Android Programming - CS 386P

University of Texas at Austin

Final Project Writeup

Spring 2025

Team Members: Kevin Salazar

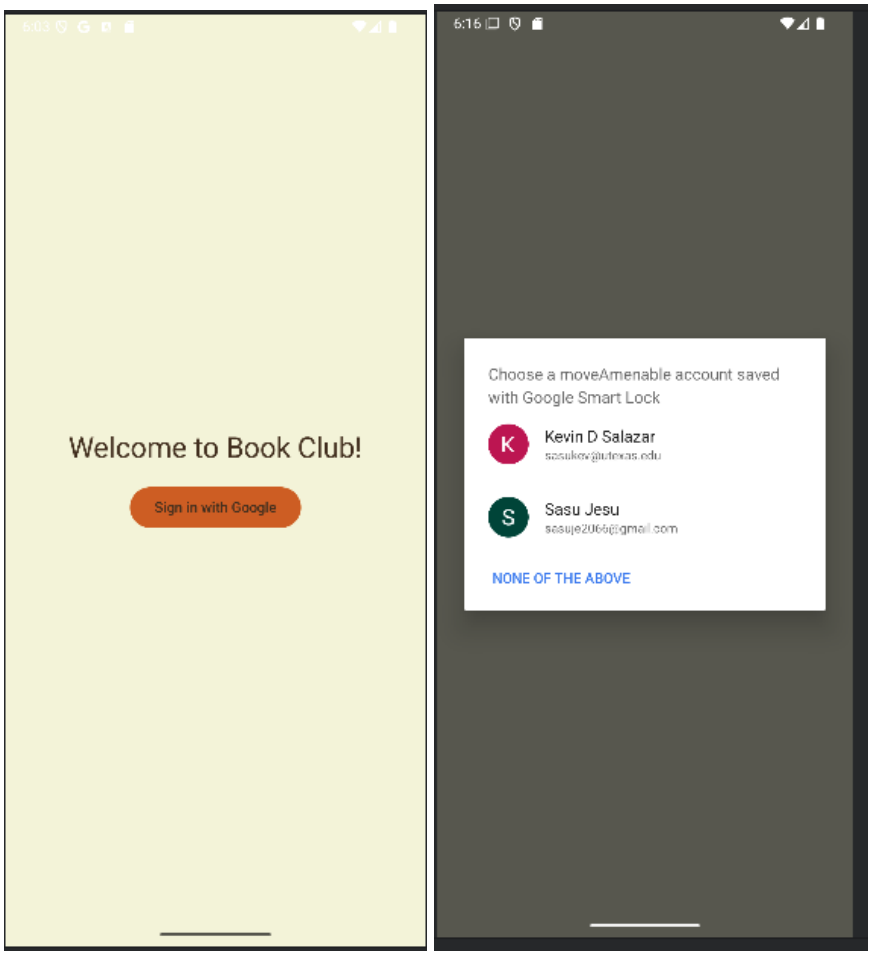
App Title:
BookClub
[formerly moveAmenable]

App Description:
Interactive Application dedicated to Books, where users can search books, add them to a personal reading list, and be present with a reading community by using comment threads on each book.

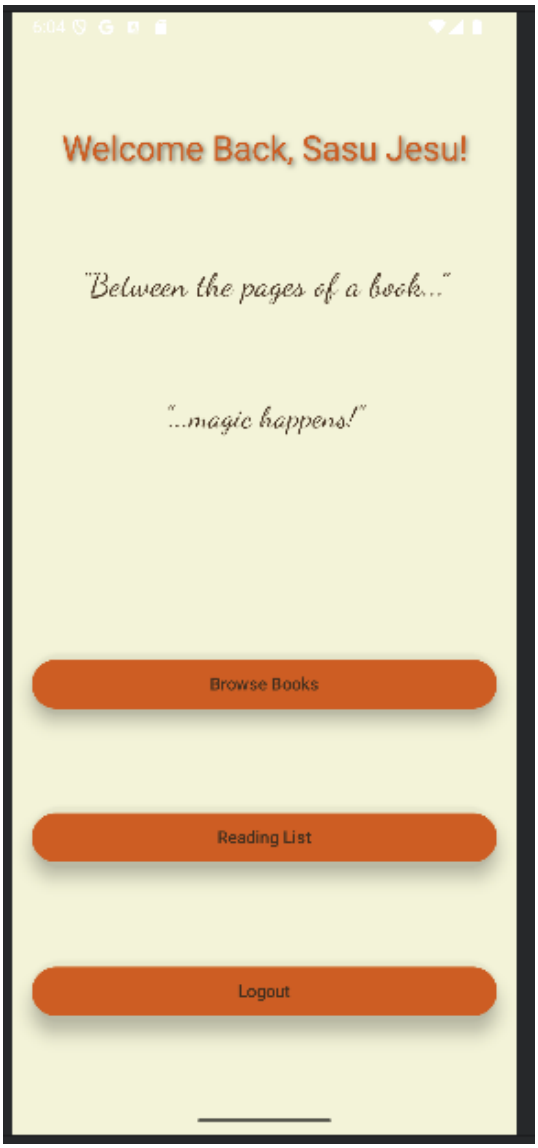
App functionalities:
- Log in with Google account (FireAuth)
- Search for books utilizing the Google Books API.
- Add books to their personal reading lists.
- Delete books to their personal reading lists.
- Database functionality (FIrebase) to manage personal reading lists.
- Ability to see details of a book.
- Engage in shared comment threads with other users within the app.

