Flash Card Study App

SET08114 Mobile Applications Development

Cormac Williamson

40398181

Introduction

The project given was to either create a full mobile application from scratch, utilizing several different software and design techniques, or continue development on an existing travel app. The former was chosen for the scope of the project. The app idea chosen was to create a tool which used flashcards as a method to help students study. This app would primarily be aimed at any student from the ages of 10 to 24, since most people in this age bracket have access to a mobile device. It was important for the design to be clean and minimal, as well as easy to use. The app was developed using react native, which opened up the possibility of making the app available on iOS in the future. The core features of the application were defined as allowing the user to create, update or delete a deck of cards, as well as be able to use the decks for study.

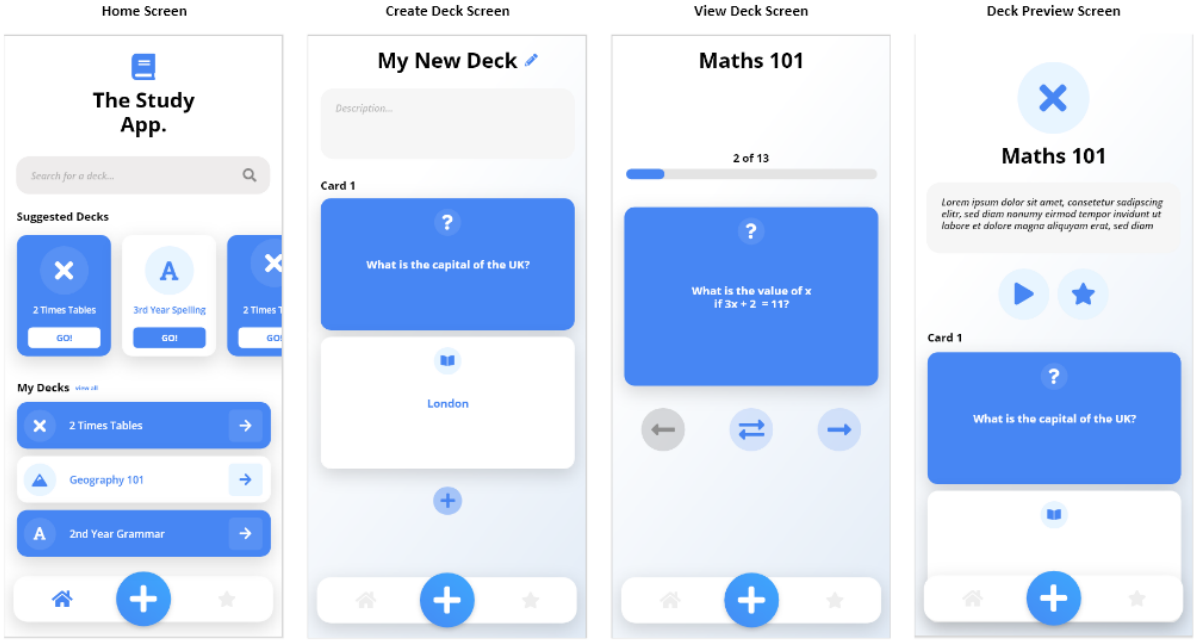
Inspiration

The primary piece of inspiration for the application was Quizlet. While Quizlet are not only limited to flash cards, it is certainly a feature that has been well fleshed out. One thing found when using Quizlet's flashcard feature, was the complex UI, which could be an issue for younger children who might not be able to grasp it quickly. It was important that when designing this application to have a much more simplistic user interface.

When designing the UI, the website Dribbble was used to gather inspiration for each of the elements.

Design

For the design of the app, it was decided a light theme would be used throughout, with the accent colour of light blue. There weren't too many screens to design, since the nature of the app was clean and uncomplicated.



As the app was developed using react native, several design decisions had to be made before the development phase began.

One example of this was how the components were going to be styled. One option would be to have all the styling rules in the same file as the components themselves. This solution proved to be less favourable since it meant overcrowding each file. So, it was decided to keep all the styling separate, this also helped to promote the reuse of style classes.

Another example of an important design decision was which kind of components to write when developing the application. One way to do it is to use class components. This method is effective if you have a lot of big components with lots of functionality. However, since the plan for this app was to keep each component modular and not do too much in one place, it was decided the entire application would be developed using functional components and React Hooks, a relatively new feature with React.

The folder structure was done in such a way as to keep the higher level components as only containing other high level components, meaning there was little styling required in these components, as they were just a place to put all the smaller components together, to create the layout.

One other objective of the structure was to keep any database interaction code independent from the view layer code. This was achieved with the use of ‘handlers’ which abstracted all of the query code away from the view layer.

For the database, it was decided the app would use a local sqlite3 database. In order to achieve this, the node module ‘react-native-sqlite-storage’ was used to provide the glue between react native and sqlite. The database design for the application was kept small, with only 3 tables, as shown below. Since the application didn’t have any ‘backend’ so to speak, the database was seeded with fake data to simulate what it would look like if a user could see decks from other people.

