 **CMPS 312 Project Phase 2**

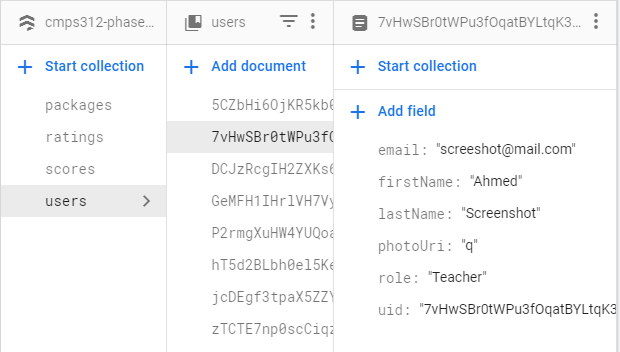


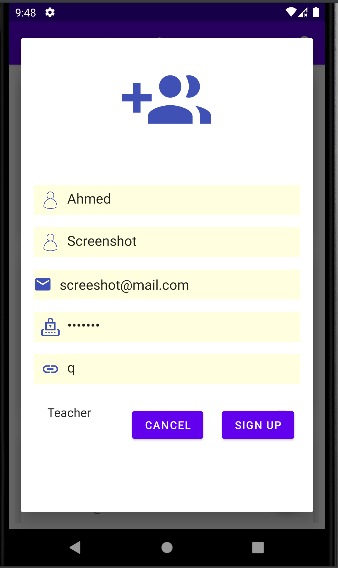
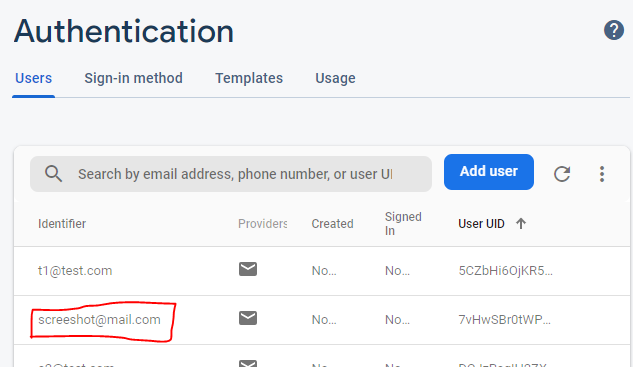
|  |  |
| --- | --- |
| **Group Id:** | G4 |
| **Group Members:** | Mohammed Abushahla 201706282  Abdulla Khedr 201702167  Hasan Alsamra 201707182  **Emails:**  [ma1706282@student.qu.edu.qa](mailto:ma1706282@student.qu.edu.qa);  [ak1702167@student.qu.edu.qa](mailto:ak1702167@student.qu.edu.qa);  [ha1707182@student.qu.edu.qa](mailto:ha1707182@student.qu.edu.qa); |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | % | Functionality\* | Quality of the implementation | Score |
| **1) Cloud Firestore Database Design and Implementation**  Repositories to interact with Firestore | 50 | Working(100) |  |  |
| **2) Download a learning package and their associated media**  - Populate the database with the learning package data fetched from Cloud Firestore.  - Download Cloud Firestore media files associated with the package to the mobile device.  - If the downloaded package already exists in the local DB, sync the the local package with the online version.  - Record *Unscamble Sentence* and *Match Definition* scores in the local DB if no internet is available then push the recorded scores to Firestore when the internet connection is available. | 30 | Working(100) |  |  |
| **3) Signup and Signin using Firebase Authentication** | 10 | Working(100) |  |  |
| **4) Design and Testing Documentation**  **\* Design documentation:**  - 4 key lessons learned from Phase 1.  - MVVM architecture diagram.  - Firestore database schema diagram and SQLite database schema diagram.  **\* Testing documentation:** with evidence of working implementation using snapshots illustrating the results of your solution testing (you must use the provided template). | 10 | Done (100%) |  |  |
| 6) **Discussion of the project contribution** of each team member [-10pts if not done] |  | Done |  |  |
| **Total** | 100 |  |  |  |
| Copying and/or plagiarism or not being able to explain or answer questions about the implementation | -  100 |  |  |  |

