

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 forgotten 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 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 scoreboard 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	<ul style="list-style-type: none"><li>• Solely responsible for Create Quiz functionality, including database implementation for storing quizzes.</li><li>• Implemented recycler view for viewing quiz questions.</li><li>• Implemented notifications that occur every 24 hours if the user has not opened the app.</li><li>• Implemented database collection for storing quiz scores and made it so user scores are automatically saved to database upon finishing a quiz.</li><li>• Assisted with building the general skeleton of the app.</li><li>• Made it so the dashboard displays the currently logged in users top 5 quiz scores</li><li>• Added app logo</li></ul>

Maksym Bondarenko	<ul style="list-style-type: none"> <li>Implemented a "Storyboard" fragment transition template that allows for the cycling between fragments</li> <li>Created initial core functionality of Edit quiz via the mentioned Storyboard functionality</li> <li>Redefined Theme.xml to allow for custom color themes</li> <li>Assisted with UI re-design to be more consistent within the brand's theme</li> <li>Defined explicit button and header theming</li> </ul>
Olivia Sheehan	<ul style="list-style-type: none"> <li>Ensured that the quiz id matched in our program and the firebase to ensure consistency</li> <li>Created unit tests</li> <li>Conclusion of the Final Report</li> <li>Appendix for the Final Report</li> <li>Summary/Abstract for the Final Report</li> <li>Limitations for the Final Report</li> <li>Some of the concepts in "List of Implemented Concepts" for the Final Report</li> </ul>
Jared Ryan	<ul style="list-style-type: none"> <li>Created Navigation drawer to switch between main fragments and skeleton of application.</li> <li>Implemented Login and Signup Fragment and functionalities.</li> <li>When signing up, the users data is stored into the "users" firebase collection</li> <li>When logging in, the user data is retrieved from the firebase.</li> <li>Created Forgot password functionality so user can create a new password if forgotten</li> <li>Created Dashboard Fragment. Upon logging in, the user is sent here.</li> <li>Implemented a view model live data to display a welcome message for the current user</li> <li>Created a spinner that pulls quiz names from database to easily select the quiz you want to take</li> <li>Implemented user sessions. Session starts when logged in</li> <li>Implemented Logout feature. Logging out ends the user session so another user can sign into their account</li> </ul>

Casey McGuan	<ul style="list-style-type: none"> <li>• Designed the core feature of the app the “take quiz” function</li> <li>• Debugging</li> <li>• Created the system to store the users scores for other app features</li> <li>• Cleaned up unused and excess code from different functions throughout</li> </ul>
Jacob Irwin	<ul style="list-style-type: none"> <li>• Implemented the Scoreboard Feature</li> <li>• Debugged the app</li> <li>• Implemented profile pictures, Usernames, and score within scoreboard app</li> <li>• Created scoreboard fragment</li> </ul>
Simon Hausmaninger	<ul style="list-style-type: none"> <li>• Implemented Edit Quiz functionality</li> <li>• 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.</li> <li>• Implemented scoreboard fragment connecting it to the Firebase</li> <li>• The Scoreboard Fragment manages a Fragment within an Android app, presenting a Recycler View populated with user scores retrieved from Firebase Firestore</li> <li>• 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.</li> <li>• Assisted in the passing and handling of information between fragments and activities</li> </ul>

### 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:

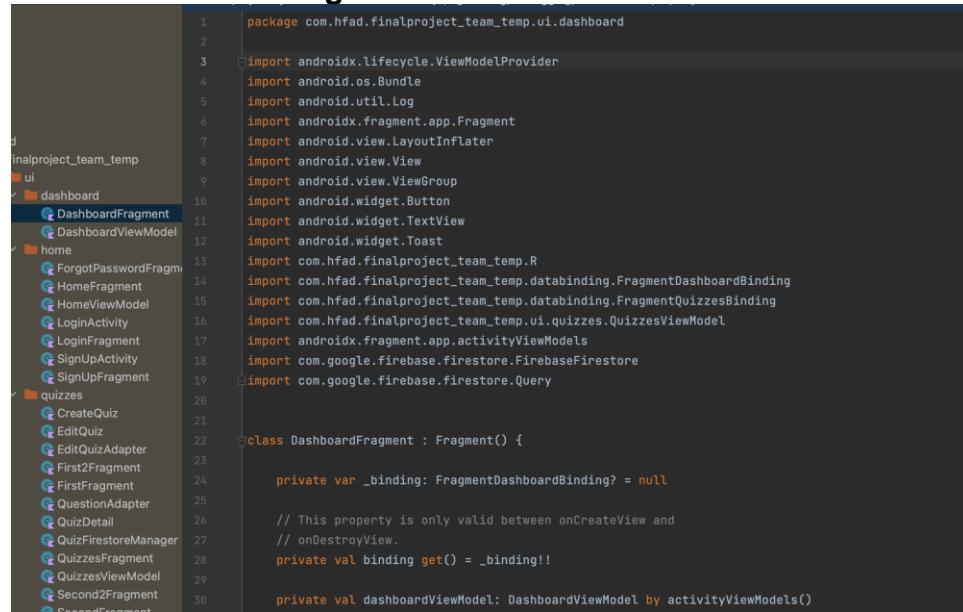
## Appendix:

### CODE

### UI

#### Dashboard

- **DashboardFragment:**



```
1 package com.hfad.finalproject_team_temp.ui.dashboard
2
3 import androidx.lifecycle.ViewModelProvider
4 import android.os.Bundle
5 import android.util.Log
6 import androidx.fragment.app.Fragment
7 import android.view.LayoutInflater
8 import android.view.View
9 import android.view.ViewGroup
10 import android.widget.Button
11 import android.widget.TextView
12 import android.widget.Toast
13 import com.hfad.finalproject_team_temp.R
14 import com.hfad.finalproject_team_temp.databinding.FragmentDashboardBinding
15 import com.hfad.finalproject_team_temp.databinding.FragmentQuizzesBinding
16 import com.hfad.finalproject_team_temp.ui.quizzes.QuizzesViewModel
17 import androidx.fragment.app.activityViewModels
18 import com.google.firebase.firestore.FirebaseFirestore
19 import com.google.firebase.firestore.Query
20
21
22 class DashboardFragment : Fragment() {
23
24     private var _binding: FragmentDashboardBinding? = null
25
26     // This property is only valid between onCreateView and
27     // onDestroyView.
28     private val binding get() = _binding!!
29
30     private val dashboardViewModel: DashboardViewModel by activityViewModels()
```

```
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
    //}
}

    textViews.addAll(listOf(
        root.findViewById(R.id.textView),
        root.findViewById(R.id.textView7),
        root.findViewById(R.id.textView8),
        root.findViewById(R.id.textView9),
        root.findViewById(R.id.textView10)
    ))
    return root
}

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

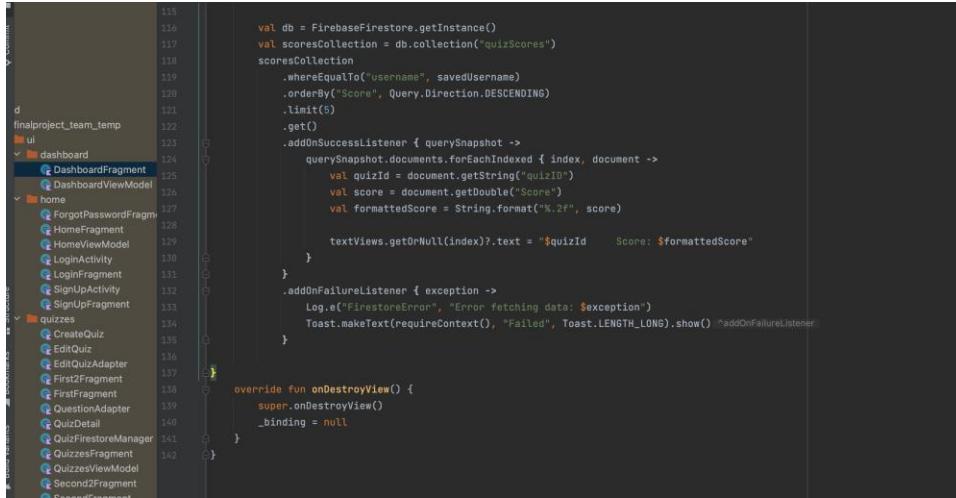
    val savedUsername = dashboardViewModel.loadUsername(requireContext()) // obtain username of currently logged in user

    val username = arguments?.getString("username")
    if (dashboardViewModel.welcomeMessage.value.isNullOrEmpty()) {
        username?.let {
            dashboardViewModel.setWelcomeMessage(it, requireContext())
        }
    }
}

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

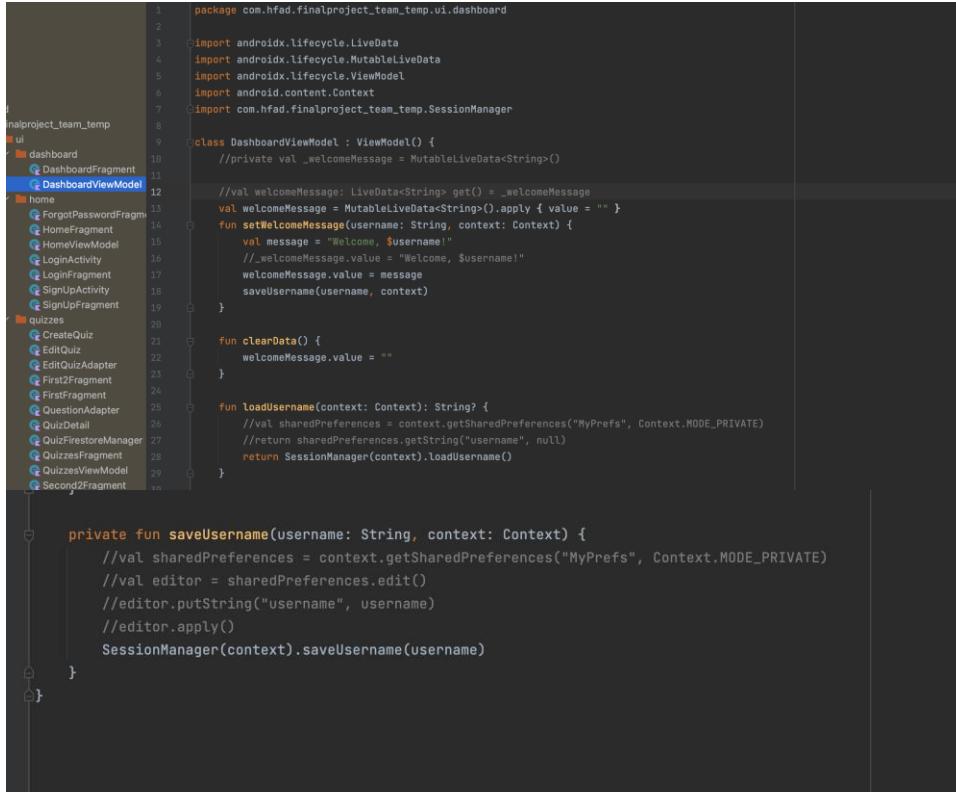
    val username = arguments?.getString("username")

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



```
115     val db = FirebaseFirestore.getInstance()
116     val scoresCollection = db.collection("quizScores")
117     scoresCollection
118         .whereEqualTo("username", savedUsername)
119         .orderBy("score", Query.Direction.DESCENDING)
120         .limit(5)
121         .get()
122             .addOnSuccessListener { querySnapshot ->
123                 querySnapshot.documents.forEachIndexed { index, document ->
124                     val quizId = document.getString("quizID")
125                     val score = document.getDouble("Score")
126                     val formattedScore = String.format("%.2f", score)
127
128                     textViews.getOrNull(index)?.text = "$quizId Score: $formattedScore"
129                 }
130             }
131             .addOnFailureListener { exception ->
132                 Log.e("FirestoreError", "Error fetching data: $exception")
133                 Toast.makeText(requireContext(), "Failed", Toast.LENGTH_LONG).show()
134             }
135         }
136
137     override fun onDestroyView() {
138         super.onDestroyView()
139         _binding = null
140     }
141 }
```

- DashboardViewModel



