

LearnIt

By William Cassel, Olivia Sheehan, Simon Hausmaninger, Jared Ryan, Casey McGuan, Jacob Irwin, Maksym Bondarenko

About LearnIt

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.

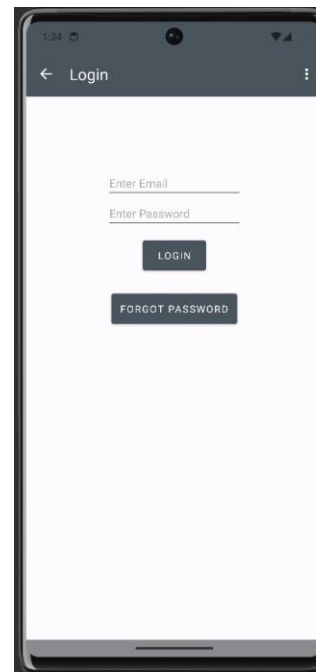
Mobile Navigation Drawer

- Makes transitioning between fragments as seamless as possible.
- The navigation drawer gives the user easy access to other fragments from wherever they are in the application.

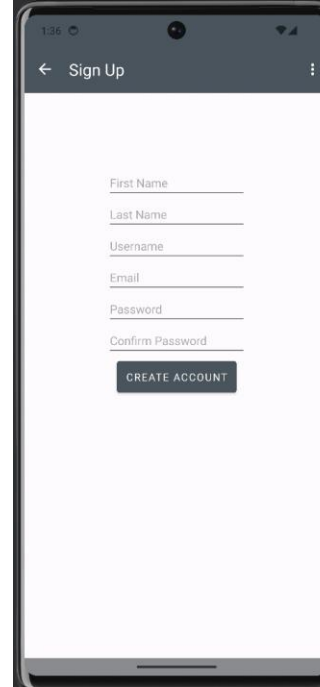


Login & Sign Up

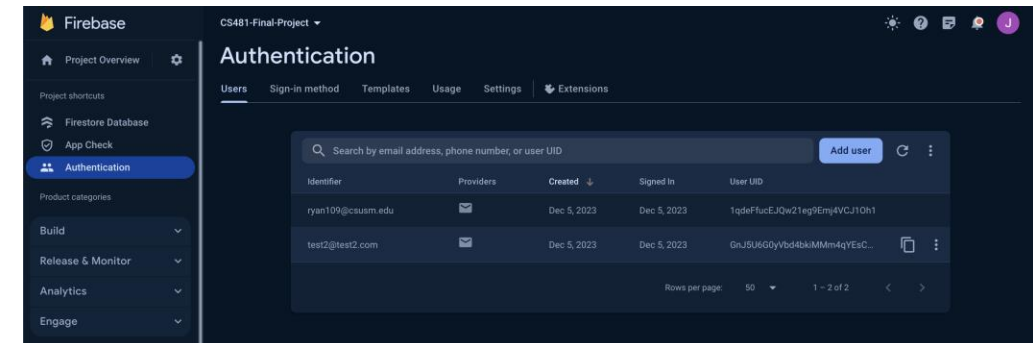
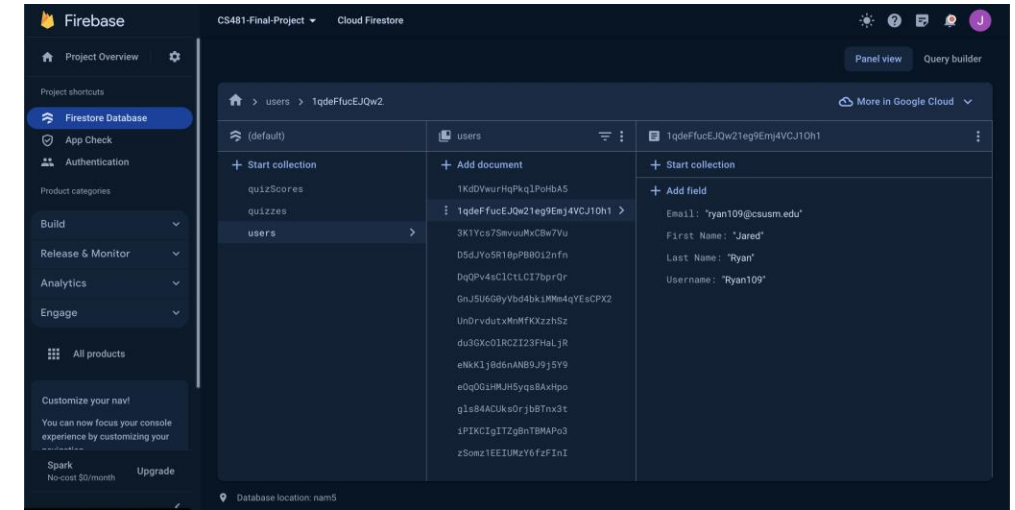
- Allows the users to create an account using Firebase Authentication to keep sensitive information private
- Forgot password functionality
- Sessions
 - Session begins when the user logs in
 - Session ends when the user signs out, then another user can log in and use their account.



