SENG 3210– Applied Software Engineering

YourCodex Project

Jose Contreras (T00714272)

Brock Young (T00708314)

March 3, 2025.

List of Tables

List of Figures

1. Introduction

The proposed design of a Book Recommender System aims to bridge the gap between avid readers and undiscovered books. As more and more technology surrounds us with each passing year, a new demographic yearns to return to more traditional media to escape the overwhelming digitalization of everything. However, many readers do not know where to start. YourCodex aims to connect the world of traditional media with the modern digital era to strike a happy medium between the old and the new. Book readers will be able to use the platform in order to find undiscovered gems, and books that would otherwise remain unknown to them.

Books have been around for centuries, remaining mostly the same for most of their history. They simply provide compiled sequences of text for readers to absorb through their pages. However with the advent of technology, an opportunity has arisen to add a modern twist to how we approach books. Through the YourCodex service, readers will be able to rate previously read books and be recommended new ones based on a smart matchmaking algorithm. This modernization technique of matching readers with books best suited to their interests aims to attract new readers who rely on technology or may not know what books to start reading.

Existing matchmaking algorithms have proven to have large success. Applications such as TikTok, Instagram, Youtube, and other media platforms thrive largely due to their algorithm recommending users content tailored to their interests. Much of this content is often viewed under a negative light, reducing attention spans, and causing addiction to technology. YourCodex will aim to employ a matchmaking algorithm in a positive manner, connecting users to long-form media to increase attention span and knowledge.

The following sections of the report will specify the important design considerations that went into creating YourCodex. It will outline the problem definition and how it is addressed, the functional and non-functional requirements, constraints and objectives. Solutions will be proposed, discussed, and ranked according to how they satisfy the requirements. A final solution will be chosen and be discussed in detail. The scope and outcomes of the development project will be mentioned, as well as the management aspects and roles of each team member. A final conclusion will cement the key takeaways from the developed product and report.

2. Design Problem

2.1. Problem Definition

In the current society, users intend to utilize their devices to achieve some degree of entertainment. Nevertheless, users are easily overwhelmed by the intense amount of entertainment material available on the internet, producing user's exhaustion and keeping them away from what they actually would enjoy in their leisure time. Recently, it is somewhat common to receive feedback from users that involves the following idea: *"There are too many things to watch but I don't find anything that I like"*, where users are solely blinded by the amount of information/material offered by the entertainment industry. The previous stated situation is the main problem that some entertainment services are currently facing.

However, there are some key solutions or counter-measurements taken by experts in order to retain user's attention, the main one being the application of Recommendation Systems. This type of system takes as an input the consumed materials of a user and finds common points through those in order to find new materials that present those commonalities which the user might like. Therefore, it is a filter based-on the taste of the user and it is unique to every user.

The current project is the creation of a recommendation system as the main idea and delivery, intended for a ebooks application. The scope for this project is to create a prototype of the App, "YourCodex", that functions as a recommendation system and shows the main information of the recommended books such as Title, year of publication and other, without having the ability to actually read the content of those books, focusing solely in the "Recommendation" feature of the App.

2.2. Design Requirements

The initial requirements can be categorized as follows:

1. Functional Requirements:

The functional requirements for the developed system showcase what key features the finished product should be able to display. They are listed as below

Application Name: YourCodex

- After some deliberation the application was named 'YourCodex'
- The 'Your' portion of the name adds emphasis on the product serving the user and personalizing recommendations
- The 'Codex' portion of the name references the theme of the recommender being based around books, but in a more fun and intriguing way than just 'library'
- 'YourCodex' ultimately serves as a memorable name which relates to the product being designed and the personal aspect of the recommendations

Remote Access & Book Recommendation

- Users of the system will be able to download the application and access the service anytime from any location
- **Alternatively, users will be able to access the service through a web portal via Firebase
- Users will be able to search books available on the service and mark them as read, along with their rating and review
- The service will compile all user ratings for books and formulate an average rating to display to users
- Books will be rated to users based on genre, popularity, rating, and users previously read and rated books

Searching Option:

- Books will be stored with an associated title, author, tags specifying genre, rating, etc
- Users will be able to search for books based on the title and author
- Users will be able to filter search results based on criteria such as minimum rating

Administrator Controls:

- Administrators will be granted elevated privileges to the book recommender service to improve quality of the product
- Administrators will have options to:
- Add books: Create new entries for newly published or available books for users to read and rate
- Edit books: Update book information such as description, title, author, etc.
- Delete books: Remove books from the service to adhere to local policies or regulations regarding banned books or for other reasons

Real-time Dashboard:

- The system will have a home menu upon launching the app which features a real-time dashboard
- The dashboard will dynamically update to showcase the top ten recommended books based on the internal system algorithm
- The user will be able to interact with the shown book entries

Account Creation:

- The system will facilitate the creation of user accounts and storage of account information in the system
- Users will be able to login to their accounts to view their saved data and personalized book recommendations

View Book Ratings & Recommendations

- Users will be able to view other users reviews left on books
- Users will be able to view other user profiles and the books they recommend

2. Non-Functional Requirements:

The non-functional requirements outline how well the system performs its defined functions. The defined system non-functional requirements are as listed below:

Performance:

- Users of the application must not observe any noticeable latency or input delay
- Launch of the application should take under 5 seconds on 80% of all supported devices
- Navigating between application screens should take under 200 ms

Security & Data Integrity:

- User login information should not be stored on product system in unsecure way (i.e. without encryption or hashing)
- System should not ask for or store user sensitive personal information in database

Modifiability:

- The system should be built with modular components to facilitate easy implementation of new features
- The system should remain as simple as possible without unnecessary convolution

Compatibility:

- The developed product must be available to a wide range of devices (from at least API level 19 (KitKat))
- The system UI should support as many different resolutions as possible (i.e. standard phone size, tablet size, etc)

Usability & Accessibility:

- The system should be intuitive and user-friendly, allowing maximum amount of demographics to use the service
- The design should use pleasing colours and sufficiently large text which do not impair colour blind or hard of sight individuals

Scalability:

- The system should be programmed in such a way to easily enable future features
- The system should be able to support large numbers of books and recommendations without significant delays or impacted performance

Energy Efficiency:

- The system should be designed in such a way to not consume large amounts of energy or resources and minimize environmental impact

3. Solution

3.1 Solution 1

The first solution was constructed as abstractly and generally as possible, defining the design methodology and process rather than concrete implementation. It was thought that through compilation of previous lab work, and some modifications and additions, a working solution might be created. This would implement previously covered concepts such:

- Version control through Github (L1)
- Login window (L3)
- Pop up messages through Toast (L3)
- Displayed book information through list views (L3)
- Test plans (L5)
- Unit tests with Espresso (L6)
- Multiple screens through intent (L7)
- Back buttons (L7)
- Data management through Firebase (L8)
- Search for data in database (L9)

In addition to the above features, a variety of wishlisted features to create were brainstormed, such as:

- Viewing user profiles
- Administrative dashboard
- Web portal interface
- Sophisticated multifactor matchmaking algorithm incorporating genre, popularity, rating, and users past read and rated books
- Filtered searching using custom criteria such as minimum rating
- Mark books as read and rated
- View other readers ratings on books

The implementation of the book data was to be manually added by the developers through a proprietary book class, and storage on the database. This would allow for custom books and the possibility to add any book which exists without the limitation of if third party services have it.

3.2 Solution 2

The second solution aimed to address the shortcomings of the first solution, including scope and feasibility of development, and concrete methods of implementation. Additionally, it was determined that it was unwise to compile previous lab code into a single module, rather than take the previously elicited knowledge and recreate the application from scratch for better organization.

The first solution proposed several wishlisted ideas that ended up being out of scope. It was ultimately decided to prioritize core functionality, and at least temporarily eliminated extraneous features to streamline the development process and make deadlines. The prioritized features included:

- Version control development process through git and Github

- Login and signup windows for user account creation and sign in
- Application status updates through Toast popups
- Display of book objects through formatted thumbnail, title and author fields
- Test plans and unit tests
- Multiple screens and activities through intent
- Search functionality for books
- Fetching of books through Google Books API
- Storage of critical information through Firebase
- Administrative window to add, delete, and manage user accounts and books
- Ratings of books
- Recommendation of books via algorithm

Instead of creating each book entry and data from scratch manually, this solution aimed to make use of the existing Google Books API in order to import the large existing dataset of books. This would enable a hugely significant amount of supported books, improving the overall quality of the application.