```
1 package com.hfad.finalproject_team_temp.ui.dashboard
2
3 import androidx.lifecycle.LiveData
4 import androidx.lifecycle.MutableLiveData
5 import androidx.lifecycle.ViewModel
6 import android.content.Context
7 import com.hfad.finalproject_team_temp.SessionManager
8
9 class DashboardViewModel : ViewModel() {
10     //private val _welcomeMessage = MutableLiveData<String>()
11
12     //val welcomeMessage: LiveData<String> get() = _welcomeMessage
13     val welcomeMessage = MutableLiveData<String>().apply { value = "" }
14     fun setWelcomeMessage(username: String, context: Context) {
15         val message = "Welcome $username"
16         //_welcomeMessage.value = "Welcome, $username!"
17         welcomeMessage.value = message
18         saveUsername(username, context)
19     }
20
21     fun clearData() {
22         welcomeMessage.value = ""
23     }
24
25     fun loadUsername(context: Context): String? {
26         //val sharedpreferences = context.getSharedPreferences("MyPrefs", Context.MODE_PRIVATE)
27         //return sharedpreferences.getString("username", null)
28         return SessionManager(context).loadUsername()
29     }
30
31     private fun saveUsername(username: String, context: Context) {
32         //val sharedpreferences = context.getSharedPreferences("MyPrefs", Context.MODE_PRIVATE)
33         //val editor = sharedpreferences.edit()
34         //editor.putString("username", username)
35         //editor.apply()
36         SessionManager(context).saveUsername(username)
37     }
38 }
```

Home

- ForgotPasswordFragment

```
1 package com.hfad.finalproject_team_temp.ui.home
2
3 import ...
4
5
6 class ForgotPasswordFragment : Fragment() {
7     private val db = FirebaseFirestore.getInstance()
8     @SuppressLint("MissingInflatedId")
9     override fun onCreateView(
10         inflater: LayoutInflater,
11         container: ViewGroup,
12         savedInstanceState: Bundle?
13     ): View {
14         val view = inflater.inflate(R.layout.fragment_forgot_password, container, false)
15
16         val emailOrUsername = view.findViewById<EditText>(R.id.textEmailOrPassword).text.toString()
17         val newPassword = view.findViewById<EditText>(R.id.textNewPassword).text.toString()
18         val confirmPassword = view.findViewById<EditText>(R.id.textConfirmPassword).text.toString()
19
20         val submitButton = view.findViewById<Button>(R.id.bSubmit)
21         val backtoLogin = view.findViewById<Button>(R.id.bBacktoLogin)
22
23         submitButton.setOnClickListener {
24             if (checkIfExists(emailOrUsername)) {
25                 if (newPassword == confirmPassword) {
26                     updatePassword(emailOrUsername, newPassword)
27                     //Toast.makeText(requireContext(), "Password Successfully Reset", Toast.LENGTH_LONG).show()
28                 } else {
29                     Toast.makeText(requireContext(), "Passwords do not match", Toast.LENGTH_LONG).show()
30                 }
31             } else {
32                 Toast.makeText(requireContext(), "Email or Username Not Found", Toast.LENGTH_LONG).show()
33             }
34         }
35         backtoLogin.setOnClickListener {
36             findNavController().navigate(R.id.action_forgotPassword_to_Login)
37         }
38
39         // Inflate the layout for this fragment
40         return view
41     }
42
43     private fun checkIfExists(emailOrUsername: String, callback: (Boolean) -> Unit) {
44         db.collection("users")
45             .whereEqualTo("Email", emailOrUsername)
46             .get()
47             .addOnCompleteListener { emailQuery ->
48                 if (emailQuery.isSuccessful && emailQuery.result != null && !emailQuery.result!!.isEmpty) {
49                     callback.invoke(true)
50                 } else {
51                     callback.invoke(false)
52                 }
53             }
54     }
55
56     private fun updatePassword(emailOrUsername: String, newPassword: String) {
57         db.collection("users")
58             .document(emailOrUsername)
59             .update("Password", newPassword)
60             .addOnSuccessListener {
61                 Log.d("UpdatePassword", "Password Updated successfully")
62             }
63             .addOnFailureListener { e ->
64                 Log.e("UpdatePassword", "Error updating password", e)
65             }
66
67             .whereEqualTo("Email", emailOrUsername)
68             .get()
69             .addOnSuccessListener { emailQuery ->
70                 if (emailQuery.documents.isNotEmpty()) {
71                     val doc = emailQuery.documents[0]
72                     val docId = doc.id
73                     val userId = doc.get("userId")
74                     val user = User(userId, docId, emailOrUsername, newPassword)
75                     db.collection("users").document(docId).set(user)
76                     callback.invoke(true)
77                 } else {
78                     callback.invoke(false)
79                 }
80             }
81
82         // val userId = "wclM9HMaxpC90wgSHSS2"
83         db.collection("users")
84             /*
85             .document(userId)
86             .update("Password", newPassword)
87             .addOnSuccessListener {
88                 Log.d("UpdatePassword", "Password Updated successfully")
89             }
90             .addOnFailureListener { e ->
91                 Log.e("UpdatePassword", "Error updating password", e)
92             }
93
94             */
95
96             .whereEqualTo("Email", emailOrUsername)
97             .get()
98             .addOnSuccessListener { emailQuery ->
99                 if (emailQuery.documents.isNotEmpty()) {
100                     val doc = emailQuery.documents[0]
101                     val docId = doc.id
102                     val user = User(docId, emailOrUsername, newPassword)
103                     db.collection("users").document(docId).set(user)
104                     callback.invoke(true)
105                 } else {
106                     callback.invoke(false)
107                 }
108             }
109
110         //val docId = doc.id
111         //val user = User(docId, emailOrUsername, newPassword)
112         //db.collection("users").document(docId).set(user)
113     }
114 }
```

```
183     Log.d("updatePassword", "Found user by email")
184
185     db.collection("users")
186         .document(userID)
187         .update("Password", newPassword)
188         .addOnSuccessListener {
189             Log.d("UpdatePassword", "password updated successfully")
190             Toast.makeText(requireContext(), "Password Successfully Reset", Toast.LENGTH_LONG).show() ^addOnSuccessListener
191         }
192         .addOnFailureListener { e ->
193             Log.e("updatePassword", "error updating password", e)
194         } ^addOnSuccessListener
195     } else {
196         db.collection("users")
197             .whereEqualTo("Username", emailOrUsername)
198             .get()
199             .addOnSuccessListener { usernameQuery ->
200                 if (usernameQuery.documents.isNotEmpty()) {
201                     val doc = usernameQuery.documents[0]
202                     val userID = doc.id
203
204                     Log.d("updatePassword", "Found user by username")
205                     db.collection("users")
206                         .document(userID)
207                         .update("Password", newPassword)
208                         .addOnSuccessListener {
209                             Log.d("UpdatePassword", "password updated successfully")
210                             Toast.makeText(requireContext(), "Password Successfully Reset", Toast.LENGTH_LONG).show() ^addOnSuccessListener
211                         }
212                         .addOnFailureListener { e ->
213                             Log.e("updatePassword", "error updating password", e)
214                         } ^addOnSuccessListener
215                 }
216             }
217         }
218     }
219 }
220 }
```

- HomeFragment

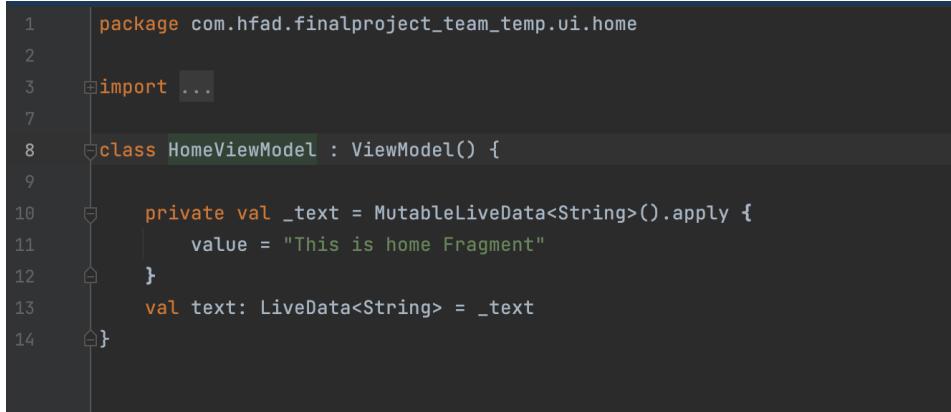
```
1 package com.hfad.finalproject_team_temp.ui.home
2
3 import ...
4
5
6
7
8 class HomeFragment : Fragment() {
9     private lateinit var sessionManager: SessionManager
10
11     private var _binding: FragmentHomeBinding? = null
12
13     // This property is only valid between onCreateView and
14     // onDestroyView.
15     private val binding get() = _binding!!
16
17     override fun onCreateView(inflater: LayoutInflater,
18                             container: ViewGroup?,
19                             savedInstanceState: Bundle?
20     ): View {
21
22         val view = inflater.inflate(R.layout.fragment_home, container, false)
23
24         sessionManager = SessionManager(requireContext())
25
26         val signUpButton: Button = view.findViewById(R.id.bSignup)
27         signUpButton.setOnClickListener {
28             findNavController().navigate(R.id.action_nav_home_to_signupFragment)
29         }
30
31         val loginButton: Button = view.findViewById(R.id.bLogin)
32
33     }
34
35 }
```



```
project_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

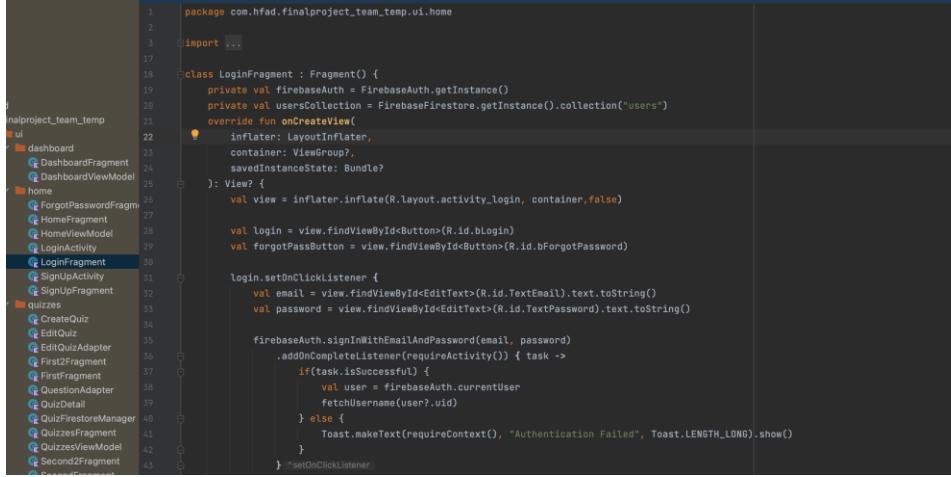
40
41     val loginButton: Button = view.findViewById(R.id.bLogin)
42     loginButton.setOnClickListener {
43         if (sessionManager.isLoggedIn()) {
44             ...
45         } else {
46             findNavController().navigate(R.id.action_nav_home_to_loginFragment)
47         }
48     }
49     return view
50 }
51
52 override fun onDestroyView() {
53     super.onDestroyView()
54     _binding = null
55 }
```

- HomeViewModel



```
1 package com.hfad.finalproject_team_temp.ui.home
2
3 import ...
4
5 class HomeViewModel : ViewModel() {
6
7     private val _text = MutableLiveData<String>().apply {
8         value = "This is home Fragment"
9     }
10    val text: LiveData<String> = _text
11 }
```

- LoginFragment



```
1 package com.hfad.finalproject_team_temp.ui.home
2
3 import ...
4
5 class LoginFragment : Fragment() {
6     private val firebaseAuth = FirebaseAuth.getInstance()
7     private val usersCollection = FirebaseFirestore.getInstance().collection("users")
8
9     override fun onCreateView(
10         inflater: LayoutInflater,
11         container: ViewGroup,
12         savedInstanceState: Bundle?
13     ): View? {
14         val view = inflater.inflate(R.layout.activity_login, container, false)
15
16         val login = view.findViewById(R.id.bLogin)
17         val forgotPassButton = view.findViewById(R.id.bForgotPassword)
18
19         login.setOnClickListener {
20             val email = view.findViewById<EditText>(R.id.TextEmail).text.toString()
21             val password = view.findViewById<EditText>(R.id.TextPassword).text.toString()
22
23             firebaseAuth.signInWithEmailAndPassword(email, password)
24                 .addOnCompleteListener(requireActivity()) { task -
25                     if(task.isSuccessful) {
26                         val user = firebaseAuth.currentUser
27                         fetchUsername(user?.uid)
28                     } else {
29                         Toast.makeText(requireContext(), "Authentication Failed", Toast.LENGTH_LONG).show()
30                     }
31                 }.setOnCompleteListener
32 }
```