A mobile app login screen with a dark header bar containing a back arrow and the word "Login". The main area is white and contains two input fields: "Enter Email" and "Enter Password". Below these fields are two buttons: "LOGIN" and "FORGOT PASSWORD".

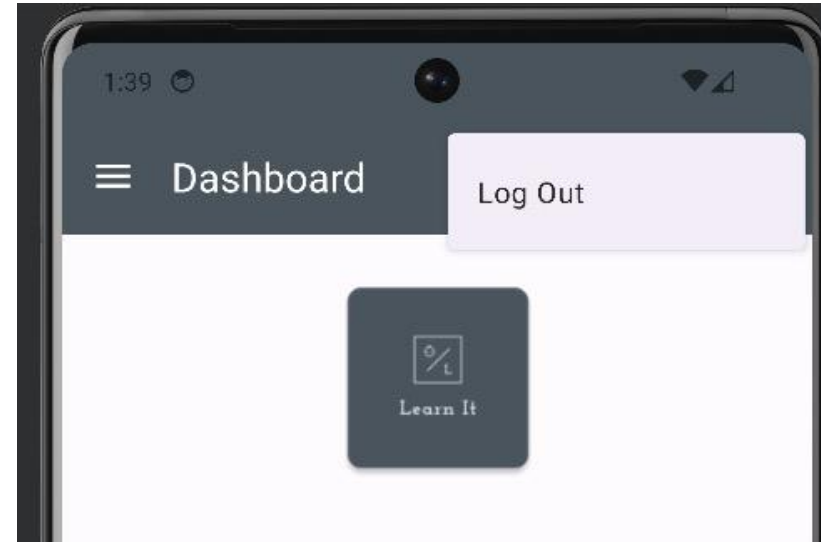


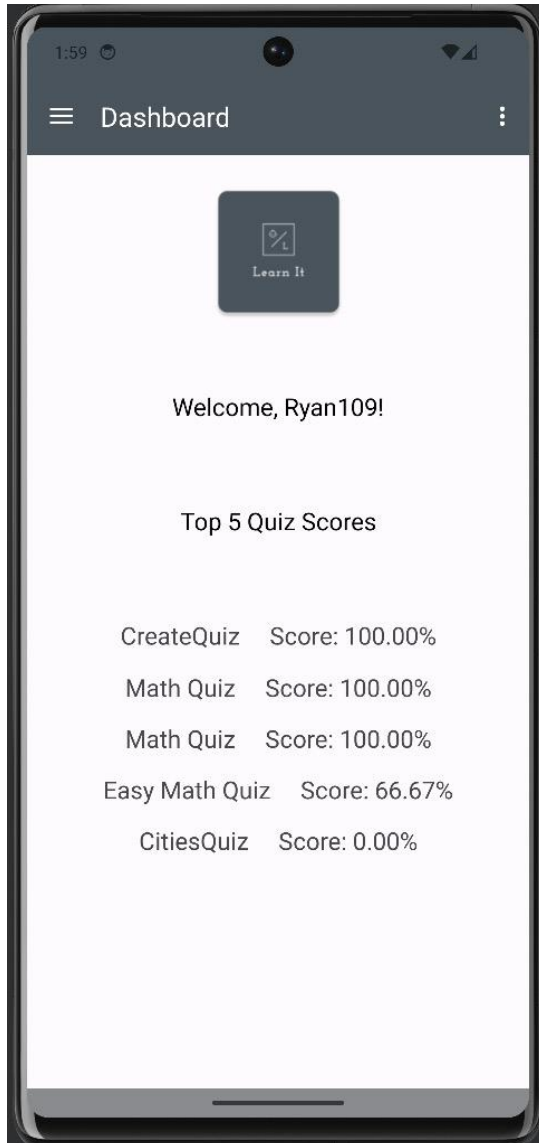
A mobile app sign up screen with a dark header bar containing a back arrow and the words "Sign Up". The main area is white and contains five input fields: "First Name", "Last Name", "Username", "Email", and "Password". Below these fields is a "Confirm Password" field and a "CREATE ACCOUNT" button.