The second solution was less abstract then the first, outlining more softwares that would be employed, such as:

- Android Studio
- Git & Github
- Firebase
- Toast
- Espresso
- Volley
- Picasso

Most of the core logic for this solution would go into just the user class. Key information for each user would be stored in Firebase, such as the user ID, password, favoured genres, past read books, and more. The information regarding the books would be able to be fetched directly through Google Books API, to save database storage and ensure efficient use of resources.

3.3 Final Solution

The final solution further refined the second solution into a polished final product, with a more refined design process and key objectives to accomplish. It incorporated the strengths from both Solution 1, and Solution 2, while minimizing the shortcomings. Most of the key elements from Solution 2 were carried over, but many small details were changed to better accomplish previously set out objectives and functions.

Solution 3 was designed in such a way that pairs strong functionality with an aesthetic user interface. The application was made to be intuitive and useful, with key features that would be expected from a book recommender software. The design achieves the key criteria in such a way to elevate the user experience to the highest level. It employs several different activities for separate functionality, making use of design principles such as strong cohesion, and minimal coupling. Examples of the different activities include:

- Admin Dashboard
- Login screen

- Sign up screen
- User home page
- Search screen
- Book preview

The employed software and libraries were similar to Solution 2, with some minor tweaks. Volley and Picasso were exchanged for Glide and Retrofit. Additionally, along with Retrofit, a Gson converter was added to convert fetched json files into java objects. These three softwares allow for the fetching of data from the Google Books API, displaying images from a URL, and converting the fetched data (in json files) into java objects. The full list of software and libraries is listed below:

- Android Studio
- Git & Github
- Firebase
- Toast
- Espresso
- Glide
- Retrofit
- Gson

The development process of Solution 3 heavily revolved around a git and Github workflow. The two developers collaborated over a single shared repository, and followed a conventional version control process. Unique features would originate from separate branches, and be merged into other branches depending on completion status, interoperability, priority, and other factors. Commits were made frequently, labelled according to the changes, and branches merged upon function completion. In the rare case of merge conflicts, resolution would occur through Android Studios graphical user interface.

The Google Books API served as a core feature of our system. Using the API allowed us to instantaneously fetch countless amounts of books semi-instantly, saving time and improving scalability and quality of the application. Limiting the application to only be able to manually add books would severely incapacitate the functionality of the product, and was not a feasible option. The API provided the following useful field for each book:

- Title of book
- Description of book
- Author of book
- Subject or genre of book
- Thumbnail of book

The following class diagram was implemented in order to fully represent the final solutions with a detail structure of the its system:

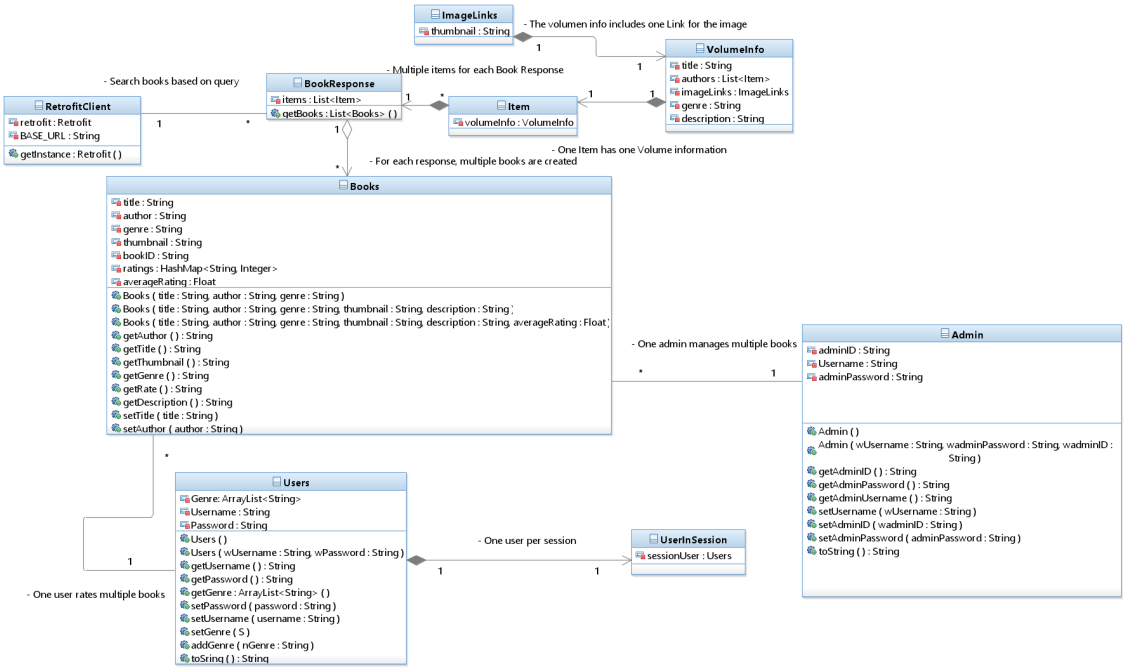

Figure 1: Class diagram for Final Solution

3.1.1 Features and the software architecture

The final solution aimed to satisfy the key functionalities expected from a Book Recommender system in an elegant and polished way. The key features are outlined as below:

- Version control development process through git and Github
    - Enables efficient collaboration between several developers
    - Provides backup of project in case of computer failure
    - Encourages documentation and good coding practices
    - Provides timeline of development process
- Login and sign up windows for user account creation and sign in
    - Users are able to login to existing accounts to view their information
    - Users are able to create new accounts to interact with the software
- Application status updates through Toast pop ups
    - Users are able to see popup messages in case of unexpected system behaviour
    - Users are able to see helpful messages to assist in system navigation or describe system actions
- Display of book objects through formatted thumbnail, title and author fields
    - Book objects were created in a visually appealing way
    - Book objects display key information such as title and author in efficient way
- Test plans and unit tests
    - Test plans were implemented to ensure system behaves as expected
    - Unit tests were performed to monitor system interaction
- Multiple screens and activities through intent
    - System functionalities were separated into separate screens and activities to facility strong cohesion and weak coupling
    - Intent was used to swap data and user focus between designed activities
- Search functionality for books

    - Users are able to search for any specific book by title that exists in Google Books API (over 40 million books)
- Fetching of books through Google Books API
    - Book information is fetched with specifications on fields such as title, author, description, and genre
- Storage of critical information through Firebase
    - Critical information is stored on the cloud through Firebase services
    - User data such as user id, password, ratings, read books are stored on cloud and not locally to facilitate access through various devices
    - Information is still available even in case of catastrophic local system failure
- Administrative window to add, delete, and manage user accounts and books
    - Administrative users with elevated privileges can manage objects inside system
    - Both users and books can be added, deleted, or managed through the dashboard
- Ratings of books
    - Books are able to be rated by each user
    - Ratings of each book are averaged and displayed to the users
    - Top 10 books are recommended to every user
- Recommendation of books via algorithm
    - Algorithms recommends books on unique user by user basis
    - Algorithm takes into account book rating, and genre to tailor recommendations

3.1.2 The system interfaces

Inside YourCodex app, there have been several considerations into the type of events that can be handled with ease and keeping a reliable front for the users. Mostly, this events are of the type signal, where there is a expected response from the system to the user interaction with it, involving the following:

1. User presses button: Throughout the entire app, there are different activities that are called or “displayed” when the user presses a specific button. For example, if a user wants to enter the homePage activity, they need to press the login button inside the MainActivity in order to be validated (their entered credentials) and go from one activity to another.
2. User presses a AlertDialog: AlertDialog components inside YourCodex interface were implemented to allow users to select from a range of “specified” values in order to set their preferences regarding books (normal user) or to edit the actual information regarding a book (Admin level), where all the desired options are saved after the user presses the button “Ok” inside the AlertDialog,
3. Users press a book: The App is able to make clickable images with the thumbnail of a book appending at the bottom the title and the author of each book. These clickable sets of elements are grouped into a horizontal carousel in order to display more books while keeping consistency with the actual available space inside the device. When the User presses one of these images, it will redirect it to a new Activity where details such as Title, author and synopsis and Thumbnail are displayed in a safe manner.

Nevertheless, the development team also considered the importance of the implementation of time-triggered events where all the effort is set into the reading and obtaining the data from the database, where time-triggered events are set to be done inside the Firebase Methods such as onDataChange and onCancelled. As well some time-triggered events can be found directly inside the android studio environment where the running of the app and moving through the multiple Intents behaved as desired inside the Performance Objective.

3.1.3 The user interface design

Much of the effort of the app development was put into the design of the user interface. It is important for the users to not struggle to look at our interaction with the application in order to maximize retention and enjoyment. The design of the application starts before even opening the software. YourCodex has a unique icon that displays on the app launcher, which helps promote branding and give the user perspective of what the application is about.