It was clear from the design, icons were going to be an important part of the application. In order to simplify the use of icons in the application, the node module ‘react-native-vector-icons’ was used. This opened up the application to allow icons from the popular library Font Awesome.

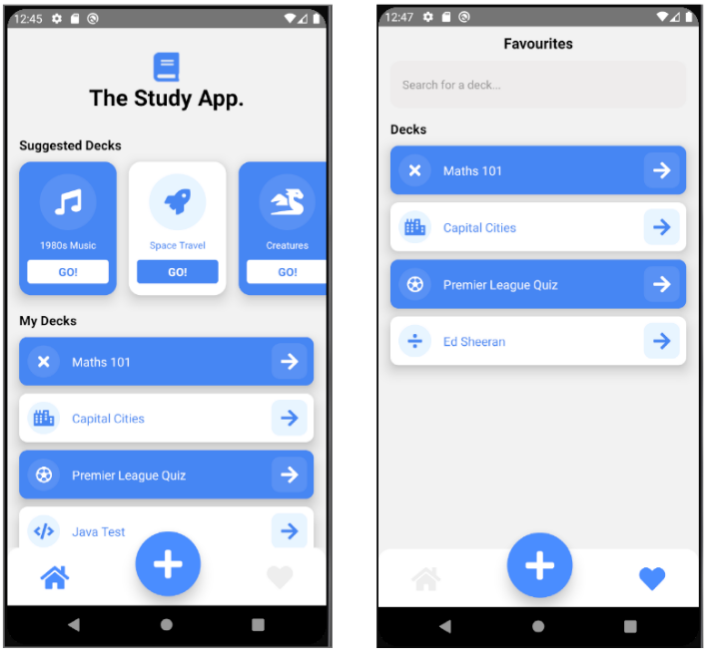
For navigating between screens, instead of just dynamically removing and showing the screen components, it was decided the ‘react-navigation’ node module would be used to simplify this task. This meant screens could be navigated, from and to, with full history support, as well as being able to pass route parameters to each screen.

Final Implementation

The following screens are where the user can browse for decks of cards. The ‘Suggested Decks’ section on the home screen is where users can see decks that weren’t created by them.

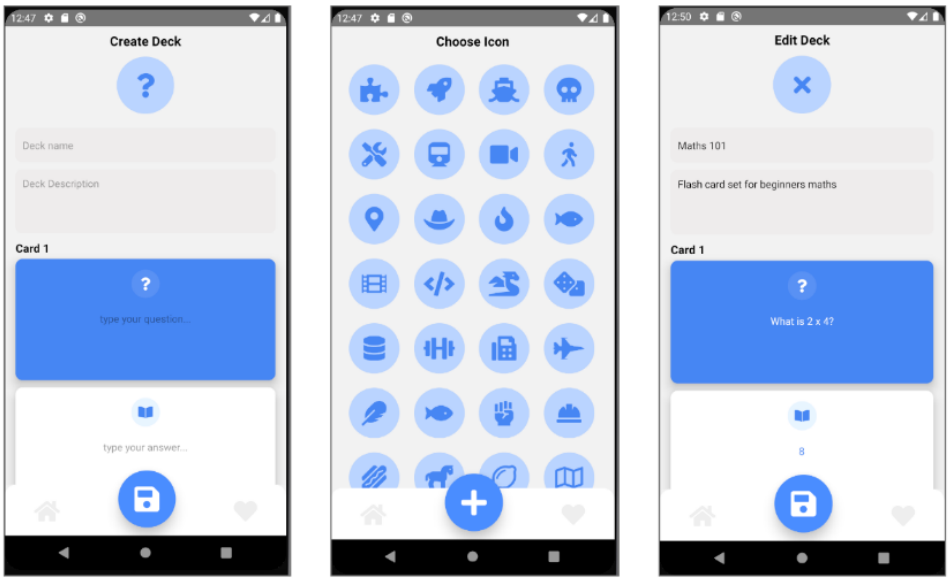
Since the user can favourite decks, it made sense for there to be a place where they could go to see all their favourite decks. The search bar allows you to search through the favourited decks.

Browsing Pages



For the deck creation screens, it was important that the UI was simple and easy to navigate, so that the user could quickly create and edit flash card decks. To change the icon for the deck, the user could just click on the icon at the top and select from a long list of icons (as shown in the middle screen). At this stage, the user can also create as many cards for their deck as they wish, using the plus button underneath each card. The middle navigation button changes from being the usual plus to take you to create a card, to being a save button.