Logout

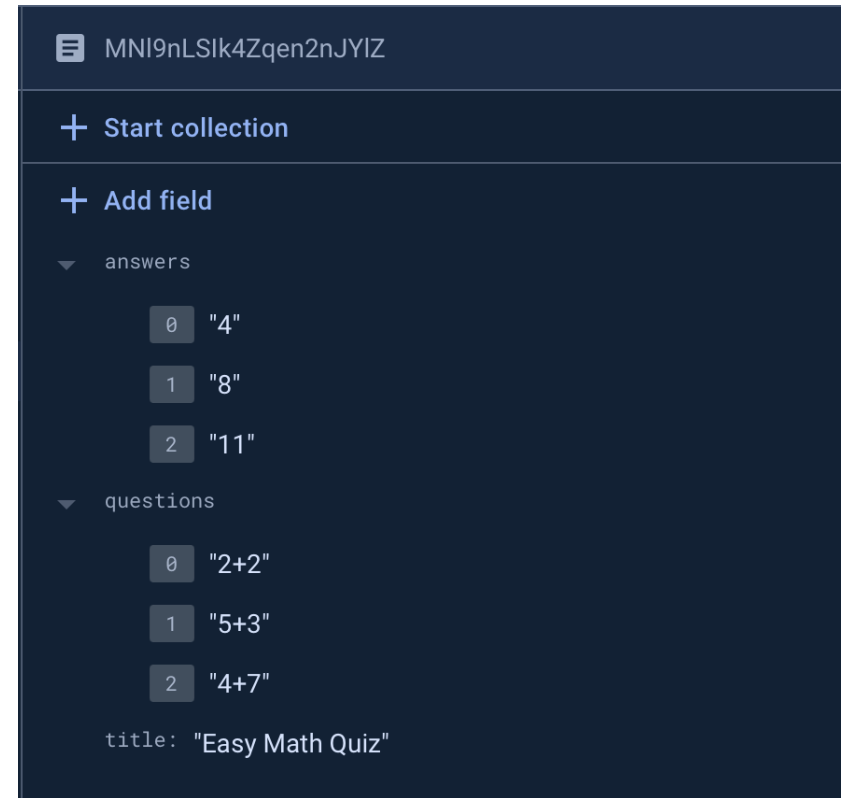
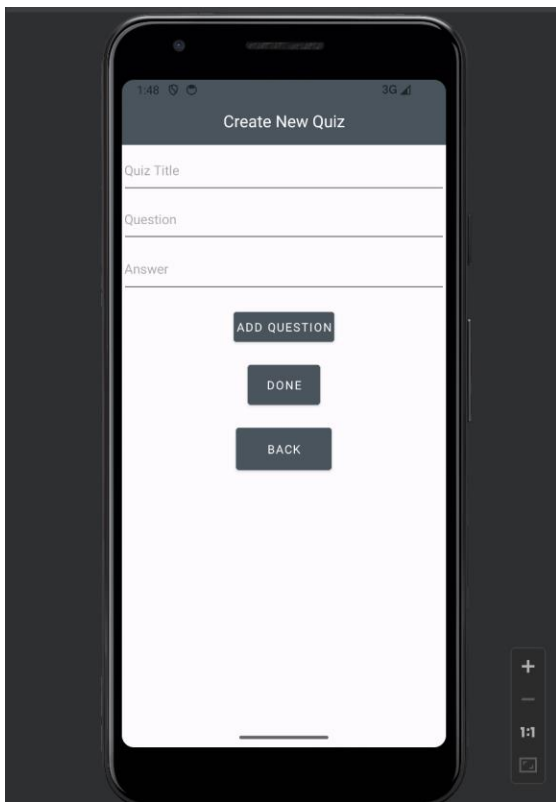
- When the user logs out, the users session is ended, the users data is cleared from the fragments and the user is sent back to the login/Signup page allowing the user to log into another account.





