CRITERION C : COMPLEXITY

NOTE: The pictures of the app are all given in the Appendix.

* The application follows the MVVM (model, view, view-model) architecture model as recommended in the session by Mr. Pierson and the structure is based on the template provided.
* My initial choice of technology was java swing with my SQL DB. But after my introduction to Flutter, I decided to use Flutter and Firebase. 1) I was motivated to learn new technology 2) I thought it was cool to implement a mobile application as I always wanted to understand and implement one. 3) The framework was very mature and easy to learn.
* Initially looking at the template and reading through the documentation I felt that most of the functionality is already available, and it will be very fast to implement it. I underestimated the learning curve and the shift in approach from traditional programming using Java. It took me a while to understand MVVM and the involved configurations and mapping files. The template was the savior and was the starting point for me to hit the ground running.

1. I tried to only depend on the packages and code samples from pub.dev. For example: - Flip\_card.dart , flutter\_dropdown.dart etc.
2. Registration/Login Screens allow new users to sign up and enable access to the Menu page which has options to play a game or perform card admin options. Only authenticated users can access the Menu page.
3. Currently all the users have access to all the menu options. We can extend the application to implement authorization and have access to certain pages based on the role of the user.

* Firebase Authentication:

1. I used Firebase Registration, login modules with email/password-based authentication (There are multiple other options like phone based, face or g-mail based authentication). I went with email/password-based authentication as it was simple and free
2. Once I had the login working and I learnt how all the aspects of login, sign up etc were available out of the box, I was encouraged to switch my choice of DB from mySQL to Firebase.

* Firebase DB:

**The list of users as stored in the database:**

A screenshot of a computer

Description automatically generated

**The log in method I used (email and password):**

A screenshot of a computer

Description automatically generated

* Once again, I had to struggle to understand document-based databases. I learnt how it is different from relational DB. I learnt how it is flexible and can accommodate even if the structure of the data gets modified later in the implementation. It allowed me to experiment with multiple structures. Initially I wanted to have Decks with a list of lessons and lessons with a list of flash cards. Later I optimized it to just have flash\_cards db with a list of lessons with Q/A flash cards within them.

**My database with all my data in it:**

Graphical user interface, text, application

Description automatically generated

1. My database is flash\_cards
2. Document is <lesson name>
3. Flash cards or Q/A cards are stored as sets with question as the key and answer as the value of the pair.
4. Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. This helped me avoid/save on writing code to synchronize across the clients as lot of the complexity was already handled by the framework.

NOTE: We are going to use the Registration/Login Screens to capture new users and enable access to the Teachers View for authenticated users only!

SUMMARY OF TECHNIQUES:

* **Firebase Authentication:**
* Signing up a user
* Checking for correct credentials/absent credentials
* Remembering a user and logging them in next time
* **Firebase :**
* Storing flashcards, lessons, and user-specific information
* Adding new flashcards, lessons
* Editing flashcards, lessons
* Deleting flashcards, lessons

A screenshot of a computer

Description automatically generated

I imported the flip\_card package which is unique to my code as I used to add the effect of the cards flipping on my flashcards page.

A screenshot of a computer

Description automatically generated

This adds flashcards into the database, more specifically into the lesson you want it to go into.

A screenshot of a computer

Description automatically generated

This code will shuffle the question and answer cards to prepare for the game.