Cornell Notebook

Kyle R. Olson

As someone who struggles taking notes, I wanted to create an app that will help students in my same position. My solution is a note taking app that utilizes the note format created by Cornell University. The Cornell Notes format places emphasis on note-taker interaction by prioritizing key components of notes such as the essential questions, the notes themselves, as well as questions the notetaker may have while writing.

Initially, the user is greeted by a login screen, and presents them with options to sign in with a Google Account, Apple ID, or email and password. This is so that, as notes are stored in the cloud, the notes are tied to their account, allowing for security of notes and a piece of mind knowing their data can be restored if the device fails.

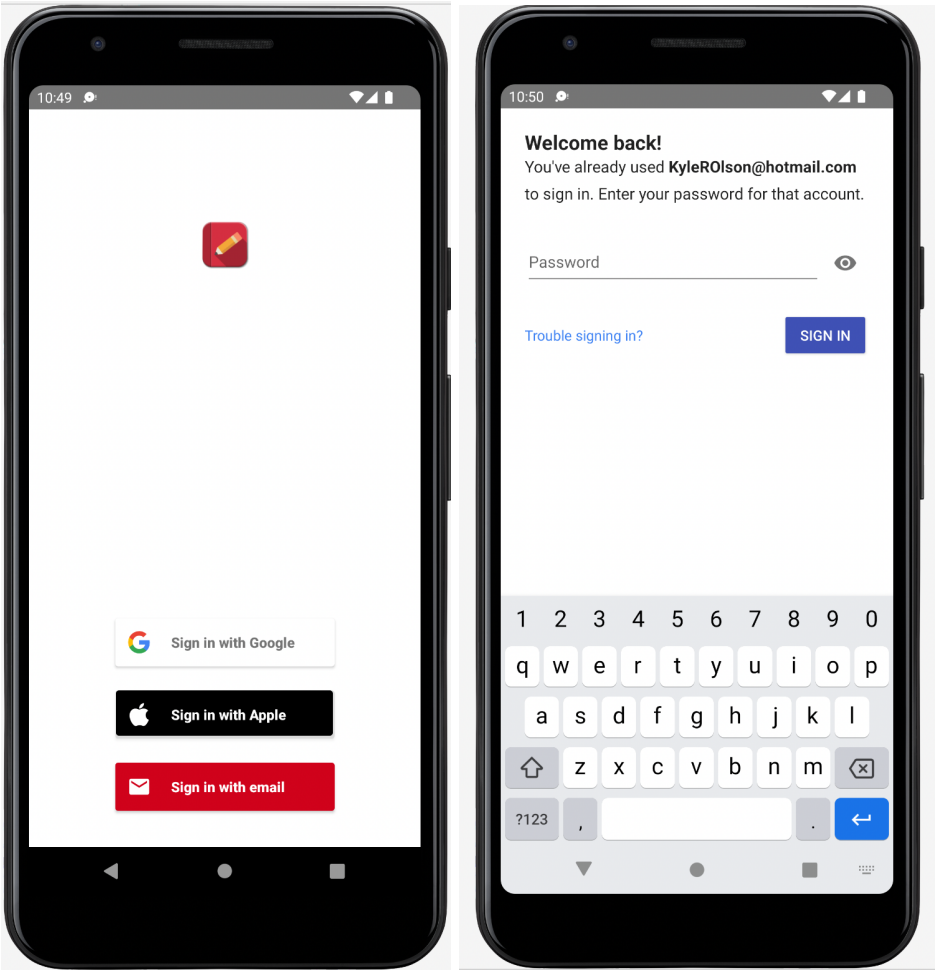
Upon successful sign in, the user is redirected to a home screen. If a user has already signed in before, they will automatically be redirected. At the top of this screen is a search bar to search all notes. Underneath the search bar is a scrollable container of the notebooks that will act as folders for the notes. The notebooks can be customized by giving them a title and background image. Lastly, pinned to the bottom will be a list of recently opened notes. This will function as a reminder for users to circle back and review notes they took.

If a user taps a notebook, they will be presented with a different screen, which features a heading so they know they are in the correct folder, and a list view of notes in the notebook container. The search bar will change so that it'll search for notes that are just in that container.

The last screen will be the note itself. There will be an input with fields for the user to enter their topic, essential question, and notes. At the bottom of the screen is a toolbar, so that the user might be able to create headings, change the style of text, etc. This bar is automatically pushed up to right above the keyboard when the user starts typing.

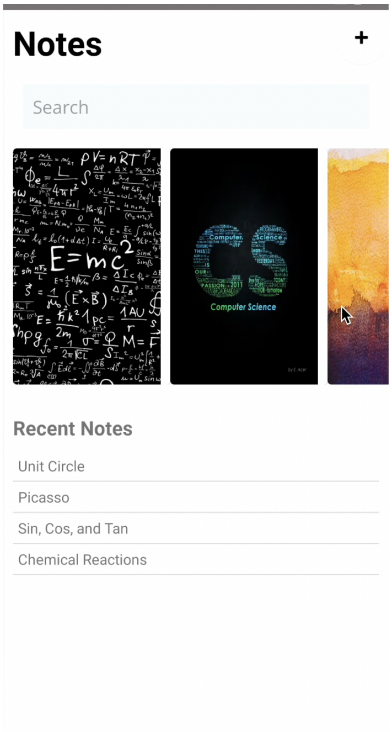
1. Home Screen-
    a. Clean in design, yet functional
    b. Create Notebook (folder) button
    c. Search Bar
    d. Heading
    e. Cards layout for different folders
    f. Customizable background for each card
    g. List of recently viewed notes
2. Folder View-
    a. Contains each note within a folder
    b. Displayed as list
    c. Similar header to homescreen
    d. Name of folder at top
    e. Search bar
    f. Create note button
3. Note View-
    a. Will show the topic at top
    b. Essential Question
    c. Notes Section
    d. Can use formatting tools to interact with notes
    e. Notes are stored using Firebase Realtime Database with their API

