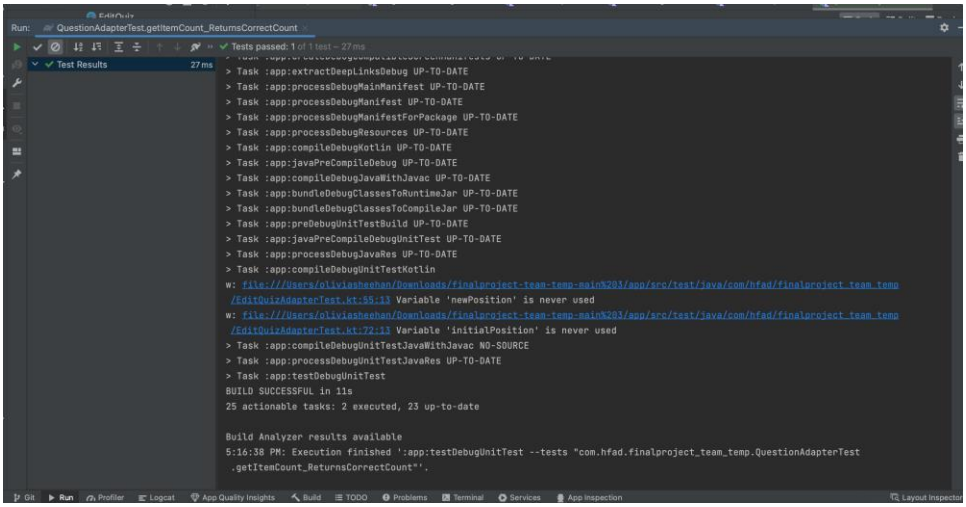


Question Adapter

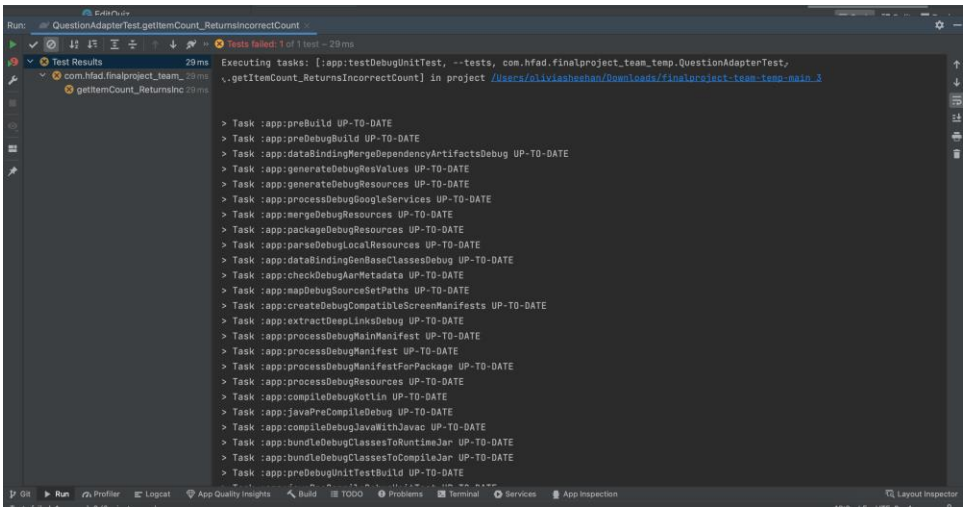
- Get Count

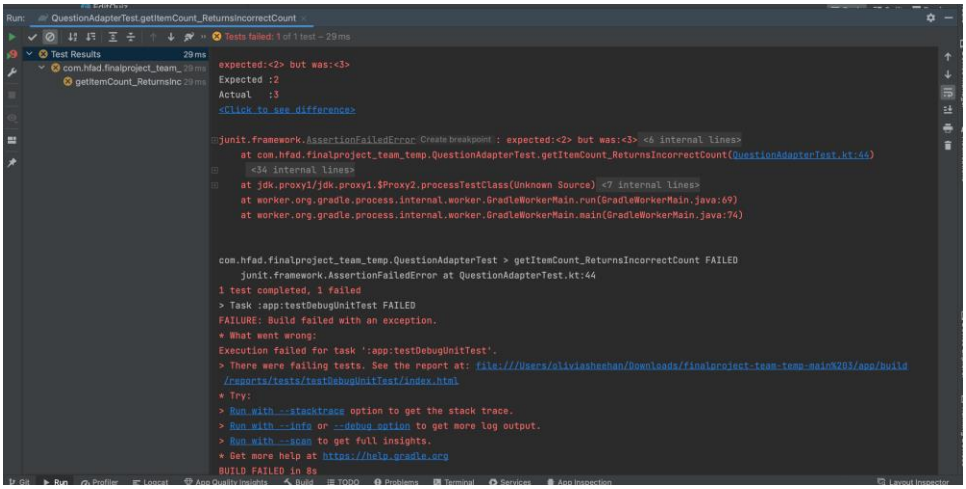
```
32          //editQuizAdapter.onAttachedToRecyclerView(recyclerViewMock)
33      }
34
35      @Test
36      fun getItemCount_ReturnsCorrectCount() {