Screenshots of App:

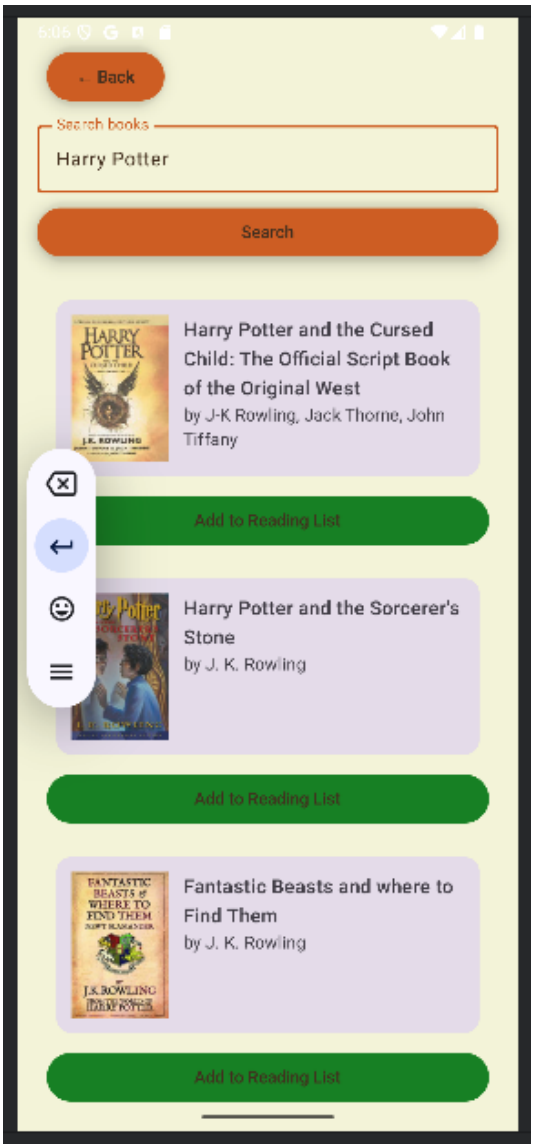
Signing in:



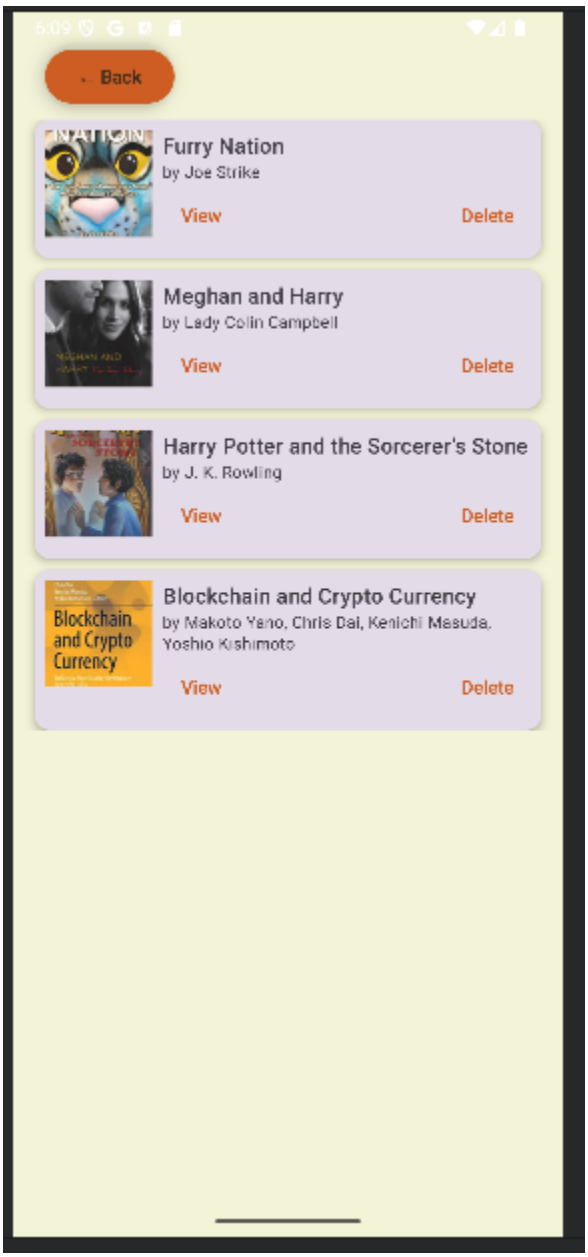
MainScreen:



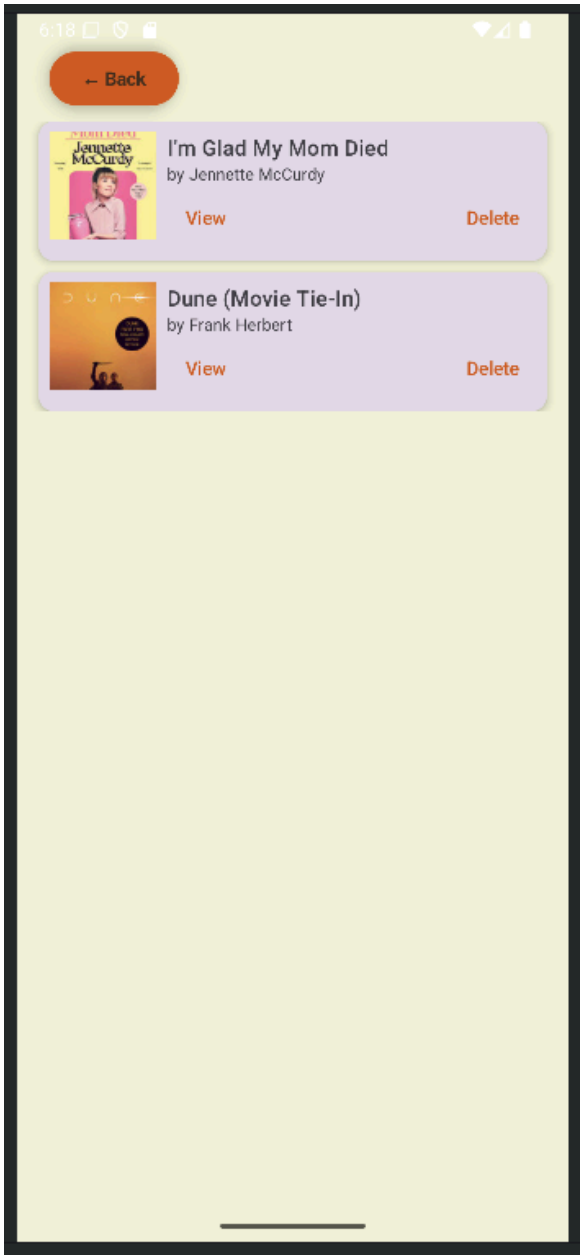
SearchScreen:



ReadingListScreen (Unique per user):