Login Screen-



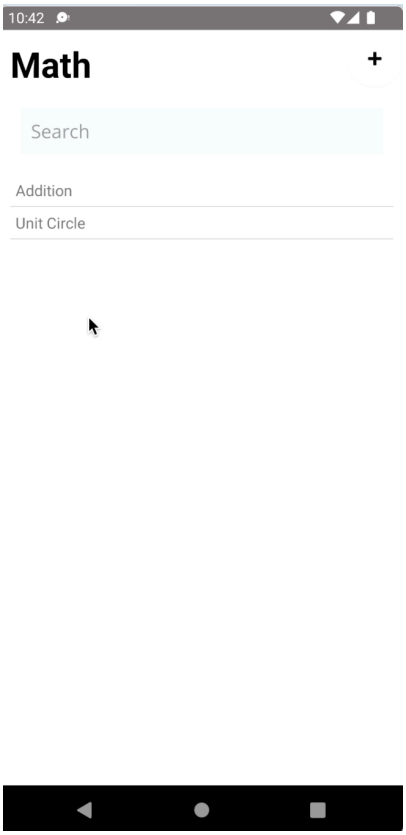
Initially the viewer is presented with the login screen. This is important as it assigns the user a UID, which is used in the reference call to Firebase when setting the directory. If the user doesn't have an account already, it creates one and assigns a unique UID. This is all handled server side with Firebase Authentication.

Home Screen-



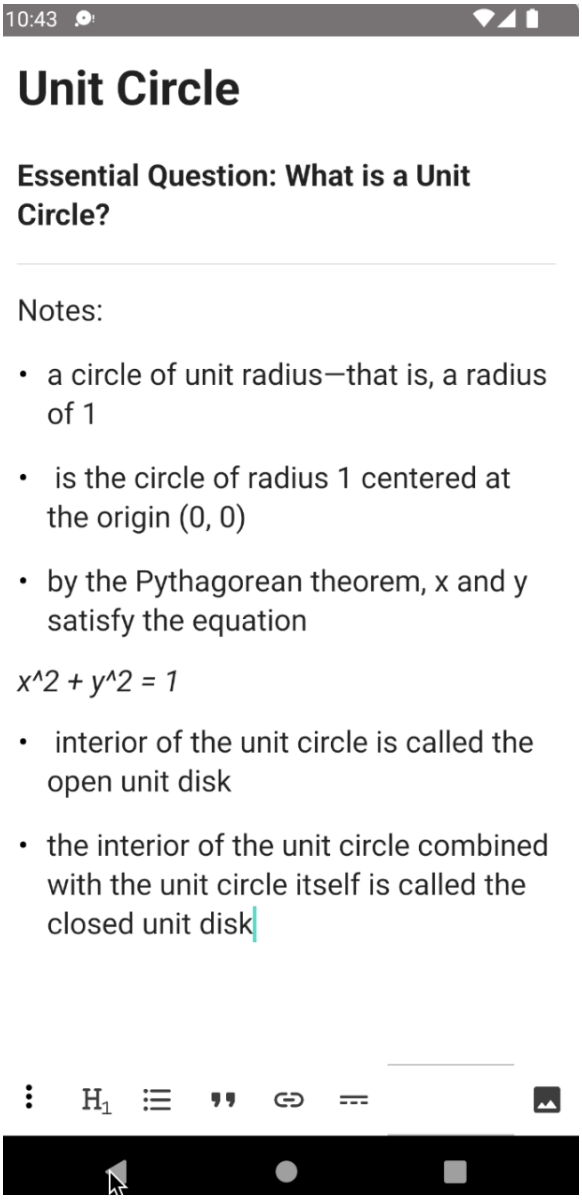
Upon successful sign in, the user is redirected to the homescreen, with their UID passed as well. The notebooks are generated in a for loop. It creates a Material Card View for each child in the user directory and gets it’s background image, stored as a url string in Firebase. Neither the search or recent notes functionality were implemented due to time constraints.

List View-



When a user clicks on a notebook, the user is redirected to the List View Activity with the path passed as a parameter. This is to set the new directory so only notes in that folder are shown. A call to the Firebase API is made so each of the titles can be stored in an array, which is passed to an Array Adapter to draw the list.

Notes View-



Lastly and most importantly is the note view. It uses the MarkDEditor library (https://github.com/bxute/MarkDEditor) to create the formatting for the note, and adds the rich text editor at the bottom. The content can be retrieved as JSON and stored as a string in Firebase. It is then loaded in by parsing the JSON back.