Dashboard

- Upon logging in, the user is sent to the Dashboard where, using mutable live data, a welcome message is displayed for the logged in users.
- When the users completes quizzes, their top five quiz scores are displayed along with the name of the quiz they took.

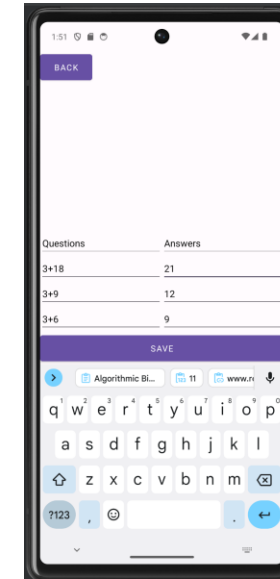


Creating Quizzes

- Allows for users to create a wide range of quizzes
- Questions and answers are stored in lists for easy access

Editing Quizzes

- Allows for users to edit quiz questions and answers
- Code snippet validates the document's existence, creates a reference to the Firestore database, and updates the specified document's fields using `quizRef.update(...)`, while also attaching success and failure listeners to log corresponding messages for successful updates or encountered errors.



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

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


Taking Quizzes

- ViewModel is used in the TakeQuiz functionality to manage dashboard related data
- Navigation:
 - The back button and next button are set to navigate back when clicked using an onClickListener
- The quiz id is used to retrieve the quiz data from the firebase database, the questions are stored in lists in the database.
- Quiz Taking
 - The next button is set up to iterate through each question and display it to the ui
 - Checks if the user's answers are correct and tracks the number of correct responses
 - Once the quiz is complete, a score is calculated
 - A score is added to the firebase as well as the quiz id and the username



```
WilliamCassel +1
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, requestCode = 0, intent, PendingIntent.FLAG_IMMUTABLE)
    }

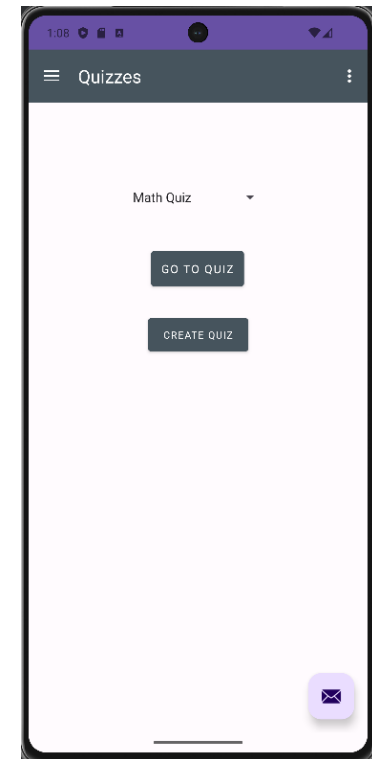
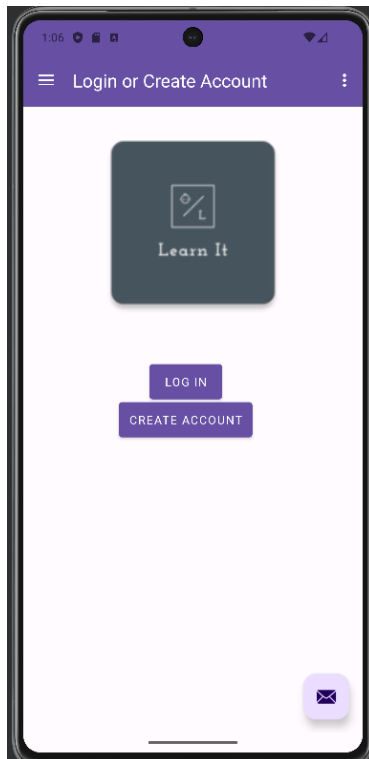
    val calendar: Calendar = Calendar.getInstance().apply { this: Calendar
        // Set the alarm to activate in 24 hours
        timeInMillis = System.currentTimeMillis()
        add(Calendar.HOUR_OF_DAY, amount = 24)
    }

    // 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
    )
}
```