```
    } "setOnClickListener"
}
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

```
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}")
    }.addOnCompleteListener
} else {
    Toast.makeText(requireContext(), "Authentication Failed", Toast.LENGTH_LONG).show() "addOnCompleteListener"
}
}

} 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

```
1 package com.hfad.finalproject_team_temp.ui.quizzes
2
3 import ...
4
5
6 class CreateQuiz : AppCompatActivity() {
7
8     private lateinit var appBarConfiguration: AppBarConfiguration
9     private lateinit var binding: ActivityCreateQuizBinding
10
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         window.statusBarColor = ContextCompat.getColor(this, R.color.theme_purple)
14         binding = ActivityCreateQuizBinding.inflate(layoutInflater)
15         setContentView(binding.root)
16
17         val fragment = QuizzesFragment() // Replace with your fragment class
18         supportFragmentManager.beginTransaction()
19             .replace(R.id.nav_gallery, fragment)
20             .addToBackStack(null) // Add the fragment to the back stack
21             .commit()
22
23         var questionList = arrayListOf<String]()
24         var answerList = arrayListOf<String]()
25         val addButton = findViewById<Button>(R.id.addButton)
26         val doneButton = findViewById<Button>(R.id.doneButton)
27         val doneButton = findViewById<Button>(R.id.doneButton)
28         addButton.setOnClickListener{
29             val title = findViewById<EditText>(R.id.title).text.toString()
30             val question = findViewById<EditText>(R.id.question).text.toString()
31             val answer = findViewById<EditText>(R.id.answer).text.toString()
32             questionList.add(question)
33         }
34
35         doneButton.setOnClickListener{
36             val intent = Intent(this, QuizResultActivity::class.java)
37             intent.putExtra("questionList", questionList)
38             intent.putExtra("answerList", answerList)
39             startActivity(intent)
40         }
41
42     }
43
44 }
```

```

    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: String = StringBuffer()
                if(it.isSuccessful) {
                    for(document in it.result!!) {
                        Log.d("dbfirebase", "retrieve:" +
                            "${document.data.getValue("Title")}" +
                            "${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

```
1 package com.hfad.finalproject_team_temp.ui.quizzes
2
3 import ...
4
5
6 class EditQuiz : AppCompatActivity() {
7
8     private lateinit var recyclerView: RecyclerView
9     private lateinit var adapter: EditQuizAdapter
10    private lateinit var quizId: String
11    private lateinit var db: FirebaseFirestore
12
13    override fun onCreate(savedInstanceState: Bundle?) {
14        super.onCreate(savedInstanceState)
15        setContentView(R.layout.activity_edit_quiz)
16        quizId = intent.getStringExtra("quizId") ?: ""
17
18        db = FirebaseFirestore.getInstance()
19
20        setSupportActionBar(toolbar)
21        toolbar.setNavigationOnClickListener {
22            finish()
23        }
24
25        setupActionBar()
26        setupRecyclerView()
27
28        val backButton = findViewById<Button>(R.id.backButton)
29        backButton.setOnClickListener {
30            finish()
31        }
32
33        supportActionBar?.setDisplayHomeAsUpEnabled(true)
34    }
35
36    private fun setupActionBar() {
37        // Enable the back button in the ActionBar/Toolbar
38        supportActionBar?.setDisplayHomeAsUpEnabled(true)
39    }
40
41 }
```

```
    42     }
    43
    44     override fun onSupportNavigateUp(): Boolean {
    45         // Handle the back button press
    46         finish() // Close this activity and go back
    47         return true
    48     }
    49
    50     private fun setupRecyclerView() {
    51         db.collection("quizzes")
    52             .whereEqualTo("title", quizId)
    53             .get()
    54             .addOnSuccessListener { querySnapshot ->
    55                 val questionList = mutableListOf<String>()
    56                 val answerList = mutableListOf<String>()
    57
    58                 for (document in querySnapshot.documents) {
    59                     val questions = document["questions"] as? List<String>
    60                     val answers = document["answers"] as? List<String>
    61
    62                     questions?.let { questionList.addAll(it) }
    63                     answers?.let { answerList.addAll(it) }
    64                 }
    65
    66                 recyclerView = findViewById(R.id.recyclerViewEditQuiz)
    67                 adapter = EditQuizAdapter(questionList.toMutableList(), answerList.toMutableList())
    68                 recyclerView.layoutManager = LinearLayoutManager(this)
    69                 recyclerView.adapter = adapter
    70
    71                 val saveButton = findViewById<Button>(R.id.saveButton)
    72             }
    73
    74             .addOnFailureListener { exception ->
    75                 Log.d("EditQuizActivity", "Error getting documents: ${exception.message}")
    76             }
    77         }
    78
    79     }
    80
    81     companion object {
    82         const val QUIZ_ID = "QUIZ_ID"
    83     }
    84 }
```

```
    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()
    }.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
    }
}
```

```

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
      QuizzesView
  database
    quizzes
      EditQuiz

```

```

    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**

```

finalproject_team_temp
  ui
    dashboard
      DashboardFragment
      DashboardViewModel
    home
      ForgotPasswordFragment
      HomeFragment
      HomeViewModel
      LoginActivity
      LoginFragment
      SignUpActivity
      SignUpFragment
    quizzes
      CreateQuiz
      EditQuiz
      EditQuizAdapter
      EditQuizAdapter
      First2Fragment
      FirstFragment
      QuestionAdapter
      QuizDetail
      QuizFirestoreManager
      QuizzesFragment
      QuizzesViewModel
      QuizzesView
      Second2Fragment
      SecondFragment
      TakeQuiz
    scoreboard

```

```

package com.hfad.finalproject_team_temp.ui.quizzes

import ...

class EditQuizAdapter : 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_quiz, 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])
        }
    }
}

```

```

finalproject_team_temp
  ui
    dashboard
      DashboardFragment
      DashboardViewModel
    home
      ForgotPasswordFragment
      HomeFragment
      HomeViewModel
      LoginActivity
      LoginFragment
      SignUpActivity
      SignUpFragment
    quizzes
      CreateQuiz
      EditQuiz
      EditQuizAdapter
      EditQuizAdapter
      First2Fragment
      FirstFragment
      QuestionAdapter
      QuizDetail
      QuizFirestoreManager
      QuizzesFragment
      QuizzesViewModel
      QuizzesView
      Second2Fragment
      SecondFragment
      TakeQuiz
    scoreboard

```

```

        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

    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
            notifyDataSetChanged()
        }
    }
}

```

```
55     }
56
57     fun updateQuestionAtPosition(position: Int, updatedQuestion: String) {
58         if (position >= 0 && position < questionList.size) {
59             questionList[position] = updatedQuestion
60             notifyItemChanged(position + 1) // +1 for the header
61         }
62     }
63
64     fun updateAnswerAtPosition(position: Int, updatedAnswer: String) {
65         if (position >= 0 && position < answerList.size) {
66             answerList[position] = updatedAnswer
67             notifyItemChanged(position + 1) // +1 for the header
68         }
69     }
70
71     // Function to retrieve updated answer list
72     fun getUpdatedAnswers(): List<String> {
73         return answerList
74     }
75
76 }
```

The screenshot shows the code for the `EditQuizAdapter` class. It contains methods to update questions and answers at specific positions in their respective lists. It also provides a function to get the updated answer list.

- QuestionAdapter

```
1 package com.hfad.finalproject_team_temp.ui.quizzes
2
3 import ...
4
5 class QuestionAdapter (private val questions: List<String>, answers: List<String>): RecyclerView.Adapter<QuestionAdapter.ViewHolder>() {
6     inner class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
7         val questionTextView: TextView = itemView.findViewById(R.id.textQuestion)
8
9         init {
10             itemView.setOnClickListener {
11                 val intent = Intent(itemView.context, QuizDetail::class.java)
12                 intent.putExtra("ID", questionTextView.text)
13
14                 itemView.context.startActivity(intent)
15             }
16         }
17     }
18
19     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
20         val view = LayoutInflater.from(parent.context).inflate(R.layout.question_item, parent, false)
21         return ViewHolder(view)
22     }
23
24     override fun onBindViewHolder(holder: ViewHolder, position: Int) {
25         val question = questions[position]
26         holder.questionTextView.text = question
27     }
28
29     override fun getItemCount(): Int {
30
31     }
32
33     companion object {
34         const val TAG = "QuestionAdapter"
35     }
36 }
37
38 }
```

The screenshot shows the code for the `QuestionAdapter` class. It extends `RecyclerView.Adapter` and defines a `ViewHolder`. It handles item click events by starting an activity with the question ID. It also overrides `onCreateViewHolder`, `onBindViewHolder`, and `getItemCount` methods.

- QuizDetail

```

d
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
      TakeQuiz
    scoreboard

```

```

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.TextView
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.firebaseio.firebaseio.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)
    }
}

```

```

  Structure
    SignUpActivity
    SignUpFragment
    quizzes
      CreateQuiz
      EditQuiz
      EditQuizAdapter
      First2Fragment
      FirstFragment
      QuestionAdapter
      QuizDetail
      QuizFirestoreManager
      QuizzesFragment
      QuizzesViewModel
    Second2Fragment
    SecondFragment
    TakeQuiz

```

- QuizFirestoreManager

```

1 import com.google.firebaseio.FirebaseFirestore
2
3 class QuizFirestoreManager {
4     private val db = FirebaseFirestore.getInstance()
5     private val quizzesCollection = db.collection("quizzes")
6
7     fun addQuiz(quizId: String, quizData: Map<String, Any>){
8         // Set the document ID to be the same as the quizId
9         val quizRef = quizzesCollection.document(quizId)
10
11         quizRef.set(quizData)
12             .addOnSuccessListener {
13                 // Quiz added successfully
14             }
15             .addOnFailureListener { e ->
16                 // Handle failure
17             }
18     }
19
20     fun editQuiz(quizId: String, updatedQuizData: Map<String, Any>) {
21         quizzesCollection.document(quizId)
22             .update(updatedQuizData)
23             .addOnSuccessListener {
24                 // Quiz updated successfully
25             }
26             .addOnFailureListener { e ->
27                 // Handle failure
28             }
29     }
30
31     fun deleteQuiz(quizId: String) {
32         quizzesCollection.document(quizId)
33             .delete()
34             .addOnSuccessListener {
35                 // Quiz deleted successfully
36             }
37             .addOnFailureListener { e ->
38                 // Handle failure
39             }
40     }
41
42     fun deleteQuiz(quizzesFragment: QuizzesFragment) {
43         quizzesCollection.document(quizzesFragment.quizId)
44             .delete()
45             .addOnSuccessListener {
46                 // Quiz deleted successfully
47             }
48             .addOnFailureListener { e ->
49                 // Handle failure
50             }
51     }
52
53     fun collection(): CollectionReference {
54         return quizzesCollection
55     }
56
57     companion object {
58         @Volatile
59         private var instance: QuizFirestoreManager? = null
60
61         fun getInstance(): QuizFirestoreManager {
62             if (instance == null) {
63                 synchronized(QuizFirestoreManager::class.java) {
64                     if (instance == null) {
65                         instance = QuizFirestoreManager()
66                     }
67                 }
68             }
69             return instance!!
70         }
71     }
72 }

```

## ● QuizzesFragment

