FACULTY OF ENGINEERING AND APPLIED SCIENCE

Department of Electrical, Computer and

Software Engineering

SOFE 4640U/Fall 2020

Mobile Application

Development Project

Umar Qureshi

Vicki Derbyshire

Introduction & Objectives

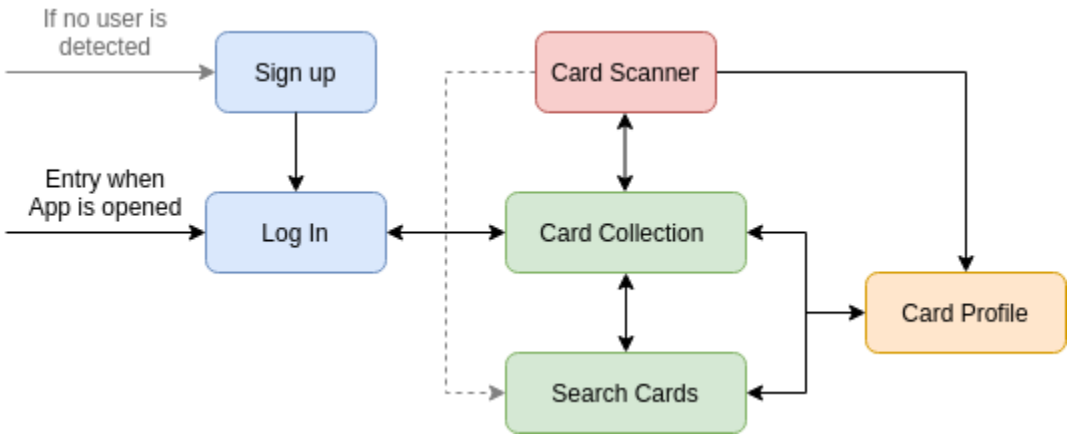
Magic the Gathering is both a collectible and digital collectible card game. The reason behind the creation of this app is to meet the goal of aiding those who enjoy playing this game to be able to keep track of all the cards that they have gathered so far. We as card collectors constantly wonder how many cards we have in our collection as well as the type of cards we have, another problem when it comes to card collection has to do with the fact that many of our beloved cards will inevitably get lost as our collection it’s bigger and bigger. Upon successful completion of this the card is then stored into our database specifically designed to easily track all the cards a person has collected.

Outcomes

The outcome our app is attempting to achieve is to build a program that is first of all, fun to use and secondly easy to use. Our main goal was to design an application that does not require more than 3 button presses in order to gain access to every feature our app has to offer. We as app lover's know from previous experiences that an app can very quickly become annoying and confusing if it is too complicated to use. In our opinion every application should be as simple as possible so that age groups of all categories can use it.

The goal we want to achieve is for our user to easily search or scan cards to read information about them and add them to their collection.

App Structure

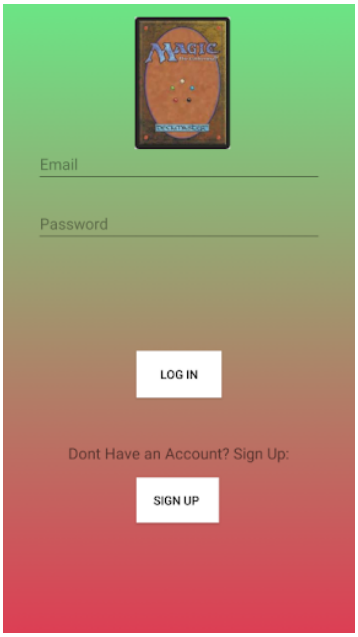


Above is the structure of our app and how the user will be able to flow through it