Deck Creation Pages

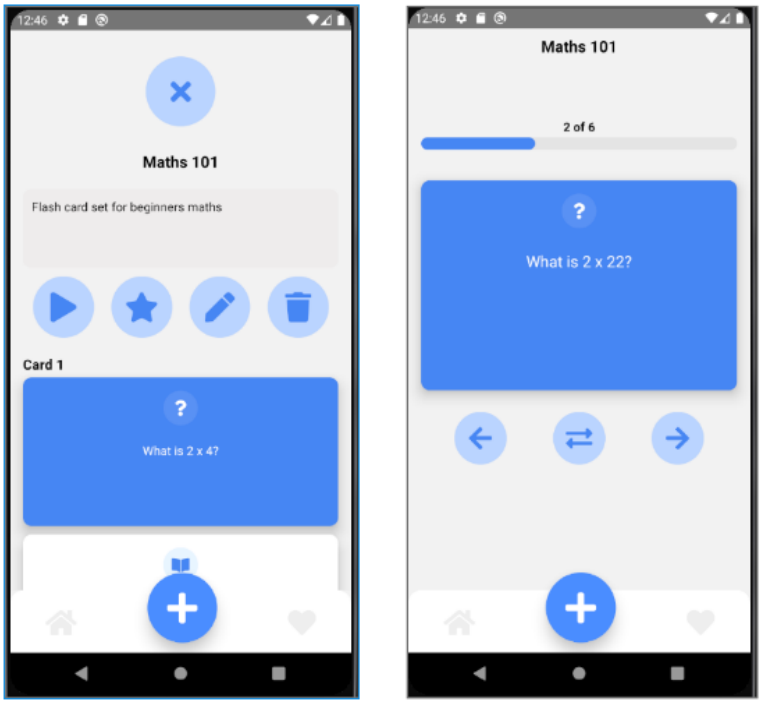


For the screens where the user interacted with the decks, it was important to have any action buttons large and available to the user straight away, thus granting a better user experience.

This was achieved on the deck summary screen by having a row of buttons underneath the description of the deck. The ‘edit’ and ‘delete’ buttons only show if the deck was created by the user, otherwise only the ‘favourite’ and ’play’ buttons are present.

With regards to the deck viewer screen, it made sense to have the card take up the most of the screen, since that is what the user is most interested in. Below it, there are three buttons, two for navigating between the cards in the deck, and another for flipping the card to reveal the question or the answer.

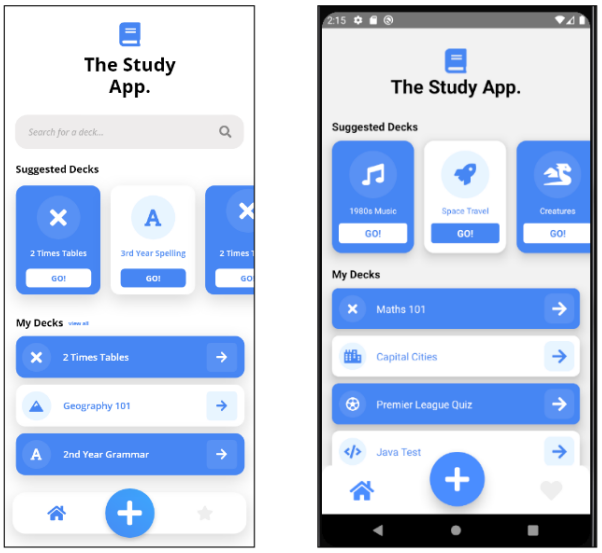
Deck Interaction Screens



Evaluation

The final product ended up being very close to the final design. However there were a few things which came up in the development process that hadn't been accounted for when in the design phase.

Design compared with final prototype



For instance, it was decided the search bar should be dropped from the home screen, since it wasn't obvious how the UI should react when a user searched for an item. Another example is in the design of the bottom navigation bar. It was planned to have an offset from the edge, however, it was decided this unnecessarily complicated the UI and was also removed.

There were also a few instances where a certain aspect of the UI had not been adequately thought out or designed, and therefore needed to go back to the design stage in order to properly find a solution. One example of this is in the original design, there was no place for the user to choose an icon for their deck of cards.

Deck viewer screen design compared with final prototype

However, there were several cases where the final implementation closely resembled the final product. In the above example, excluding the navigation bar, there are very few changes from the final design.

Improvements

Overall the application has achieved the core goals that were set out during the design stage. However there is plenty room for improvement and there are several features which could still be implemented in order to grant the user a better experience.

A major feature which could drastically improve the application would be to create a full backend with the concept of users. This would mean the users would be able to use decks other people created, as well as share ones they created with others.

One other improvement that could be made is to use images instead of icons. This would give the app another level of customizability, which would lead to an improved user experience. Another place images could be used in the application would be in the flash cards themselves.

One thing that potentially could be improved within the codebase is the styling format. While the structure is there to promote reusability, it was not necessarily optimized in this example.

Another feature that could be implemented to improve the user experience would be to add the concept of age or difficulty levels. This would mean the user would be able to filter for decks at their skill level, instead of currently seeing materials for all age groups.

Node Modules

- react-native-vector-icons
  https://www.npmjs.com/package/react-native-vector-icons
- react-native-flip-card
  https://www.npmjs.com/package/react-native-flip-card
- react-native-sqlite-storage
  https://github.com/andpor/react-native-sqlite-storage
- react-navigation/native
  https://reactnavigation.org/

Websites Referenced

- Dribbble
  https://dribbble.com/
- Quizlet
  https://quizlet.com/en-gb