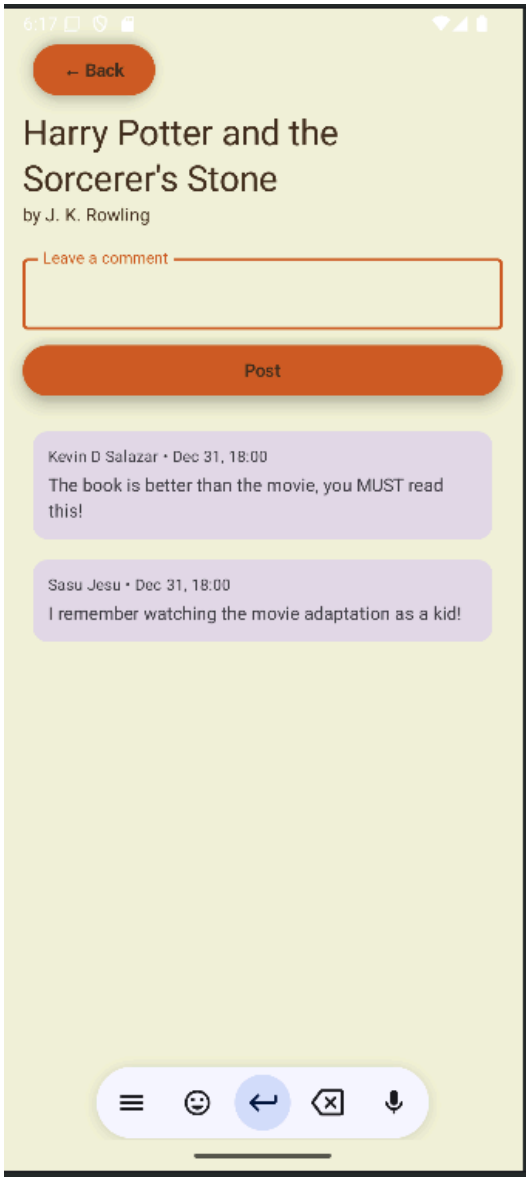


Reading list of user ‘Sasu Jesu’.



Reading list of user ‘Kevin D Salazar’.

BookDetailScreen (author name and reviews from different users):



APIs used:
-Google Books API

Android Features:

- Jetpack Compose was used, along with navigation for a smoother and simpler smooth screen navigation.
- Compose theming by using color palettes, button UI customizations such as shadows.
- Firestore's designated Android feature for real time updates (snapshotListener).

Third Party Services and Libraries:

- FirebaseAuthentication: Used to give the app Google sign-in capabilities.

Pros/Benefits: It was a quick implementation with compose, somehow felt faster than the implementation done in class.

Cons/Challenges: The logout would somehow not prompt the suggestion of using a different google account when signing into the app, it would just automatically sign in with the initial logged in credentials.
From the images attached, it can be seen that the app “moveAmenable” enabled smart locking for those credentials on screen, some configurations through the firebase console had to be changed.

- FirebaseFirestore:Real-Time database.

Pros/Benefits: Used in the comments and reading lists, easy to display information while maintaining uniformity with the database.

Cons/Challenges: Having to invest in security rules for personalized data (reading lists) and shared data (comment threads).
In order to deal with deserialization issues between database data models you must go back to the console to fix any mismatches (it occurred to me with the ‘authorsfield’).

- Coil: Handles Image loading from remote URLs (in this case, google books thumbnails).

Pros/Benefits: Works with Compose, not complicated to integrate.

*Cons/Challenges:*Making sure of proper caching and error handling, at some point the thumbnails failed to load in the reading list screen.

- Material3 Compose:

Pros/Benefits: Used in the UI, it was nice to compare with the UI that we dealt inside the class.

Cons/Challenges: Having to invest in security rules for personalized data (reading lists) and shared data (comment threads).
In order to deal with deserialization issues between database data models you must go back to the console to fix any mismatches (it occurred to me with the ‘authorsfield’).

Ai Generated components:
AI Agent used was Github Copilot
-To help with gradle debugging, we used Groovy DSL during the course projects and now we get a modern version of the DSL, it was confusing at first, made me want to give up, and asked

the copilot about the differences between the DSLs. This made me learn about them quickly and continue with the project development.

-To design the UI structure, give examples of color palettes to choose from that could be used in the application theme. Ended up choosing the warm color palette that was recommended to me.

-Asked to generate logs used for debugging, during the authors field mismatches when implementing the firebase service.

-Creation of FancyBackButton, with the goal to have a less basic UI, I wanted to use something different than the usual arrow symbol button that takes you back to the previous screen, so I asked to generate some examples and I chose the one I preferred the most.

Noteworthiness Descriptions:

- UI/UX/display code

Attempted to have a warm/cozy app with the usage of the Welcome back message and the subtitles with reading phrases, I believe this could set the mood of reading and the “magic” that brings to book lovers solely through words and meaning.
Used book cards to have each book's information presented in a neat manner.
Implementation of personalized buttons for specific actions and navigation.