Main Screens

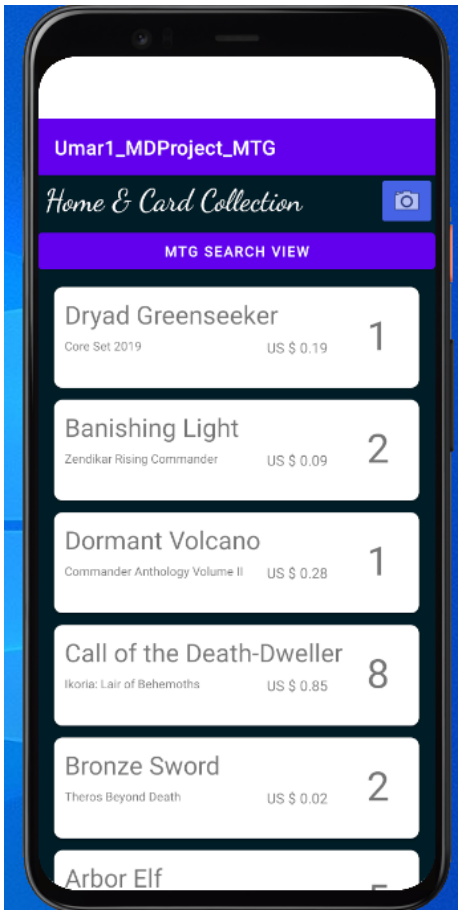
Login View:

- The user must press on the ‘Email’ text box in order to input his email
- The user must press on the ’password’ textbox in order to input his password
- The user presses the ‘LOG IN’ button in order to log into the application
- If the user does not already have an account you must press the ‘SIGN UP’ button to create one



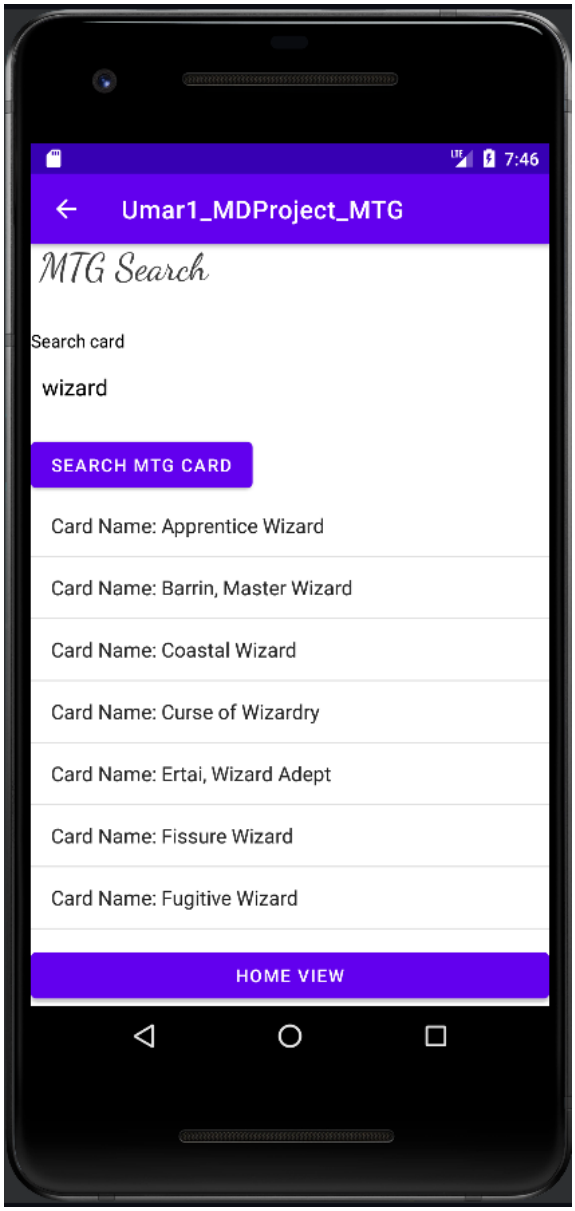
User Card Collection View:

- The user can view their card collection in a recyclerview
- The user can tap on a card and view its information
- The user can change the view to the Search View



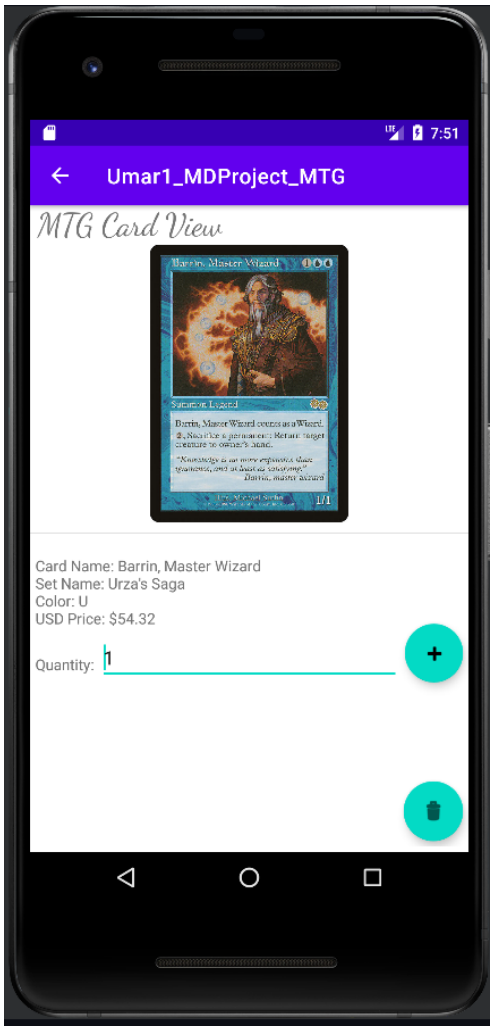
Search View:

- The user can search for cards
- When cards have been searched a user can tap on one and will be taken to the cardview page



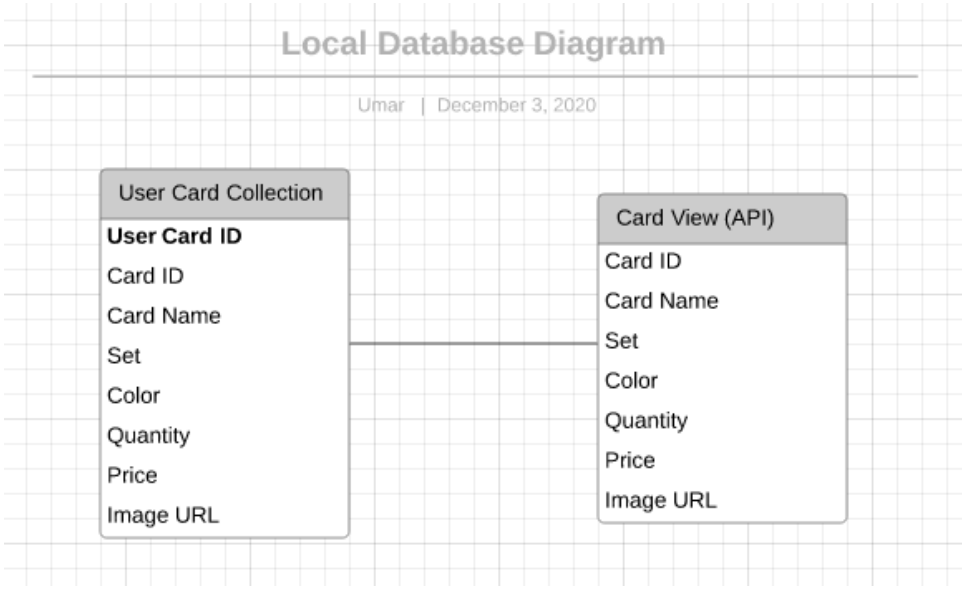
Card View:

- Shows data and picture of the card
- The plus button adds the card to the collection
- Delete button removes card from collection



Database Design:

Simple Database design where rows are populated with the when a user adds a card where data is retrieved from the api



Limitations

Our application is completely safe to use as it does not require any sensitive personal information. Such as; bank account information, SIN numbers, Living address, etc… The only information the user of our application is at risk of exposing Is his or her full name as well as email address.

Challenges Faced

Implementing the Search was difficult as scryfalls api had limited documentation. I tried many queries but no luck. I then thought maybe the website itself uses the api and I did my desired search there. I took a look at the url and I saw a query similar to the documentation. I took that query and looked at documentation again to backwards engineer it and finally got the Json file of my desired query.

After getting the Search in the listview it was difficult to take the card id from the card when tapped and send it off to the cardview page to be queried for that specific card. So I created a list of IDs after the search was executed, when a user tapped a row it would take the number of the row and use it to find the correct id in the list and send the intent over which would be used to query on cardview.

Getting the image to display was difficult, we tried many different snippets of code with no luck. We then realized that it had to be done with an Async thread. After a long time we finally found some usable code and implemented it to show the image.

Some complications are faced by actions being inputted too quickly by the user. The database interactions can be a little slow to take effect and cause adverse effects (with some crashes) unless given a delay. These could be resolved with disabling UI elements until the completion of the individual transactions. There was not quite enough time to make the process seamlessly operational with our level of experience.