```

1 package com.hfad.finalproject_team_temp.ui.quizzes
2
3 import ...
4
5
6 class QuizzesFragment : Fragment() {
7     private var _binding: FragmentQuizzesBinding? = null
8     // var quiz_select = arrayListOf<String>()
9
10    // This property is only valid between onCreateView and
11    // onDestroyView.
12    private val binding get() = _binding!!
13    override fun onCreateView(
14        inflater: LayoutInflater,
15        container: ViewGroup?,
16        savedInstanceState: Bundle?
17    ): View {
18        val galleryViewModel =
19            ViewModelProvider(this).get(QuizzesViewModel::class.java)
20
21        _binding = FragmentQuizzesBinding.inflate(inflater, container, false)
22        val root: View = binding.root
23
24        // val textView: TextView = binding.textGallery // getting unresolved reference here
25        // galleryViewModel.text.observe(viewLifecycleOwner) {
26        //     textView.text = it
27        //}
28        val db = FirebaseFirestore.getInstance()
29        val spinner: Spinner = root.findViewById(R.id.QuiZSelect)
30
31        db.collection("quizzes")
32
33    }
34
35    companion object {
36        @Volatile
37        private var instance: QuizzesFragment? = null
38
39        fun getInstance(): QuizzesFragment {
40            if (instance == null) {
41                synchronized(QuizzesFragment::class.java) {
42                    if (instance == null) {
43                        instance = QuizzesFragment()
44                    }
45                }
46            }
47            return instance!!
48        }
49    }
50 }

```

```

    db.collection("quizzes")
        .get()
        .addOnCompleteListener { task ->
            if (task.isSuccessful) {
                val quizTitles = ArrayList<String>()
                for (document in task.result!!) {
                    val title = document.getString("title")
                    title?.let { quizTitles.add(it) }
                }
                val adapter = ArrayAdapter(
                    requireContext(),
                    android.R.layout.simple_spinner_item,
                    quizTitles
                )
                adapter.setDropDownViewResource(android.R.layout.simple_spinner_item)
                spinner.adapter = adapter
            } else {
                Log.w("QuizFragment", "Error getting quizzes", task.exception)
            }
        }

    val bToQuiz = root.findViewById<Button>(R.id.bToQuiz)
    bToQuiz.setOnClickListener {
        val selectedQuizTitle = spinner.selectedItem?.toString()
        Log.d(selectedQuizTitle, "selectedquiztitletest")
        val intent = Intent(activity, QuizDetail::class.java)
        intent.putExtra("ID", selectedQuizTitle)
        startActivity(intent)
    }

    val bToQuiz = root.findViewById<Button>(R.id.bToQuiz)
    bToQuiz.setOnClickListener {
        val selectedQuizTitle = spinner.selectedItem?.toString()
        Log.d(selectedQuizTitle, "selectedquiztitletest")
        val intent = Intent(activity, QuizDetail::class.java)
        intent.putExtra("ID", selectedQuizTitle)
        startActivity(intent)
    }

    val createQuiz = root.findViewById<Button>(R.id.createQuizButton)
    createQuiz.setOnClickListener {
        val intent = Intent(activity, CreateQuiz::class.java)
        startActivity(intent)
    }

    val fragment = CreateQuizFragment()
    val createQuiz = root.findViewById<Button>(R.id.createQuizButton)
    createQuiz.setOnClickListener {
        childFragmentManager.beginTransaction().replace(R.id.nav_host_fragment_content_create_quiz, fragment).commit()
        //findNavController().navigate(R.id.nav_host_fragment_content_create_quiz)
    }
    return root
}

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

```

- QuizzesViewModel

The screenshot shows the Android Studio code editor with the file `QuizzesViewModel.java` open. The code defines a `QuizzesViewModel` class that extends `ViewModel`. It contains a private `MutableLiveData<String>` variable `_text` initialized with the value "This is quizzes Fragment". A public `LiveData<String>` variable `text` is also defined, which is a reference to `_text`.

```
1 package com.hfad.finalproject_team_temp.ui.quizzes
2
3 import ...
4
5
6
7 class QuizzesViewModel : ViewModel() {
8
9     private val _text = MutableLiveData<String>().apply {
10         value = "This is quizzes Fragment"
11     }
12
13     val text: LiveData<String> = _text
14 }
```

## ● TakeQuiz

The screenshot shows the Android Studio code editor with the file `TakeQuiz.java` open. The code defines a `TakeQuiz` class that extends  `AppCompatActivity`. It imports various Android components like `Bundle`, `View`, `Button`, `EditText`, `TextView`, `Toast`, `ViewModelProvider`, `FirebaseFirestore`, and `DashboardViewModel`. The class overrides the `onCreate` method to set the content view, get the quiz ID from the intent, and initialize the dashboard view model. It also handles back button presses and initializes the next button.

```
1 package com.hfad.finalproject_team_temp.ui.quizzes
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5 import android.view.View
6 import android.widget.Button
7 import android.widget.EditText
8 import android.widget.TextView
9 import android.widget.Toast
10 import androidx.lifecycle.ViewModelProvider
11 import com.google.firebase.firestore.FirebaseFirestore
12 import com.hfad.finalproject_team_temp.R
13 import com.hfad.finalproject_team_temp.ui.dashboard.DashboardViewModel
14
15 class TakeQuiz : AppCompatActivity() {
16
17     private lateinit var quizId: String
18     private lateinit var dashboardViewModel: DashboardViewModel
19     override fun onCreate(savedInstanceState: Bundle?) {
20         super.onCreate(savedInstanceState)
21         setContentView(R.layout.activity_take_quiz)
22         quizId = intent.getStringExtra("quizId") ?: ""
23         dashboardViewModel = ViewModelProvider(this).get(DashboardViewModel::class.java)
24         val backButton = findViewById<Button>(R.id.bBack)
25         backButton.setOnClickListener {
26             onBackPressed()
27         }
28         val savedUsername = dashboardViewModel.loadUsername(this) // obtain username of current
29
30         val nextButton = findViewById<Button>(R.id.bNext)
31     }
32 }
```

```


// HomeViewModel.kt
class HomeViewModel : ViewModel() {
    // ...
}

// LoginActivity.kt
class LoginActivity : AppCompatActivity() {
    // ...
}

// LoginFragment.kt
class LoginFragment : Fragment() {
    // ...
}

// SignUpActivity.kt
class SignUpActivity : AppCompatActivity() {
    // ...
}

// quizzes
// CreateQuiz.kt
// EditQuiz.kt
// EditQuizAdapter.kt
// First2Fragment.kt
// FirstFragment.kt
// QuestionAdapter.kt
// QuizDetail.kt
// QuizFirestoreManager.kt
// QuizzesFragment.kt
// QuizzesViewModel.kt
// Second2Fragment.kt
// SecondFragment.kt
// TakeQuiz.kt
// scoreboard
// ScoreboardAdapter.kt
// ScoreboardFragment.kt
// ScoreboardViewModel.kt
// slideshow
// SlideshowFragment.kt
// SlideshowViewModel.kt
// theme
// Color.kt
// Theme.kt
// Type.kt
// alarm
// MainActivity.kt
// quizClass
// Second2Fragment.kt

// HomeViewModel.kt
class HomeViewModel : ViewModel() {
    // ...
}

// LoginActivity.kt
class LoginActivity : AppCompatActivity() {
    // ...
}

// LoginFragment.kt
class LoginFragment : Fragment() {
    // ...
}

// SignUpActivity.kt
class SignUpActivity : AppCompatActivity() {
    // ...
}

// quizzes
// CreateQuiz.kt
// EditQuiz.kt
// EditQuizAdapter.kt
// First2Fragment.kt
// FirstFragment.kt
// QuestionAdapter.kt
// QuizDetail.kt
// QuizFirestoreManager.kt
// QuizzesFragment.kt
// QuizzesViewModel.kt
// Second2Fragment.kt
// SecondFragment.kt
// TakeQuiz.kt
// scoreboard
// ScoreboardAdapter.kt
// ScoreboardFragment.kt
// ScoreboardViewModel.kt
// slideshow
// SlideshowFragment.kt
// SlideshowViewModel.kt
// theme
// Color.kt
// Theme.kt
// Type.kt
// alarm
// MainActivity.kt
// quizClass
// Second2Fragment.kt

// QuizDetail.kt
// QuizFirestoreManager.kt
// QuizzesFragment.kt
// QuizzesViewModel.kt
// Second2Fragment.kt
// SecondFragment.kt
// TakeQuiz.kt
// scoreboard
// ScoreboardAdapter.kt
// ScoreboardFragment.kt
// ScoreboardViewModel.kt
// slideshow
// SlideshowFragment.kt
// SlideshowViewModel.kt
// theme
// Color.kt
// Theme.kt
// Type.kt
// alarm
// MainActivity.kt
// quizClass
// Second2Fragment.kt


```

## Scoreboard

- ScoreboardAdapter

```
1 package com.hfad.finalproject_team_temp.ui.scoreboard
2
3 import android.view.LayoutInflater
4 import android.view.View
5 import android.view.ViewGroup
6 import android.widget.ImageView
7 import android.widget.TextView
8 import android.widget.Toast
9 import androidx.recyclerview.widget.RecyclerView
10 import com.hfad.finalproject_team_temp.R
11
12 class ScoreboardAdapter(private var titles: List<String>, private var details: List<String>, private var images: List<Int>) : RecyclerView.Adapter<ScoreboardAdapter.ViewHolder> {
13
14     inner class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
15
16         val itemTitle: TextView = itemView.findViewById(R.id.tv_title)
17         val itemDetail: TextView = itemView.findViewById(R.id.tv_description)
18         val itemPicture: ImageView = itemView.findViewById(R.id.iv_image)
19
20         init {
21             itemView.setOnClickListener { v: View ->
22                 val position: Int = adapterPosition
23                 Toast.makeText(itemView.context, "You clicked on item #${position + 1}", Toast.LENGTH_LONG).show()
24             }
25         }
26
27     }
28
29     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ScoreboardAdapter.ViewHolder {
30         val v: View = LayoutInflater.from(parent.context).inflate(R.layout.high_score, parent, false)
31         return ViewHolder(v)
32     }
33
34 }
```

```
SecondFragment 32    override fun onBindViewHolder(holder: ScoreboardAdapter.ViewHolder, position: Int) {  
TakeQuiz      33        holder.itemTitle.text = titles[position]  
scoreboard    34        holder.itemDetail.text = details[position]  
ScoreboardAdapter 35        holder.itemPicture.setImageResource(images[position])  
ScoreboardFragment 36    }  
ScoreboardViewModel 37  
slideshow     38    override fun getItemCount(): Int {  
SlideshowFragment 39        return titles.size  
SlideshowViewModel 40    }  
theme          41}
```

- ScoreboardFragment

```
1 package com.hfad.finalproject_team_temp.ui.scoreboard
2 import com.google.firebase.firestore.Query
3
4 import android.os.Bundle
5 import android.util.Log
6 import android.view.LayoutInflater
7 import android.view.View
8 import android.view.ViewGroup
9 import androidx.fragment.app.Fragment
10 import androidx.recyclerview.widget.LinearLayoutManager
11 import androidx.recyclerview.widget.RecyclerView
12 import com.google.firebase.firestore.FirebaseFirestore
13 import com.hfad.finalproject_team_temp.R
14
15 class ScoreboardFragment : Fragment() {
16
17     private var scoreboardList = mutableListOf<Pair<String, Double>>()
18     private var titlesList = mutableListOf<String>()
19     private var descriptionList = mutableListOf<String>()
20     private var imagesList = mutableListOf<Int>()
21
22     override fun onCreateView(
23         inflater: LayoutInflater,
24         container: ViewGroup?,
25         savedInstanceState: Bundle?
26     ): View? {
27         val view = inflater.inflate(R.layout.fragment_scoreboard, container, false)
28         val recyclerView: RecyclerView = view.findViewById(R.id.rv_recyclerView)
29         recyclerView.layoutManager = LinearLayoutManager(requireContext())
30         recyclerView.adapter = ScoreboardAdapter(titlesList, descriptionList, imagesList)
31
32         return view
33     }
34
35 }
```

```

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()
                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.drawable.ic_launcher_round)
            }

            val recyclerView: RecyclerView? = view?.findViewById(R.id.rv_recyclerView)
            recyclerView?.adapter?.notifyDataSetChanged() ^addOnSuccessListener
        }
        .addOnFailureListener { e ->
            Log.w("ScoreboardFragment", "Error displaying scoreboard", e)
        }
}

```

- Alarm

```

QuizzezV1
SecondF2
    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 channelId = "alarm"
            val importance = NotificationManager.IMPORTANCE_DEFAULT
            val channel = NotificationChannel("alarm", channelId, 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)
        }
    }
    Scoreboard
        Scoreboard
        Scoreboard
        Scoreboard
        Scoreboard
    Slideshow
        Slideshow
        Slideshow
    theme
        Color.kt
        Theme.kt
        Type.kt
    alarm
        MainActivity
        quizClass
        SessionManage
    wable
    ut
    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

```