37          val expectedCount = questionAdapter.itemCount
38          TestCase.assertEquals(expectedCount,  actual: 2) //Question1 + Question2
39      }
40
41      @Test
```



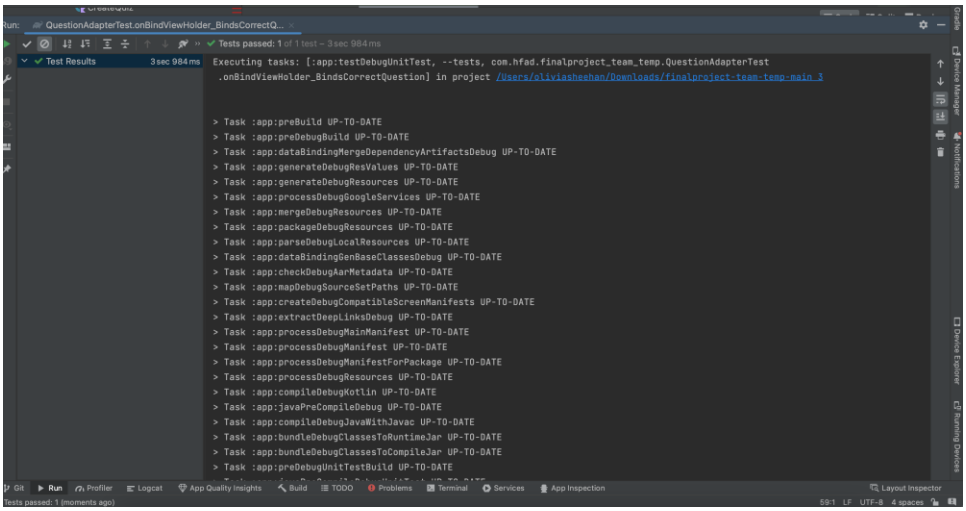


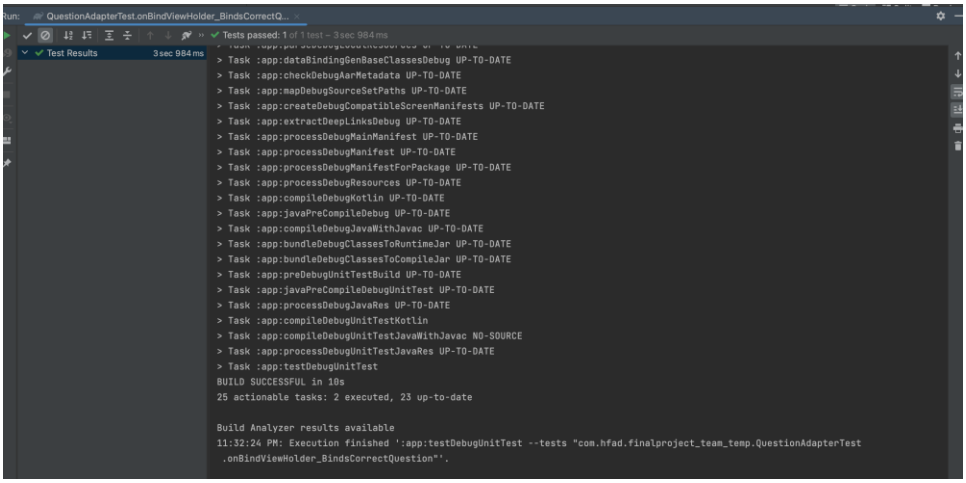
- Get Incorrect Count (intended to fail)