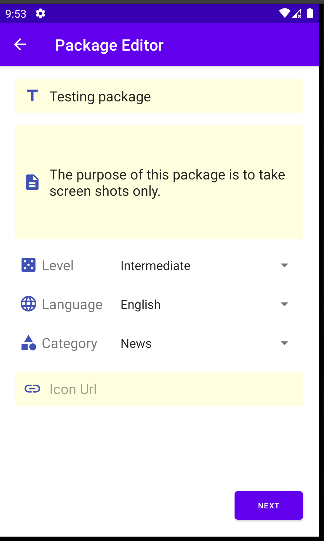
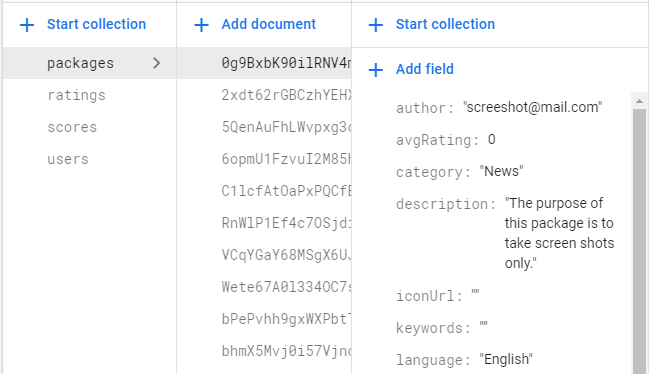
Screenshots:

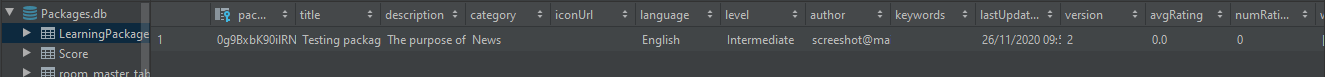
Register and adding user to firestore:

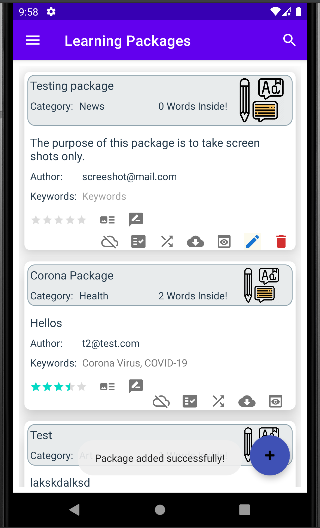
Registering new account: User added in the firestore collection and authentication users:



Adding new package: Package added to firestore:

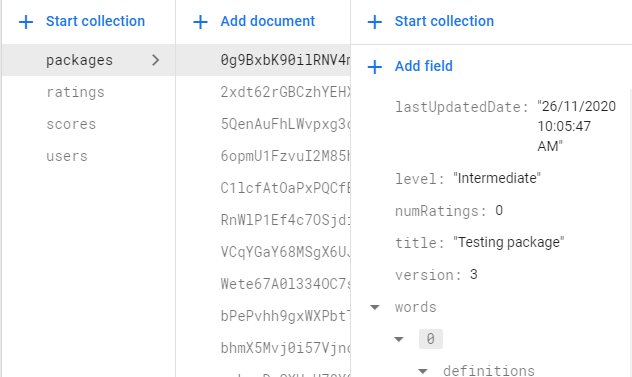


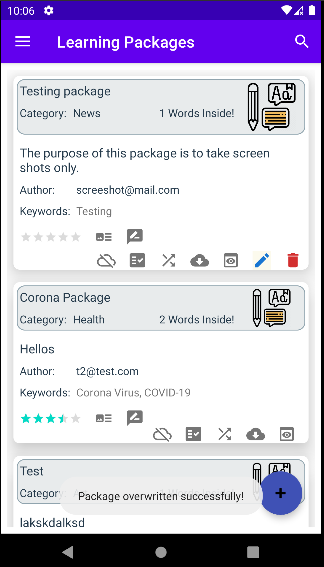
Downloading package:



After the package is downloaded, it was added to the local database from the firestore.