```

QuizzezV1
SecondF2
    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)

    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

```
1 package com.hfad.finalproject_team_temp
2
3 import ...
4
5 class MainActivity : AppCompatActivity() {
6     private lateinit var sessionManager: SessionManager
7     private lateinit var dashboardViewModel: DashboardViewModel
8
9     private lateinit var appBarConfiguration: AppBarConfiguration
10    private lateinit var binding: ActivityMainBinding
11
12    override fun onCreate(savedInstanceState: Bundle?) {
13        super.onCreate(savedInstanceState)
14
15        sessionManager = SessionManager(this)
16
17        binding = ActivityMainBinding.inflate(layoutInflater)
18        setContentView(binding.root)
19
20        dashboardViewModel = ViewModelProvider(this).get(DashboardViewModel::class.java)
21
22        setSupportActionBar(binding.appBarMain.toolbar)
23        scheduleAlarm(context: this)
24
25        /*
26         * binding.appBarMain.fab.setOnClickListener { view ->
27         *     Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
28         *         .setAction("Action", null).show()
29         * }
30
31        */
32
33        val drawerLayout: DrawerLayout = binding.drawerLayout
34
35    }
36
37    val drawerLayout: DrawerLayout = binding.drawerLayout
38    val navView: NavigationView = binding.navView
39    val navCtrl = findNavController(R.id.nav_host_fragment_content_main)
40
41    // Passing each menu ID as a set of IDs because each
42    // menu should be considered as top level destinations.
43    appBarConfiguration = AppBarConfiguration(
44        setOf(
45            R.id.nav_home,
46            R.id.nav_gallery,
47            R.id.nav_slideshow,
48            R.id.nav_dashboardFragment,
49            R.id.nav_scoreboard
50        ), drawerLayout
51    )
52
53    setupActionBarWithNavController(navController, appBarConfiguration)
54    navView.setupWithNavController(navController)
55
56
57    override fun onOptionsItemSelected(item: MenuItem): Boolean {
58        when (item.itemId) {
59            /*
60            R.id.action_settings -> {
61                return true
62            }
63            */
64            R.id.action_logout -> {
65                sessionManager.logout()
66                dashboardViewModel.clearData()
67                navCtrl.findNavController(R.id.nav_host_fragment_content_main)
68            }
69        }
70    }
71
72    R.id.action_logout -> {
73        sessionManager.logout()
74        dashboardViewModel.clearData()
75        navCtrl.findNavController(R.id.nav_host_fragment_content_main)
76    }
77
78    R.id.action_logout -> {
79        sessionManager.logout()
80        dashboardViewModel.clearData()
81        navCtrl.findNavController(R.id.nav_host_fragment_content_main)
82    }
83
84    R.id.action_logout -> {
85        sessionManager.logout()
86        dashboardViewModel.clearData()
87        navCtrl.findNavController(R.id.nav_host_fragment_content_main)
88    }
89
90    R.id.action_logout -> {
91        sessionManager.logout()
92        dashboardViewModel.clearData()
93        navCtrl.findNavController(R.id.nav_host_fragment_content_main)
94    }
95
96    R.id.action_logout -> {
97        sessionManager.logout()
98        dashboardViewModel.clearData()
99        navCtrl.findNavController(R.id.nav_host_fragment_content_main)
100   }
101 }
```

## QuizClass

The screenshot shows the Android Studio interface. On the left is the Project Structure sidebar, which lists several packages and their contents. The 'quizClass' file is highlighted in the 'quizClass' package. The main area is the code editor, displaying the following Java code:

```
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
        )
    }
}

private fun findNavController(id: Int): NavController {
    return supportFragmentManager.findFragmentById(id) as NavController
}

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)
    }
```

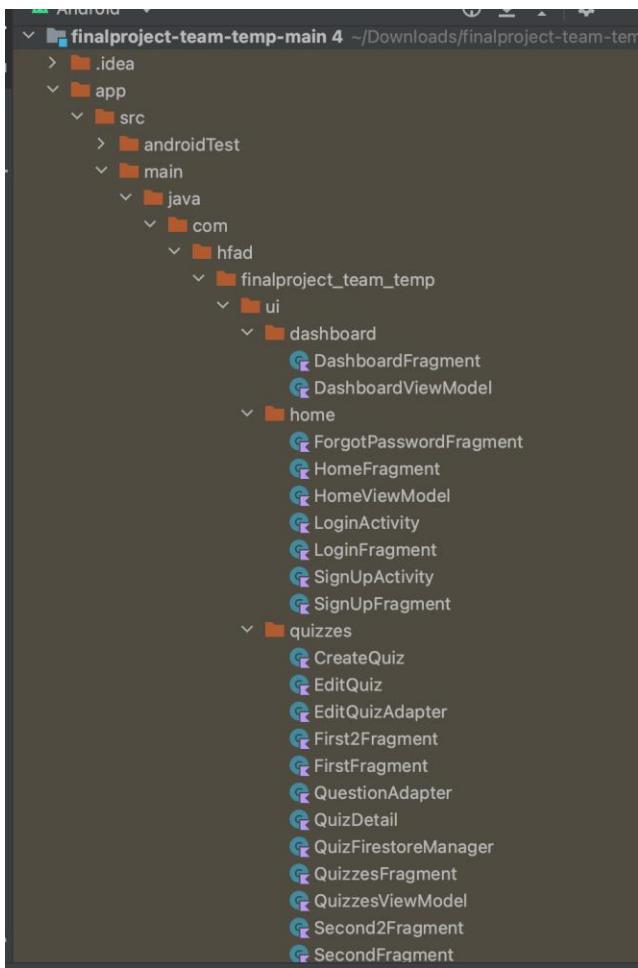
- SessionManager

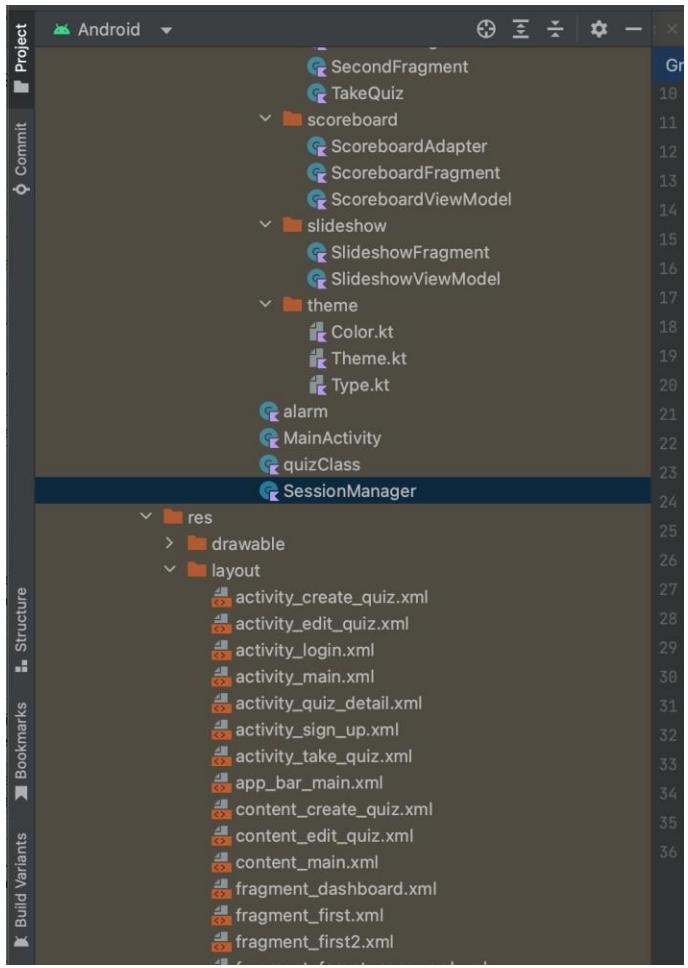
```
QuizzesView 1 package com.hfad.finalproject_team_temp
Second2Fra 2
SecondFrag 2
TakeQuiz 3 import ...
scoreboard 5
Scoreboard 6 class SessionManager (context: Context) {
Scoreboard 7     private val sharedpreferences: SharedPreferences =
Scoreboard 8         context.getSharedPreferences("user_session", Context.MODE_PRIVATE)
Scoreboard 9     private val editor: SharedPreferences.Editor = sharedpreferences.edit()
slideshow 10
SlideshowFr 10
SlideshowVi 11
theme 12
Color.kt 12
Theme.kt 13
Type.kt 14
alarm 15
MainActivity 16
quizClass 17
SessionManager 18
ble 19
ivity_create_quiz.xml 20
ivity_edit_quiz.xml 21
ivity_login.xml 22
ivity_main.xml 23
ivity_quiz_detail.xml 24
ivity_sign_up.xml 25
ivity_take_quiz.xml 26
a_bar_main.xml 27
intent_create_quiz.xml 28
intent_edit_quiz.xml 29
intent_main.xml 30
}

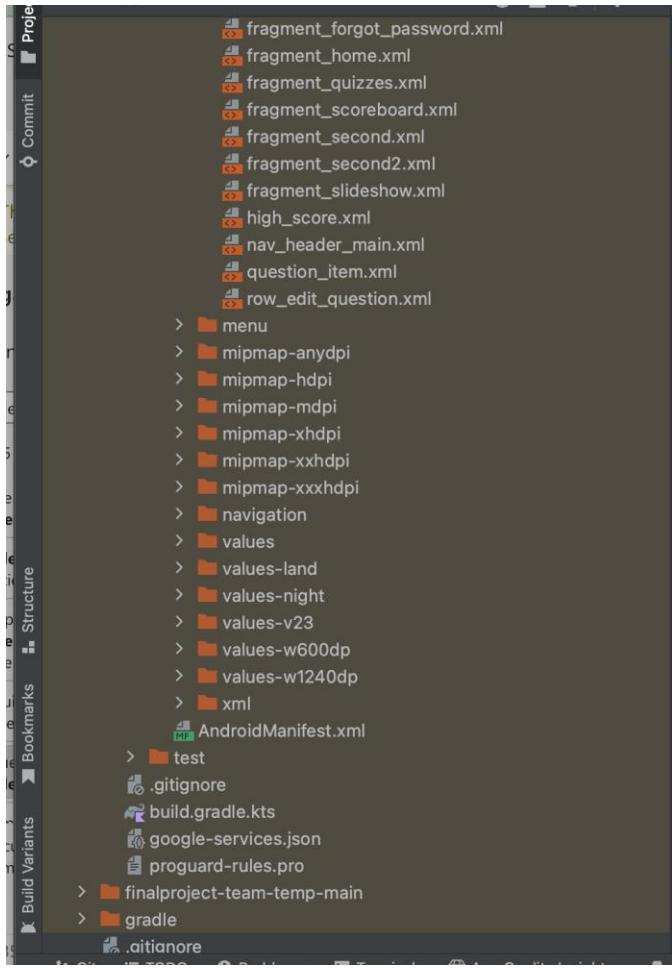
```

```
g
slideshow 18 }
SlideshowFr 19
SlideshowVi 20
theme 21
Color.kt 22
Theme.kt 23
Type.kt 24
alarm 25
MainActivity 26
quizClass 27
SessionManager 28
ble 29
ivity_create_quiz.xml 30
ivity_edit_quiz.xml 31
ivity_login.xml 32
ivity_main.xml 33
ivity_quiz_detail.xml 34
ivity_sign_up.xml 35
ivity_take_quiz.xml 36
}
```

Side Bar:

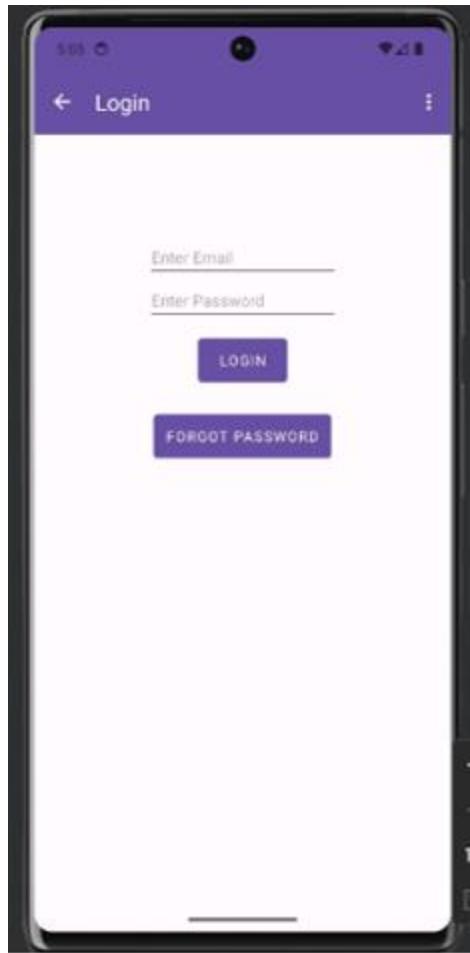


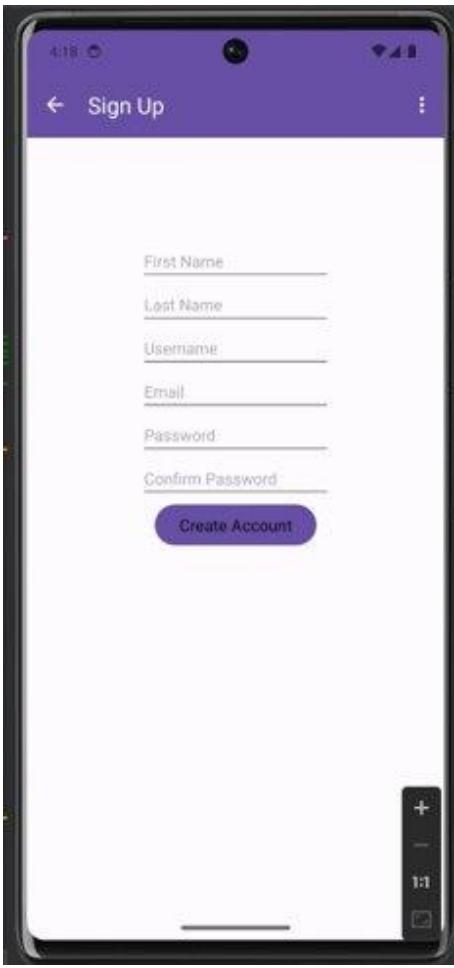




## USER INTERFACE

- Login, Sign Up, and Authentication:





The image shows the Firebase Cloud Firestore console. The left sidebar shows project settings like App Check, Authentication, and Cloud Functions. The main area shows the "users" collection with a document named "1qdeFlucEJQw2". This document contains fields: "Email" (ryan109@csusm.edu), "First Name" (Jack), "Last Name" (Ryan), and "Username" (Ryan109). Other documents in the collection have IDs starting with "1qdeFlucEJ...". A sidebar on the right shows "Panel view" and "Query builder" options.

Firebase CS481-Final Project

Project Overview Authentication

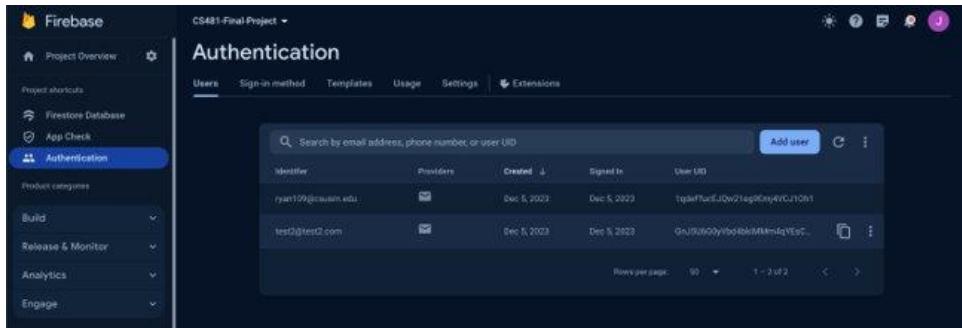
Users Sign-in method Templates Usage Settings Extensions

Identifier Providers Created Signed In User ID

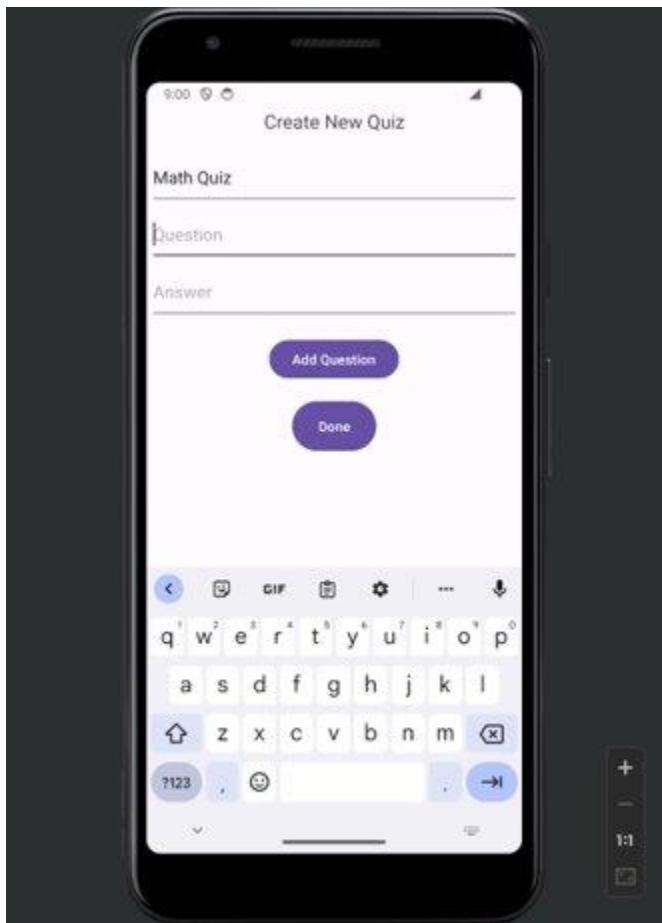
ryan109@csusm.edu		Dec 5, 2023	Dec 5, 2023	tpqfPnEJQw2legR6m4VUJ0N
test0@test0.com		Dec 5, 2023	Dec 5, 2023	GnJ996dyYbd4kM4m4gYeG...

Add user

Rows per page: 50 1 - 2 of 2 < >



- CreateQuiz



The screenshot shows the Firebase Firestore interface. On the left, there's a sidebar with a 'quizzes' collection and a list of document IDs: KdV0h0Zw623ZmaIf90qM, NnMz0ZEV31XkkFUKHcm, bjBAay5qs6Tu0teNqDtb, jJoV1gRCvA4DDj1FNhMm, ozWuPH8huD003FSGX4jG, and zQFeKn0PoJ0Ek6aBIcVC. The main area displays a single document with the ID 'B3pE4MMiTb9SmokhQmhM'. This document has two fields: 'answers' and 'questions'. The 'answers' field contains three items: index 0 with value "8", index 1 with value "12", and index 2 with value "8". The 'questions' field also contains three items: index 0 with value "3+9", index 1 with value "3+9", and index 2 with value "3+9". At the bottom of the document, it says 'title: "Math Quiz"'.

- EditQuiz

This screenshot is identical to the one above, showing the same document structure and data in the Firebase Firestore console. The document 'B3pE4MMiTb9SmokhQmhM' contains fields 'answers' and 'questions' with their respective values and indices.

- TakeQuiz
- Scoreboard
- Firebase

Home > quizzes > B3pE4MMiTb9S.

More in Google Cloud

(default)	quizzes	B3pE4MMiTb9SmokhQmhM
+ Start collection	+ Add document	+ Start collection
quizScores	B3pE4MMiTb9SmokhQmhM	+ Add field
quizzes >	KdVOH8Zw62zmaTf8qM	- answers
users	LTmYQz36gRb9vF04gVlo	0 "4"
	MN19nLSIk4Zqen2nJY1Z	1 "12"
	NnMz0ZEV31XkkFUKHKcm	2 "9"
	Z9t1csSh1KAhsxK90GKW	+ questions
	bjBAay5qs6TuOteNqDtb	0 "3+1"
	jJoV1gRCvA4DDj1FNhMm	1 "3+9"
	ozWuPH8huD003FSGX4jG	2 "3+6"
	whQwiniUUJYnzXuSMQZxc	title: "Math Quiz"
	zQFeKn0PoJ0Ek6aBicVC	

Home > quizScores > 1ymqAycloRqm.

(default)	quizScores	1ymqAycloRqmxE3ZvyU
+ Start collection	+ Add document	+ Start collection
quizScores >	0eh4ktD7Wz0RPLVYy2D	+ Add field
quizzes	0yb42XHy0Z8n3iI6LgvD	Score: 1
users	18kNjaBi7vwk2nKRFVqs	quizID: "Math Quiz"
>	1ymqAycloRqmxE3ZvyU >	username: "Ryan109"
	220fYFQz3xvSFRDv8GvJ	
	3uqwH5bYmsJkk8QTJHe9	
	7bKHJCKWcji6rqBVhwJF	

Home > users > 1qdefFucEJQw2.

(default)	users	1qdefFucEJQw21eg9Emj4VCJ1Oh1
+ Start collection	+ Add document	+ Start collection
quizScores	1KdDVwurHqPkq1PoHbA5	+ Add field
quizzes	1qdefFucEJQw21eg9Emj4VCJ1Oh1 >	Email: "ryan109@csusm.edu"
users >	3K1Ycs7SmvuMxCBw7Vu	First Name: "Jared"
	D5dJYo5R10pPB0012nfn	Last Name: "Ryan"
	DgQPv4sC1CtLCI7bprQr	Username: "Ryan109"
	GnJ5U6GByVbd4bk1MMm4qYE5CPX2	
	UnDrvduTxMnMFKXzzhSz	
	du3Gxc01RCZI23FHlJr	
	eNkK1j8d6nANB9J9j5Y9	
	e0qOGiHMJH5yqs8AxHpo	
	gls84ACUks0jbBTnx3t	
	1PIKCIGITZgBnTBMAPo3	
	zSomz1EEIUMzY6fzFinI	

- Notification
- UI Redesign







- Permission

## UNIT TESTS

### Edit Quiz Adapter Test

- getItemCount

```
QuizzesFragment
QuizzesViewModel
Second2Fragment
SecondFragment
scoreboard
ScoreboardFragment
ScoreboardViewModel
slideshow
SlideshowFragment
    @Test
    fun getItemCount_ReturnsCorrectCount() {
        val expectedCount = editQuizAdapter.itemCount
        assertEquals(expectedCount, actual) // Header + Question1 + Question2
    }
}
```

A screenshot of a code editor showing Java test code. The code is part of a class named `EditQuizAdapter` and contains a single test method `getItemCount\_ReturnsCorrectCount`. The code uses annotations like `@Test` and `@UiThread` along with `RecyclerView` and `Mockito` imports. The code itself compares the expected item count (calculated as 1 for the header plus 2 for questions) with the actual item count from the adapter.

Run: EditQuizAdapterTest.getItemCount\_ReturnsCorrectCount

Test Results 19 sec 51ms Executing tasks: [:app:testDebugUnitTest, --tests, com.hfad.finalproject\_team\_temp.EditQuizAdapterTest.getItemCount\_ReturnsCorrectCount] in project /Users/oliviasheehan/Downloads/finalproject-team-temp-main\_3

```
> Task :app:preBuild UP-TO-DATE
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:mergeDependencyArtifactsDebug 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: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:preDebugUnitTestBuild UP-TO-DATE
```

BUILD SUCCESSFUL in 27s  
25 actionable tasks: 2 executed, 23 up-to-date  
Build Analyzer results available  
2:44:58 PM: Execution finished ':app:testDebugUnitTest --tests "com.hfad.finalproject\_team\_temp.EditQuizAdapterTest.getItemCount\_ReturnsCorrectCount"'.