24-Hour Notification System

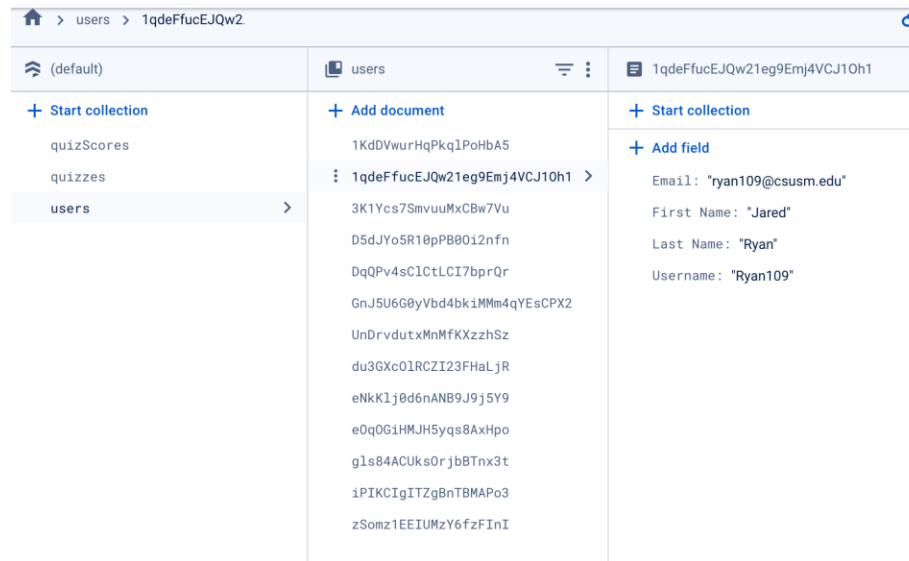
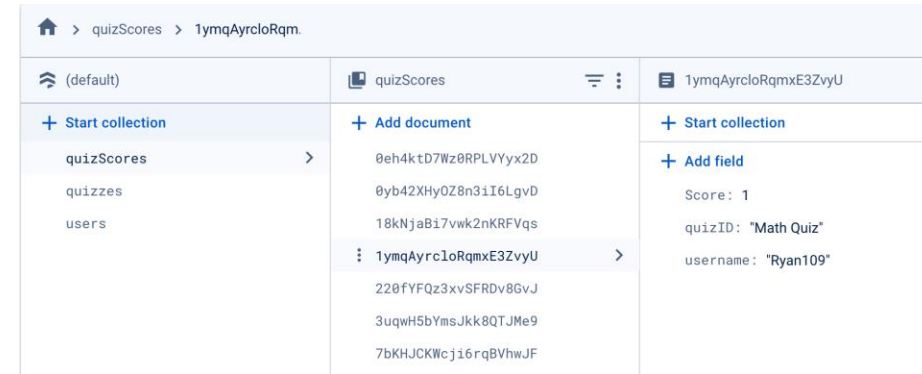
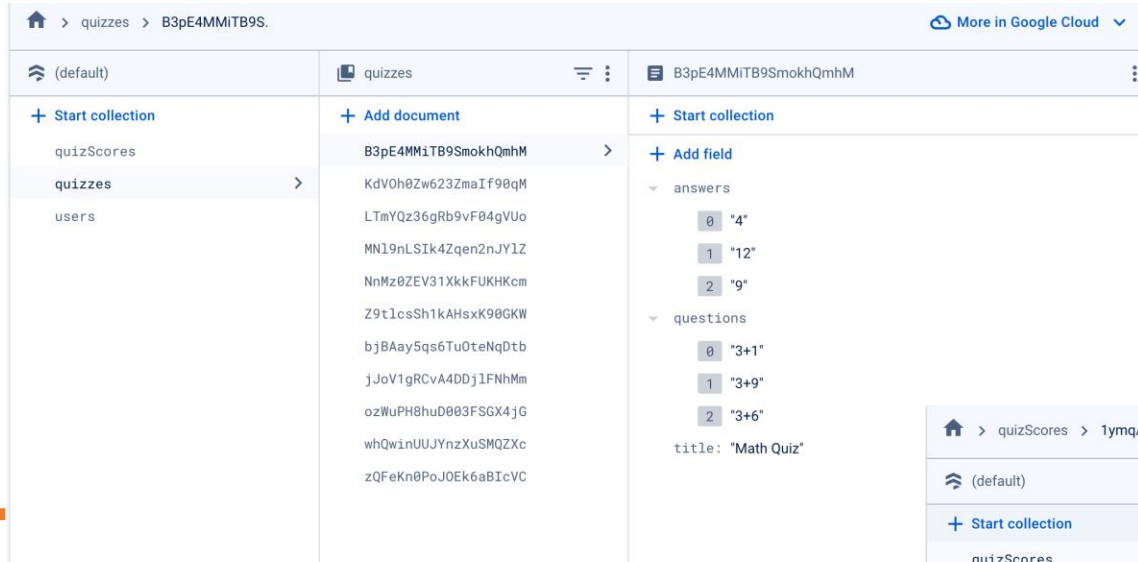
- Encourages users to build good studying habits by quizzing themselves day in and day out
- We used permission for post notifications so the user can give our app permission to notify them.
- Helps with user retention