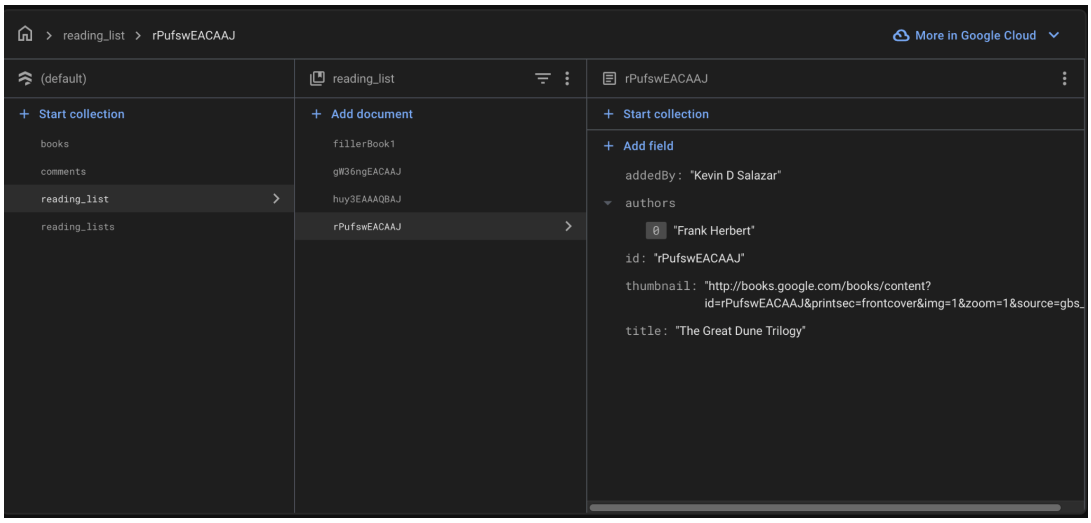
- Backend and processing logic

The creation of view models for BookSearch, ReadingList, and Comment, some of these connected to the firebase state (comment repository) and the API calls (book search repository).

The Firebase data structure includes:

reading_list/{userId}/books/{bookId} For all the personal reading lists.
comments/{bookId}/entries/{commentId}: For the comment posting under a book.

Firebase Screenshot:



Most important thing I learned in the project and difficult challenges:

Resilience is key for completion, this is mentioned because I had multiple hiccups with the initial gradle setup, I created a project with the basic Empty Activity template, and started building onto it. Made me learn more about gradle, the syncing process and the configuration processes done in them, all of this took time, but its all for the best.

I also struggled with the API set up, my initial goal was to develop a fitness based app… but after 3 attempts of setting up APIs… I was willing to deliver a project that only used mock data! But I realized that taking breaks at project hiccups, writing down the details for future reference (once there is less stress due to stalling) and creating temporary workarounds such as mock data can speed up the development process.

When the mock data was implemented, I was able to see how the firebase mock data was manipulated with Compose from the app. Had i just stalled at the API part, I possibly would never have continued the implementation (only until the API set up was done).

With this being said, I initially wanted to call the app MoveAmenable and make it fitness based, but the API issues made me migrate and redesign to a book based app BookClub app with a working API.

How to run project:
In order to run the project, the configuration files will be included in the commit so no need to worry about firebase. The Google Books API may have to be set up independently, you need to go to the Google Cloud Console and get the API key. A demo video will be included.

Report of lines of code using Cloc :

github.com/AlDanial/cloc v 1.96 T=0.85 s (309.4 files/s, 94128.2 lines/s)

Language	files	blank	comment	code
JSON	160	0	0	33776
XML	30	4325	32	15366
Text	11	0	0	14871
Java	15	1563	4479	3626
Kotlin	29	114	57	925
Bourne Shell	1	23	36	126
Gradle	3	23	11	109
DOS Batch	1	21	2	66
Properties	8	0	33	49
TOML	1	3	0	36
SQL	1	0	0	2
Markdown	1	0	0	1
ProGuard	1	3	18	0
SUM:	262	6075	4668	68953

Was not able to get the Code Frequency graph from github, I am not part of Github Team.

Link to demo video:

https://drive.google.com/file/d/1jEqfZ1q99gTXlkjNo0m-uCrsxbX5l99w/view?usp=sharing

BookClub App Demo Video.mov

Link to repo:
https://github.com/utap-s25/moveAmenable-sasukev09