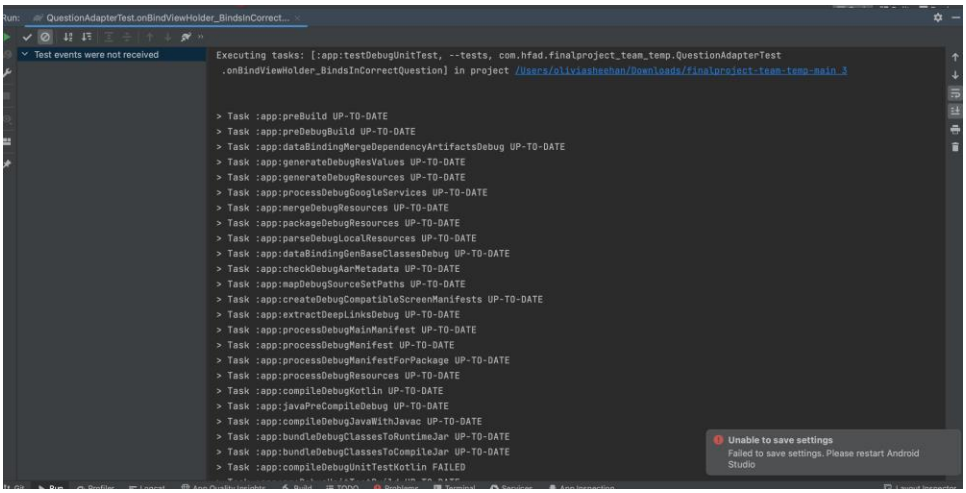


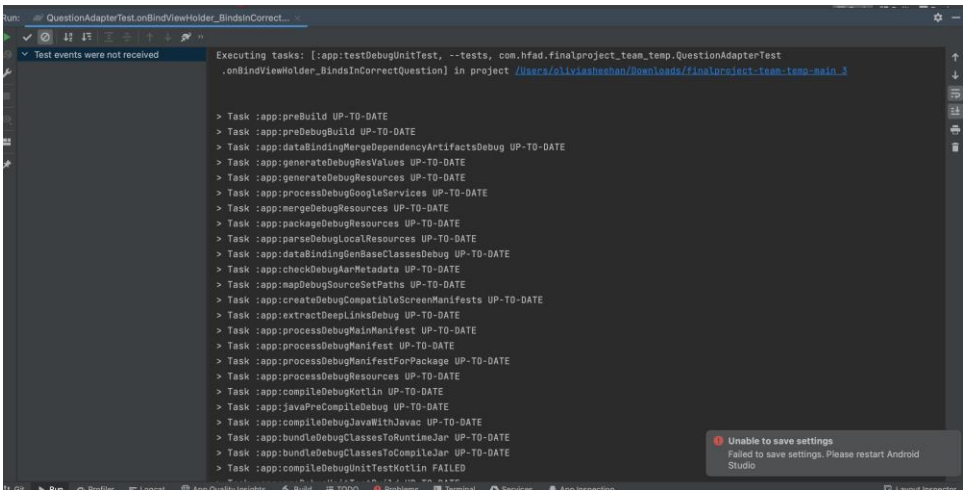
- Binds Correct Question





- Binds Incorrect Question (Intended to fail, it is out of bounds)