- Get Count Incorrect (Intended to Fail)

```
    }
    @Test
    fun getItemCount_ReturnsIncorrectCount() {
        val expectedCount = editQuizAdapter.itemCount
        assertEquals(expectedCount, actual: 2) // Header + Question1 + Question2
    }
    /*@Test
```

Run: EditQuizAdapterTest.getItemCount\_ReturnsIncorrectCount

Test Results 12 sec 601ms Executing tasks: [:app:testDebugUnitTest, --tests, com.hfad.finalproject\_team\_temp.EditQuizAdapterTest.getItemCount\_ReturnsIncorrectCount] in project /Users/oliviasheehan/Downloads/finalproject-team-temp-main\_3

```
> Task :app:preBuild UP-TO-DATE
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:mergeDependencyArtifactsDebug 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: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:preDebugUnitTestBuild UP-TO-DATE
```

```

Run: EditQuizAdapterTest.getItemCount_ReturnsincorrectCount
Test Results 12 sec 601ms
com.hfad.finalproject 12 sec 601ms
> Task :app:processDebugResources UP-TO-DATE
> Task :app:compileDebugUnitTestKotlin
> Task :app:processDebugUnitTestJavaWithJavac NO-SOURCE
> Task :app:compileDebugUnitTestJavaWithJavac UP-TO-DATE
> Task :app:testDebugUnitTest
WARNING: No manifest file found at ./AndroidManifest.xml.
 Falling back to the Android OS resources only.
 To remove this warning, annotate your test class with @Config(manifest=Config.NONE).
No such manifest file: ./AndroidManifest.xml
Properties file org/powermock/default.properties is found in 2 places:
ConfigurationSource[location:file:/Users/oliviasheehan/.gradle/caches/modules-2/files-2.1/org/powermock/powermock-core/2.0
.9/50ed2652fd31ee9c33919dfadd44504a582210/powermock-core-2.0.9.jar!/org/powermock/default.properties]
ConfigurationSource[location:file:/Users/oliviasheehan/.gradle/caches/modules-2/files-2.1/org/powermock/powermock-core/2.0
.9/50ed2652fd31ee9c33919dfadd44504a582210/powermock-core-2.0.9.jar!/org/powermock/default.properties]. Which one will be used
is undefined. Please, remove duplicated configuration file (or second PowerMock jar file) from class path to have stable
tests.Properties file org/powermock/default.properties is found in 2 places:
ConfigurationSource[location:file:/Users/oliviasheehan/.gradle/caches/modules-2/files-2.1/org/powermock/powermock-core/2.0
.9/50ed2652fd31ee9c33919dfadd44504a582210/powermock-core-2.0.9.jar!/org/powermock/default.properties]

expected:<3> but was:<2>
Expected :3
Actual   :2
<Click to see differences>

junit.framework.AssertionFailedError Create breakpoint: expected:<3> but was:<2> <5 internal lines>

```

- Update Question

```

Run: EditQuizAdapterTest.updateQuestionAtPosition_Moves...
Test Results 12sec 373ms
Executing tasks: [:app:testDebugUnitTest, --tests, com.hfad.finalproject_team_temp.EditQuizAdapterTest
._updateQuestionAtPosition_MovesQuestionToCorrectPosition] in project /Users/oliviasheehan/Downloads/finalproject_team_temp_main_3

> Task :app:preBuild UP-TO-DATE
> Task :app:preDebugBuild UP-TO-DATE
> Task :app:datasBindingMergeDependencyArtifactsDebug 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:datasBindingMergeBaseClassesDebug UP-TO-DATE
> Task :app:checkDebugAarMetadata UP-TO-DATE
> Task :app:mapDebugSourceSetPaths UP-TO-DATE
> Task :app:createDebugCompatibleScreenManifests UP-TO-DATE
> Task :app:extractDeepLinkDebug 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:bundleDebugClassesWithRuntimeJar UP-TO-DATE
> Task :app:bundleDebugClassesToCompiledJar UP-TO-DATE
> Task :app:preDebugUnitTestBuild UP-TO-DATE

D Git  Run  Debug  Profiler  Logcat  App Quality Insights  Build  TODO  Problems  Terminal  Services  App Inspection  Layout Inspector
Run: EditQuizAdapterTest.updateQuestionAtPosition_Moves...
Test Results 12sec 373ms
Tests passed: 1 of 1 test - 12 sec 373 ms
> Task :app:prePreCompileDebugUnitTest NO-SOURCE
> Task :app:processDebugJava NO-SOURCE
> Task :app:compileDebugUnitTestKotlin
> Task :app:processDebugUnitTestJavaWithJavac NO-SOURCE
> Task :app:processDebugUnitTestJavaWithJavac UP-TO-DATE
> Task :app:testDebugUnitTest
WARNING: No manifest file found at ./AndroidManifest.xml.
 Falling back to the Android OS resources only.
 To remove this warning, annotate your test class with @Config(manifest=Config.NONE).
No such manifest file: ./AndroidManifest.xml
Properties file org/powermock/default.properties is found in 2 places:
ConfigurationSource[location:file:/Users/oliviasheehan/.gradle/caches/modules-2/files-2.1/org/powermock/powermock-core/2.0
.9/50ed2652fd31ee9c33919dfadd44504a582210/powermock-core-2.0.9.jar!/org/powermock/default.properties]
ConfigurationSource[location:file:/Users/oliviasheehan/.gradle/caches/modules-2/files-2.1/org/powermock/powermock-core/2.0
.9/50ed2652fd31ee9c33919dfadd44504a582210/powermock-core-2.0.9.jar!/org/powermock/default.properties]. Which one will be used
is undefined. Please, remove duplicated configuration file (or second PowerMock jar file) from class path to have stable
tests.Properties file org/powermock/default.properties is found in 2 places:
ConfigurationSource[location:file:/Users/oliviasheehan/.gradle/caches/modules-2/files-2.1/org/powermock/powermock-core/2.0
.9/50ed2652fd31ee9c33919dfadd44504a582210/powermock-core-2.0.9.jar!/org/powermock/default.properties]

BUILD SUCCESSFUL in 21s
25 actionable tasks: 2 executed, 23 up-to-date

Build Analyzer results available
4:18:37 PM: Execution finished ':app:testDebugUnitTest --tests "com.hfad.finalproject_team_temp.EditQuizAdapterTest
._updateQuestionAtPosition_MovesQuestionToCorrectPosition"'.

```

- Update Answer at Position

The screenshot shows two panels from Android Studio. The top panel is the 'Test Results' tab, which displays a single test named 'Tests passed: 1 of 1 test - 24 sec 35 ms'. The bottom panel is the 'Build Log' tab, which shows the command run: ':app:testDebugUnitTest --tests com.hfad.finalproject\_team\_temp.EditQuizAdapterTest'. The log output details the build process, including tasks like 'preBuild', 'preDebugBuild', and various resource processing steps. It also includes a warning about a manifest file and a note about PowerMock properties being found in multiple locations. The log concludes with 'BUILD SUCCESSFUL'.

- 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)

Run: EditQuizAdapterTest.updateAnswerAtPosition\_MovesAn... ▾

Tests failed: 1 of 1 test – 13 sec 217 ms

com.hfad.finalproject\_13sec 717 ms

com.hfad.finalproject\_13sec 717 ms

com.hfad.finalproject\_13sec 717 ms

```
junit.framework.ComparisonFailure Create breakpoint : expected:<[UpdatedAnswer]> but was:<[Question1]> <2 internal lines>
    at com.hfad.finalproject_team_temp.EditQuizAdapterTest.updateAnswerAtPosition_MovesAnswerToIncorrectPosition
        (EditQuizAdapterTest.kt:3) <10 internal lines>
    at org.robolectric.RobolectricTestRunner$1.evaluate(RobolectricTestRunner.java:591)
    at org.robolectric.internal.SandboxTestRunner$2.lambda$evaluate$0(SandboxTestRunner.java:274)
    at org.robolectric.internal.bytecode.Sandbox.Lambda$runOnMainThread$0(Sandbox.java:86) <4 internal lines>

com.hfad.finalproject_team_temp.EditQuizAdapterTest > updateAnswerAtPosition_MovesAnswerToIncorrectPosition FAILED
    junit.framework.ComparisonFailure at EditQuizAdapterTest.kt:95
1 test completed, 1 failed
> Task :app:testDebugUnitTest FAILED
FAILURE: Build failed with an exception.
* What went wrong:
Execution failed for task ':app:testDebugUnitTest'.
> There were failing tests. See the report at: file:///Users/oliviasheehan/Downloads/finalproject-team-temp-main%2B3/app/build/reports/tests/testDebugUnitTest/index.html
* Try:
> Run with --stacktrace option to get the stack trace.
> Run with --info or --debug option to get more log output.
> Run with --scan to get full insights.
* Get more help at https://help.gradle.org
BUILD FAILED in 23s
25 actionable tasks: 2 executed, 23 up-to-date
1:53:03 PM: Execution finished ':app:testDebugUnitTest --tests "com.hfad.finalproject_team_temp>EditQuizAdapterTest.updateAnswerAtPosition_MovesAnswerToIncorrectPosition"'.
```

Low disk space  
Less than 50 MB is left on the system directory partition ('/dev/disk1s')

Run: EditQuizAdapterTest.updateQuestionAtPosition\_Moves... ▾

Tests failed: 1 of 1 test – 15 sec 45 ms

Task :app:processDebugUnitTestsJava UN-UN-UN-UN

Task :app:testDebugUnitTest

com.hfad.finalproject\_15 sec 45 ms

com.hfad.finalproject\_15 sec 45 ms

```
WARNING: No manifest file found at /AndroidManifest.xml.
Falling back to the Android OS resources only.
To remove this warning, annotate your test class with @Config(manifest=Config.NONE).
No such manifest file: /AndroidManifest.xml
Properties file org/powermock/default.properties is found in 2 places:
ConfigurationSource location=file:/Users/oliviasheehan/.gradle/caches/modules-2/files-2.1/org.powermock/powermock-core/2.0.9/98e5d2652fd31ee9c33919fadd44504a582210/powermock-core-2.0.9.jar!/org/powermock/default.properties
ConfigurationSource location=file:/Users/oliviasheehan/.gradle/caches/modules-2/files-2.1/org.powermock/powermock-core/2.0.9/98e5d2652fd31ee9c33919fadd44504a582210/powermock-core-2.0.9.jar!/org/powermock/default.properties Which one will be used is undefined. Please, remove duplicated configuration file (or second PowerMock jar file) from class path to have stable tests.Properties file org/powermock/default.properties is found in 2 places:
ConfigurationSource location=file:/Users/oliviasheehan/.gradle/caches/modules-2/files-2.1/org.powermock/powermock-core/2.0.9/98e5d2652fd31ee9c33919fadd44504a582210/powermock-core-2.0.9.jar!/org/powermock/default.properties

Index 3 out of bounds for length 3
java.lang.IndexOutOfBoundsException Create breakpoint : Index 3 out of bounds for length 3 <3 internal lines>
    at java.base/java.util.Objects.checkIndex(Unknown Source)
    at java.base/java.util.ArrayList.get(Unknown Source)
    at com.hfad.finalproject_team_temp.EditQuizAdapterTest.updateQuestionAtPosition_MovesQuestionToIncorrectPosition
        (EditQuizAdapterTest.kt:3) <10 internal lines>
    at org.robolectric.RobolectricTestRunner$1.evaluate(RobolectricTestRunner.java:591)
    at org.robolectric.internal.SandboxTestRunner$2.lambda$evaluate$0(SandboxTestRunner.java:274)
    at org.robolectric.internal.bytecode.Sandbox.Lambda$runOnMainThread$0(Sandbox.java:86) <4 internal lines>

com.hfad.finalproject_team_temp.EditQuizAdapterTest > updateQuestionAtPosition_MovesQuestionToIncorrectPosition FAILED
    java.lang.IndexOutOfBoundsException at EditQuizAdapterTest.kt:109
```