Upon launching the application, the user is prompted to login or sign up into their account. If the user already has an existing account, they are immediately free to enter their account details, and continue to the main application functionality. Otherwise, they are given the option to press the signup button and create their account.

Once on the main screen, the user is greeted by the simple but functional homescreen. The content is organized in such a way to maximize the options immediately available to the user, without overwhelming them. Part of this is the horizontal carousels of books available to the user. They are greeted with rows of just a few books each, but are able to scroll horizontally to expand their options. Each row of books is additionally labeled with the corresponding category, such as the top ten currently most popular books, or the books recommended just for that user. To navigate to a new menu, the user can either tap on a book, or press the search button.

Tapping on a book opens a new activity which displays key information regarding that article. Information displayed includes the thumbnail of the book, the title, author, and description of the book, the genre, and also the rating. The user also has the option to leave a rating, which affects the book's total average rating.

If the user instead taps on the search button, the application instead redirects them to the search menu. Here the user can search for any book which exists in the applications catalogue, and near instantaneously get all related results. Similarly to the home menu, the user then has the option to scroll through all displayed books, and tap on them to display more information or rate them.

3.1.4 The requirements traceability matrix

Table 1: Traceability Matrix

ID	Requirements Description	Business Need, Justification	Specification	Status	XML File
1	Select Genre	Users need a way to select their default interests	Finished	Passed	activity_create_account.xml
2	Search Books	Users need a way to search for new books they are interested in	Finished	Passed	activity_search_test.xml
3	Administrator	Administrator	Finished	Passed	activity_admin_

	Controls	s need elevated access edit and delete books			menu.xml
4	Real-time home dashboard	Users need real-time updates of the most popular books and related interests	Finished	Passed	activity_homee _page.xml
5	Account Creation	New users need a way to create accounts to access system	Finished	Passed	activity_create_ account.xml
6	View book details	Users need to be able to access more details about the book to determine if it interests them	Finished	Passed	activity_book_d etails.xml12

3.1.5 Environmental, Societal, Safety, and Economic Considerations

3.1.5.1 Environmental considerations

The software solution was designed to take into account the environmental factors and minimize contribution to the ongoing climate crisis. It accomplished this through several different avenues as outlined in the table below.

Table 2: Societal Key Considerations

Minimize paper usage of traditional books	The software designed at its core serves to digitalize the paper based medium of books. Traditionally, it is was not uncommon for book readers to buy a new book just to quickly find out they are not interested. The book would then likely just sit on a shelf or be passed around and be neglected. This would directly result in an inflated perceived demand, and wastage of paper, cutting down trees unnecessarily and making a negative environmental impact. By recommending users books, the system increases customer satisfaction rate and reduces unwanted books.
Minimize green house gases and emissions required to browse books	Traditionally in order to find a new potential book to read, readers would have to commute to the library or book store. If driving by car or other emission releasing transport devices, this indirectly leads to negative environmental impact. By allowing readers to browse through new books

	digitally from the comfort of their home, the system indirectly eliminates these emissions.
Minimize data stored in database by using third party service	The software minimizes the amount of storage used by not directly storing the whole collection of books on their own databases. Instead, the system makes use of Google Books API to fetch the books as needed, and only store the ones that are rated by local users. This greatly reduces the amount of resources needed for the application, marking greater efficiency and reduced wastage of energy.
Minimize code reuse through functions, modulation and good practices	The system was designed while taking into account good design practices to minimize wasted resources. Repeated code is grouped into functions, modulation is employed to facilitate easier debugging and code logical readability. Energy usage is reduced through efficiently written code.

3.1.5.2 Societal considerations

The system was designed to take into account the greater impact and outreach on societal factors. The application aims to achieve a net positive benefit on society through targeting specific areas - as elaborated below.

It is not hard to argue the benefits of technology, but the growing digital world is not without its downsides. The rise of social media has introduced attention grabbing social platforms such as tiktok, youtube, and instagram reels which through sophisticated algorithms can lower attention spans and cause an epidemic of phone addiction, especially among the youth. Instead of grabbing a book for entertainment, modern youth gravitate more and more towards their phone. YourCodex aims to show that the phone is not inherently detrimental to user cognition; it can be still used as a powerful tool and gateway towards other more positive mediums. The application aims to channel the addictive hold the smartphone has on users and redirect it into books through a similar benevolent matchmaking algorithm.