Run: QuestionAdapterTest.onBindViewHolder_BindsInCorrect...
Test events were not received
Executing tasks: [:app:testDebugUnitTest, --tests, com.hfad.finalproject_team_temp.QuestionAdapterTest
.onBindViewHolder_BindsInCorrectQuestion] in project /Users/oliviasheehan/Downloads/finalproject-team-temp-main 3
> Task :app:preBuild UP-TO-DATE
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:dataBindingMergeDependencyArtifactsDebug UP-TO-DATE
> Task :app:generateDebugResValues UP-TO-DATE
> Task :app:generateDebugResources UP-TO-DATE
> Task :app:processDebugGoogleServices UP-TO-DATE
> Task :app:mergeDebugResources UP-TO-DATE
> Task :app:packageDebugResources UP-TO-DATE
> Task :app:parseDebugLocalResources UP-TO-DATE
> Task :app:dataBindingGenBaseClassesDebug UP-TO-DATE
> Task :app:checkDebugAarMetadata UP-TO-DATE
> Task :app:mapDebugSourceSetPaths UP-TO-DATE
> Task :app:createDebugCompatibleScreenManifests UP-TO-DATE
> Task :app:extractDeepLinksDebug UP-TO-DATE
> Task :app:processDebugMainManifest UP-TO-DATE
> Task :app:processDebugManifest UP-TO-DATE
> Task :app:processDebugManifestForPackage UP-TO-DATE
> Task :app:processDebugResources UP-TO-DATE
> Task :app:compileDebugKotlin UP-TO-DATE
> Task :app:javaPreCompileDebug UP-TO-DATE
> Task :app:compileDebugJavaWithJavac UP-TO-DATE
> Task :app:bundleDebugClassesToRuntimeJar UP-TO-DATE
> Task :app:bundleDebugClassesToCompileJar UP-TO-DATE
> Task :app:compileDebugUnitTestKotlin FAILED
Unable to save settings
Failed to save settings. Please restart Android Studio


Run: QuestionAdapterTest.onBindViewHolder_BindsInCorrect...
Test events were not received
Executing tasks: [:app:testDebugUnitTest, --tests, com.hfad.finalproject_team_temp.QuestionAdapterTest
.onBindViewHolder_BindsInCorrectQuestion] in project /Users/oliviasheehan/Downloads/finalproject-team-temp-main 3
> Task :app:preBuild UP-TO-DATE
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:dataBindingMergeDependencyArtifactsDebug UP-TO-DATE
> Task :app:generateDebugResValues UP-TO-DATE
> Task :app:generateDebugResources UP-TO-DATE
> Task :app:processDebugGoogleServices UP-TO-DATE
> Task :app:mergeDebugResources UP-TO-DATE
> Task :app:packageDebugResources UP-TO-DATE
> Task :app:parseDebugLocalResources UP-TO-DATE
> Task :app:dataBindingGenBaseClassesDebug UP-TO-DATE
> Task :app:checkDebugAarMetadata UP-TO-DATE
> Task :app:mapDebugSourceSetPaths UP-TO-DATE
> Task :app:createDebugCompatibleScreenManifests UP-TO-DATE
> Task :app:extractDeepLinksDebug UP-TO-DATE
> Task :app:processDebugMainManifest UP-TO-DATE
> Task :app:processDebugManifest UP-TO-DATE
> Task :app:processDebugManifestForPackage UP-TO-DATE
> Task :app:processDebugResources UP-TO-DATE
> Task :app:compileDebugKotlin UP-TO-DATE
> Task :app:javaPreCompileDebug UP-TO-DATE
> Task :app:compileDebugJavaWithJavac UP-TO-DATE
> Task :app:bundleDebugClassesToRuntimeJar UP-TO-DATE
> Task :app:bundleDebugClassesToCompileJar UP-TO-DATE
> Task :app:compileDebugUnitTestKotlin FAILED
Unable to save settings
Failed to save settings. Please restart Android Studio