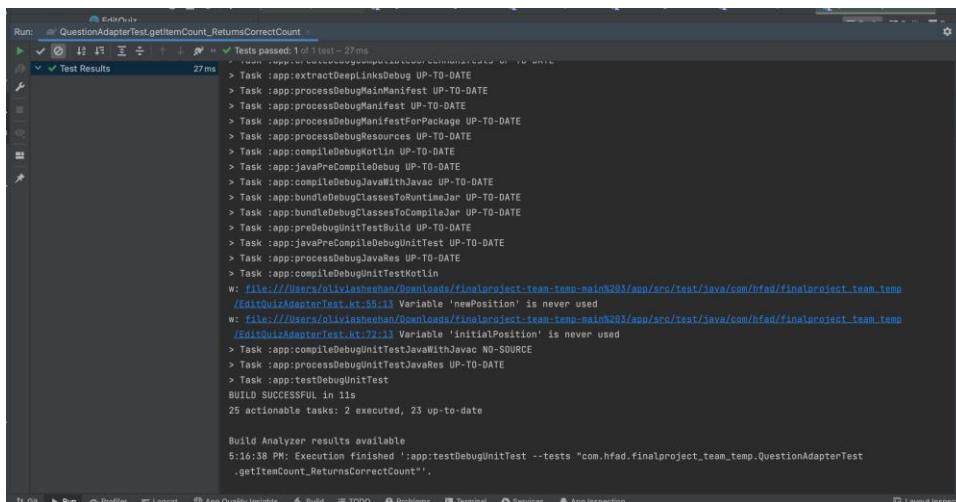
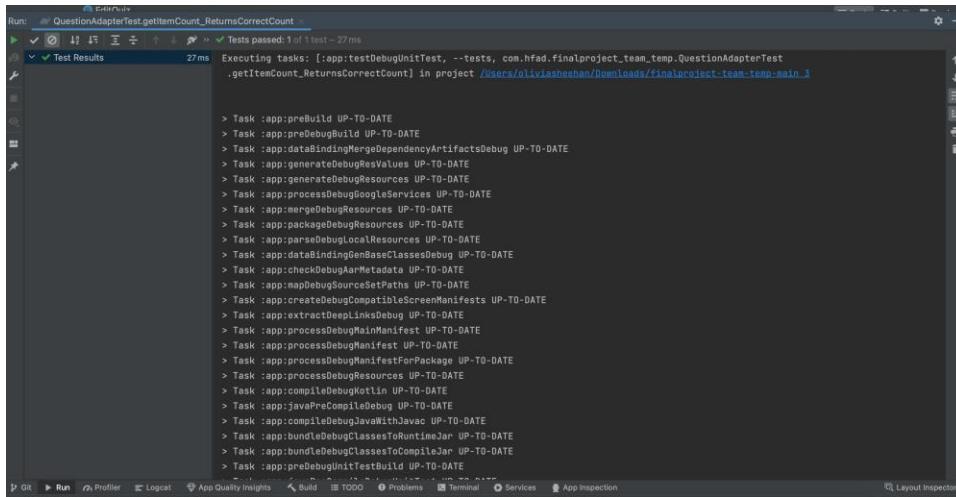
109:57 1F 1T-E 4 spaces 0

Tests failed: 1 passed: 0 (no tests app)

## Question Adapter

- Get Count

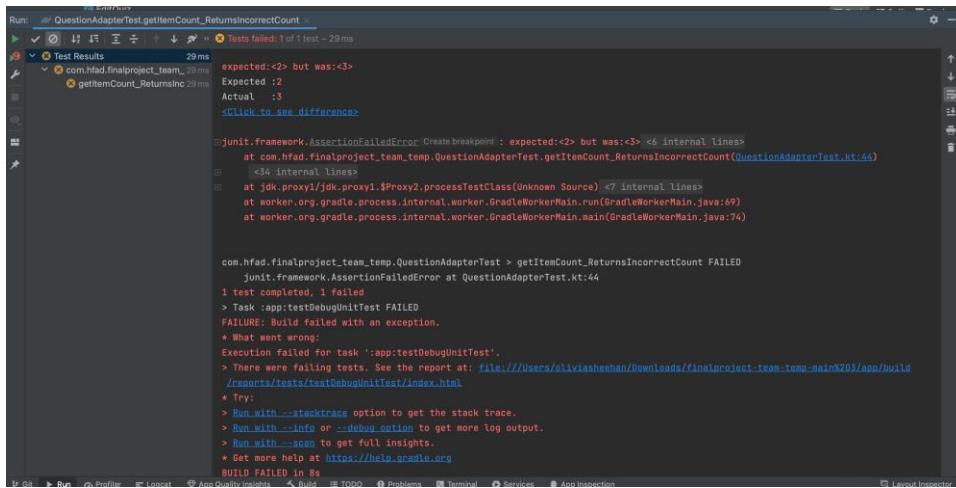
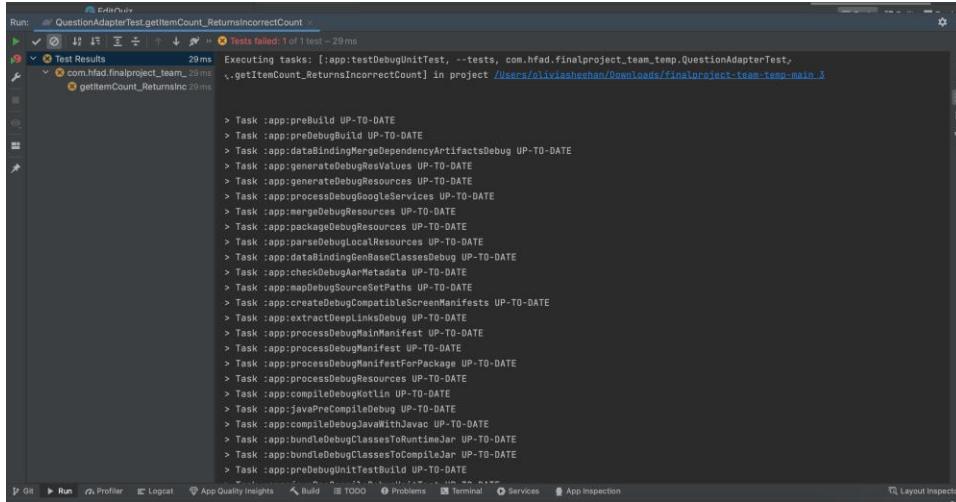
```
32     //editQuizAdapter.onAttachedToRecyclerView(recyclerViewMock)
33 }
34
35 @Test
36 fun getItemCount_ReturnsCorrectCount() {
37     val expectedCount = questionAdapter.itemCount
38     TestCase.assertEquals(expectedCount, actual: 2) //Question1 + Question2
39 }
40
41 @Test
```



- Get Incorrect Count (intended to fail)

```
40
41     @Test
42     fun getItemCount_ReturnsIncorrectCount() {
43         val expectedCount = questionAdapter.itemCount
44         TestCase.assertEquals(expectedCount, actual) // Question1 + Question2
45     }
46 }
```

The screenshot shows the Java code for the `getItemCount_ReturnsIncorrectCount` test. The test uses the `@Test` annotation and asserts that the `itemCount` property of the `questionAdapter` object equals the value 3. A comment in the code specifies that the expected count is the sum of `Question1` and `Question2`.



- Binds Correct Question

```

    }
}

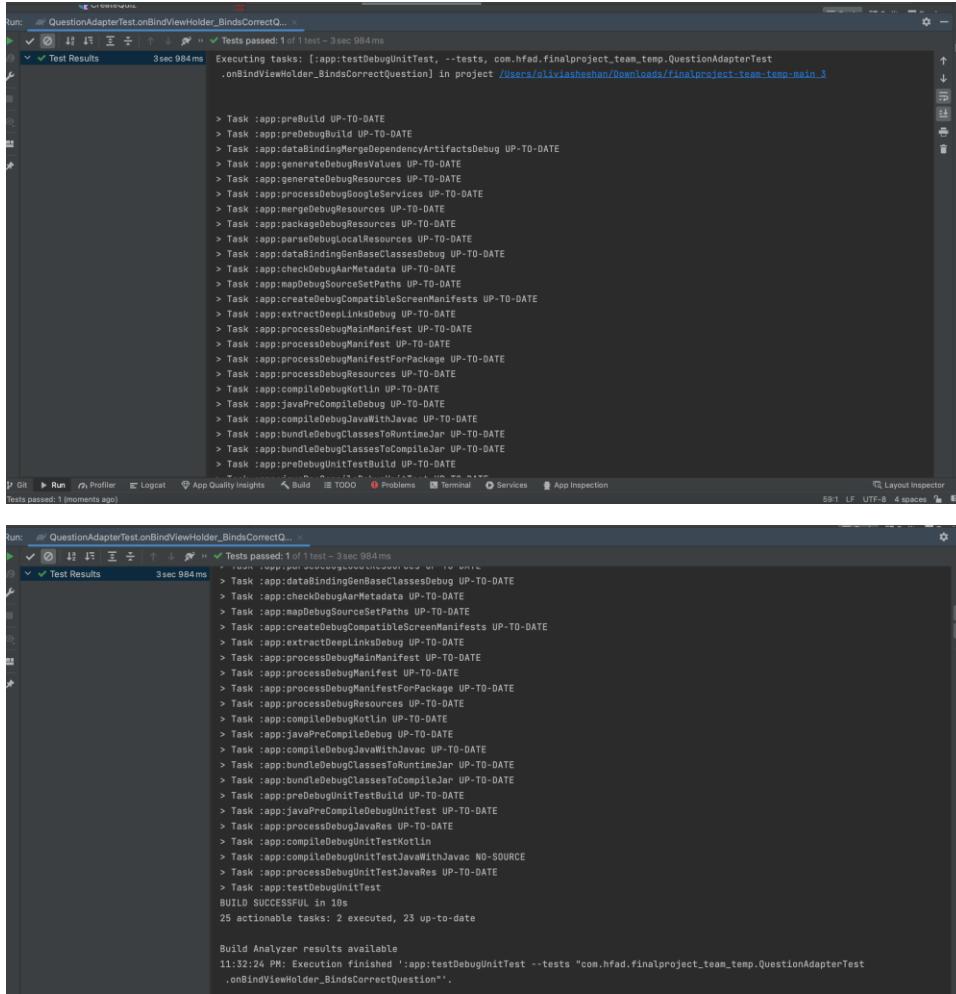
@Test
fun onBindViewHolder_BindsCorrectQuestion() {
    // Arrange
    `when`(mockView.findViewById(com.hfad.finalproject_team_temp.R.id.textQuestion)).thenReturn(mockTextView)
    `when`(mockTextView.text).thenReturn(value = "Question1") // Mocking the behavior of TextView

    val adapter = QuestionAdapter(listOf("Question1", "Question2", emptyList()))
    val viewHolder = adapter.ViewHolder(mockView)

    // Act
    adapter.onBindViewHolder(viewHolder, position = 0)

    // Assert
    assertEquals(expected = "Question1", viewHolder.questionTextView.text)
}

```



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

```

    }
    @Test
    fun onBindViewHolder_BindsInCorrectQuestion() {
        // Arrange
        `when`(`mockView`.findViewById<TextView>(com.hfad.finalproject_team_temp.R.id.textQuestion)).thenReturn(`mockTextView`)
        `when`(`mockTextView`.text).thenReturn(value = "Question1") // Mocking the behavior of TextView

        val adapter = QuestionAdapter(listOf("Question1", "Question2"), emptyList())
        val viewHolder = adapter.ViewHolder(`mockView`)

        // Act
        adapter.onBindViewHolder(viewHolder, position: 3)

        // Assert
        assertEquals(expected = "Question1", viewHolder.questionTextView.text)
    }
}

```

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/oliviashuehan/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:bindDebugGeneratedClassesDebug UP-TO-DATE
> Task :app:checkDebugAarMetadata UP-TO-DATE
> Task :app:mapDebugSourceSetPaths UP-TO-DATE
> Task :app:createDebugCompatibleScreenManifests UP-TO-DATE
> Task :app:extractDebugRDependencies 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:processDebugManifest FAILED
> Task :app:compileDebugKotlin UP-TO-DATE
> Task :app:javaPreCompileDebug UP-TO-DATE
> Task :app:compileDebugJavaWithJavac UP-TO-DATE
> Task :app:bundleDebugClassesForRuntimeJar UP-TO-DATE
> Task :app:bundleDebugClassesForCompileJar 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/oliviashuehan/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:bindDebugGeneratedClassesDebug UP-TO-DATE
> Task :app:checkDebugAarMetadata UP-TO-DATE
> Task :app:mapDebugSourceSetPaths UP-TO-DATE
> Task :app:createDebugCompatibleScreenManifests UP-TO-DATE
> Task :app:extractDebugRDependencies 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:bundleDebugClassesForRuntimeJar UP-TO-DATE
> Task :app:bundleDebugClassesForCompileJar UP-TO-DATE
> Task :app:compileDebugUnitTestKotlin FAILED
```

Unable to save settings  
Failed to save settings. Please restart Android Studio