UI Re-Design



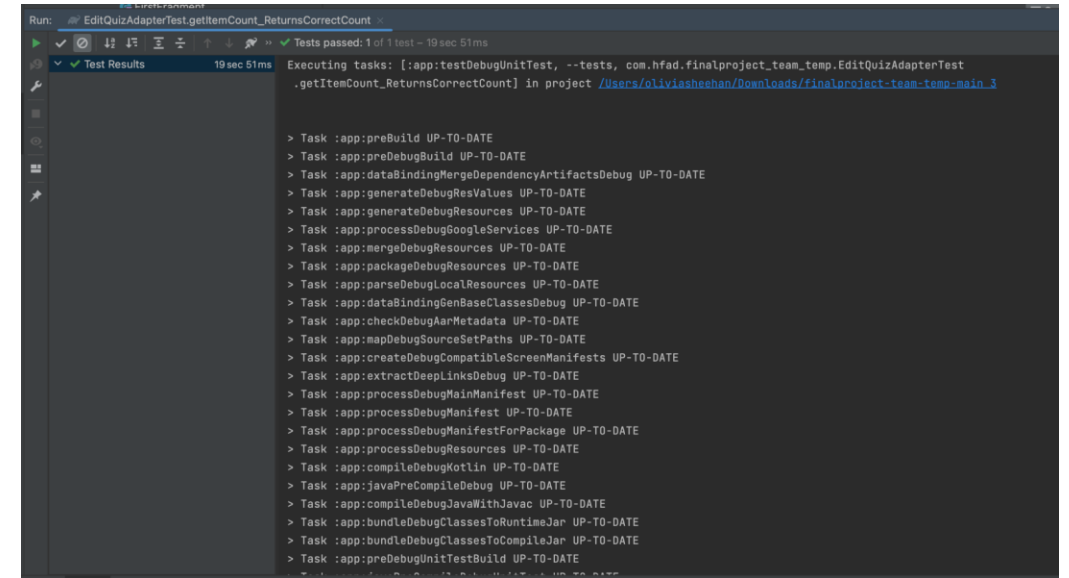
Firestore

- Simple math quiz for example
- Quizzes database for all the quizzes created and the questions/answers for that particular id are stored
- Users also have an id as well as a username so they can keep track of their quizzes and scores



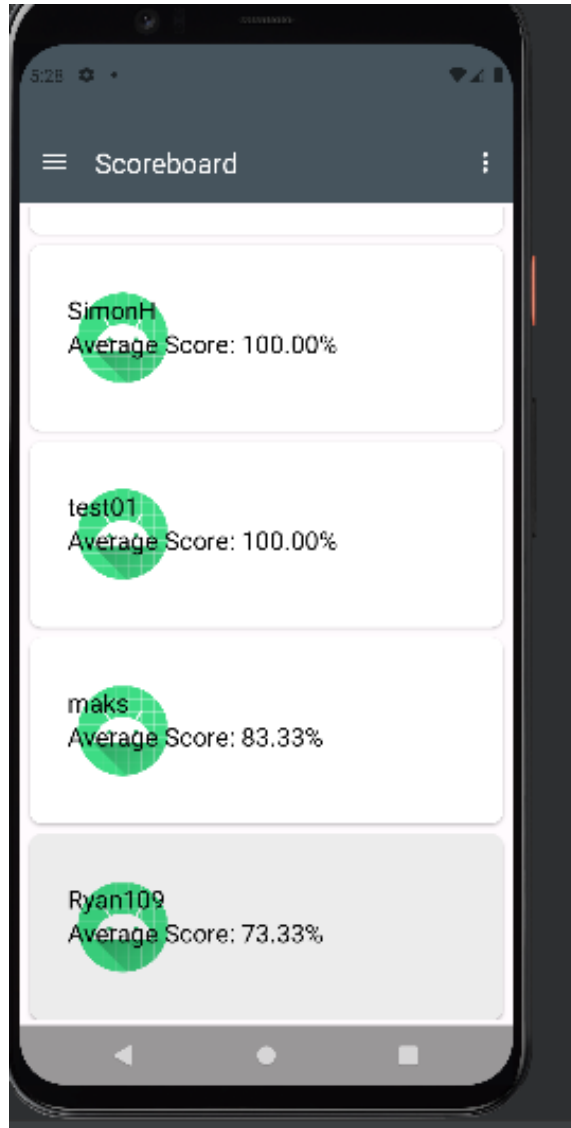
Unit Tests

- We ran unit tests for navigation, creating quizzes, editing quizzes, etc.
- We tested RecyclerView
- GetCount() for questions
- UpdateQuestion()
- UpdateAnswer()
- Out of bounds
- Tested the number of questions in the list without taking RecyclerView into account as well



```
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"'.
```



Scoreboard

Using RecyclerView I created a quiz screen showing Username and average score.

We take the values stored in the DB and output the highest 25 into our leaderboard.

Something I want to work on is implementing a icon list that users can pick from that allow them to customize their profile a bit more. This would show in the image behind the users score.

Challenges, Innovation, Merit

- Storyboard
 - Dependencies and errors arose, affecting team productivity
 - Switched to a simpler approach for create/edit quizzes
- Scoreboard
 - Switched from an activity to a fragment to combat errors
- Unit Tests
 - Since our users enter a lot of data, it was difficult to use AssertEquals
 - Tried using mock/espresso and there were a lot of dependency issues
 - Focused on running unit tests for the main activity and more straight forward functions
- Navigation
 - App was crashing on a back button press in quiz detail page resolved by adding the finish() method in the onClickListener for the back button.