Syncing the package with the higher version:

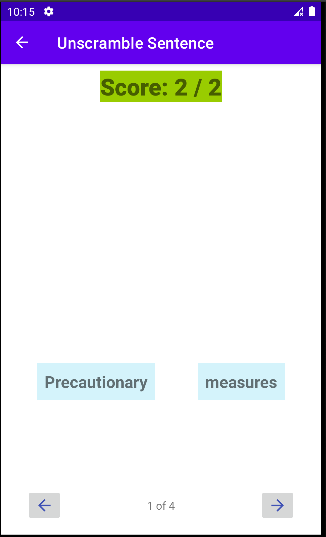
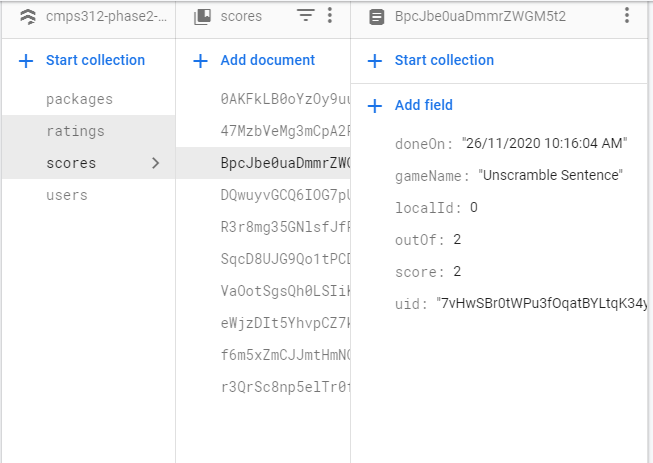
Updated the package, so it is version 3 online and 2 locally.

 After downloading the new version:After playing a game without internet, the data will be saved locally:

The version of the package after downloading it:

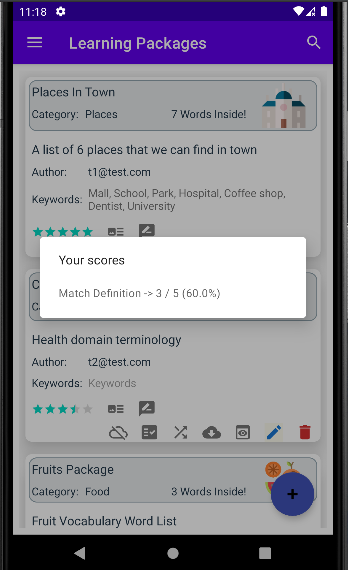
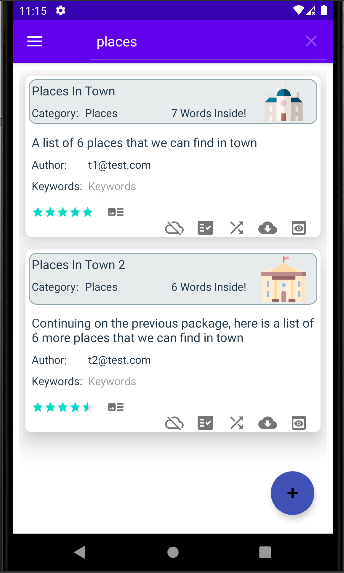
The score was added locally because there is no connection:



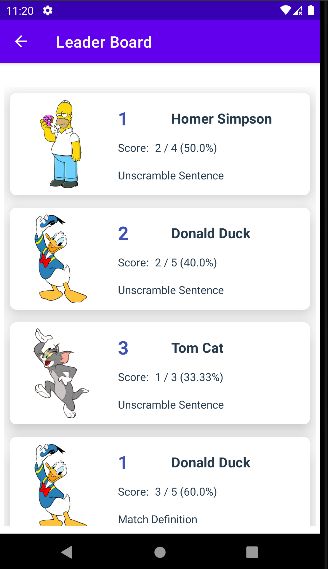


It was pushed to firestore once there is connection

Searching: View Scores:

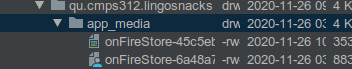


View Leaderboard:

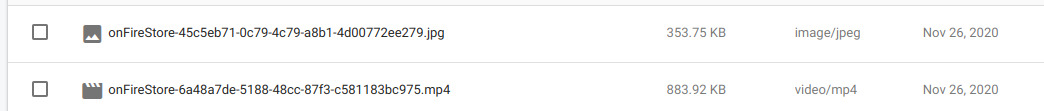


Downloading the associated media:

Locally:



Uploading the associated media:



4 Key Lessons:

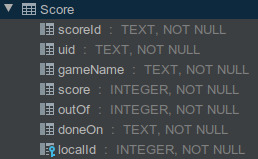
1-We learned that separating the UI from the Repository is much better, and we to make the repository communicate with the view model, and the fragments communicate with the viewmodel, so we decrease the coupling and achieve more functional and modifiable program. In phase 1 we didn't achieve this model and some of our fragments did not communicate with the viewmodel.

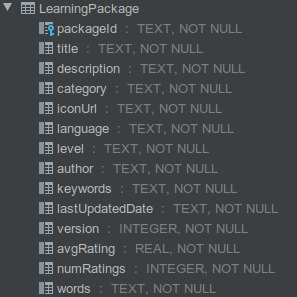
2-We learned that we can display a web page using webView that extends webViewClient() and it will load the url and display the web page, and we didn't implement this thing in phase1.

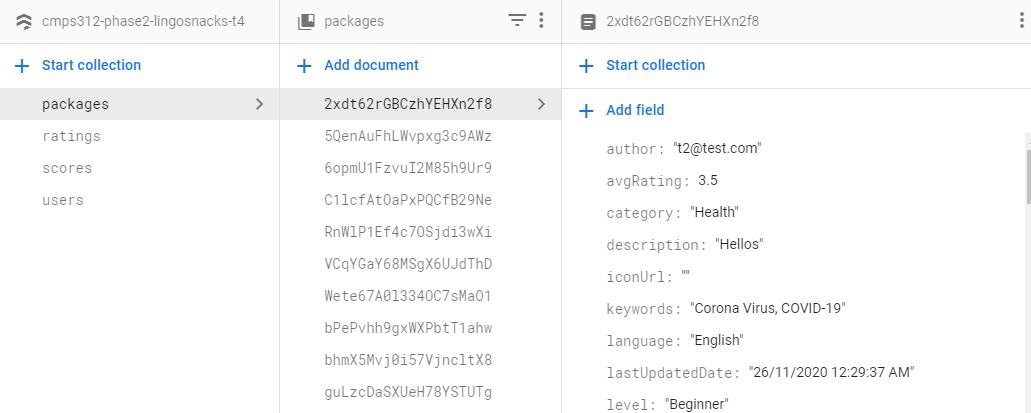
3-For adding definitions, and sentences, we did it in an inefficient, where we used a flexbox layout and inflated the strings in it, but when we saw the way it was implemented in the base solution, we learned that adding each sentence and definition in a recycler view using a button is much better.

4-For authentication, we did not implement that each user must delete and edit packages created by them, we learned that it was implemented by observing the current authenticated user and according to the id, it will show the delete and update buttons, and also it will change navigation drawer according to the current user.

SQL Database Schema:





Firestore Schema: 

MVVM Diagram: The MVVM model is a good architecture because each layer is separated from the other layers, so it enhance the modifiability of the program. Also, by having each layer to deal with only the layers above or under is better to debug and find errors.

LearnignPackage

Score

SQL

learningnPackage

user

ratings

Firestore

Model

AuthRepository

PackageRepository

PackageViewModel

PackageEditorViewModel

AuthViewModel

PackagesFragment

WordEditorFragment

PackageEditorFragment

Repository

ViewModel

UI

Students Contribution:

Student 1 (Abdullah Khedr):

He implemented the interaction between firestore and repositories, authentication, signup, and signin.

Student 2 (Hassan Alsamra):

He implemented downloading and uploading multimedia to the firestore server and part of the recording scores offline if there is no internet connection.

Student 3 (Mohammed Abushahla):

He implemented the downloading package data and checking the version to sync with the highest version, and part of the recording scores offline if there is no internet connection.

At the end we all worked together in this project, and each one wouldn’t have finished his part without the help of the others and the communication all the time.