As technology continuously gets more entwined into daily life, the older fashioned medium of books grows less appealing to young audiences. By modernizing and digitalizing the first step of finding interesting books, YourCodex aims to increase the demographic of readers. Increasing the amount and level of literacy among the populus effectively increases the amount of knowledge available to everyone, bringing more balanced and informed perspectives and less misinformation.

As the amount of new media mediums grows, books can seem outdated, archaic, and overall less appealing. Free content and entertainment can be found on numerous different platforms, such as youtube, instagram, tiktok, and more. On top of this, the additional paywall to buy a book increases the effective gap between prospective readers and social media scrollers. People who may be interested in trying reading would be put off by the large price tag, and the potential to not even like the book they tried. By making book recommendations and information more accessible to everyone, and for free, YourCodex aims to increase the availability and accessibility of books.

3.1.5.3 Safety considerations

The safety of the system has been considered into the design in several ways, as detailed below.

The access to the system is divided based on the user's role; administrators have elevated privileges and can make direct modifications to the data and information stored by the system. They are able to add, delete, and modify objects. Normal users are unable to make such significant changes, and are just limited to indirect influence on the system, with information only regarding their user account and ratings being stored on the system, and no methods to add, or delete objects.

The system was designed in such a way to minimize potential user data loss; the majority of the information in the system is fetched through the Google Books API, and is not stored locally on the systems databases. Additionally, the storage of the user data is outsourced to third party tried and tested Google Firebase, which is likely to be more reliable than a proprietary design.

The system running status is constantly monitored through various conditional logical embedded in the program to ensure smooth and predictable performance. In case of unexpected behaviour, output messages depicting the situation or displayed via Toast popup messages to more easily navigate the situation and debug. Additionally, messages regarding unpredictable behaviour are output to Logcat to enable better addressing of the situation.

Test plans were designed alongside the development of the product to ensure a high quality application. Through constant testing of the application, bugs and unexpected behaviour can be filtered out early before they become significant problems. Additional unit testing through Espresso allows for further reliability and predictability of the system.

3.1.5.4 Economic considerations

The system was designed with a goal of maximizing functionality for minimal economic overhead. The final product was designed completely for free, and minimal external resources. Use of third party systems were used to enhance functionality, but only through the free tier to enable a high quality application for negligible cost. Additionally, the system is available for free for users, with no attached price tag or advertisements.

The system makes use of the free Google Books API to access millions of books and related data for no attached cost. This implementation saves the maintainers of the application from the tedious effort of having to manually add new books constantly, however that option is still available in case of a book not being accessible through the API. While the Google Books API does have a rate limit, which for this small scale application is negligible.

The whole application was developed almost entirely through the Android Studio software, and complemented by other software such as Git, Github, and Firebase. The Android Studio development suite provides everything needed to develop a fully fledged and professional application, such as a built in IDE, compiler, drag and drop component interface, virtual device emulator, built in compatibility with Firebase and Github, and more. Additionally, the software has countless free articles and documentation outlining usage, without incurring any financial cost.

The price tag of the system was decided to be completely free, reflecting the cost of development. The goal of the project was to increase the accessibility and availability of books, reducing barriers and with no attached strings. The cost of services used, such as Firebase, and Google Books API are free for a small scale, but in case of large increases in application usage, cheap paid tiers are also available to scale up. In the case of significant uptick in user demand, which exceeds the services rate limits, the paid tiers would be further explored, along with unobtrusive in-app advertisements to pay for these tiers without decreasing application functionality.

3.1.6 Limitations

While the developed solution achieved much of the objectives and goals originally set out, it was not without its limitations. Many wishlisted ideas were documented before development was started, and unfortunately the final solution was not able to implement all extraneous features. The project scope was severely limited due to unforeseen delays, and constrained time and deadlines. The hardware needed to develop the application was unavailable for much of the developers timetables, and the majority of the work towards the developing of the application had to be completed over the weekend, and after 6pm on weekdays. Additionally the developers were constrained by other projects and engagements, which further reduced time available. The limitations the applications suffers from are described below:

- Focus on core functionality
    - The system had to prioritize on development of core functionality due to limited resource of time
    - Wishlisted features such as reading lists had to be removed from the project scope
- Limited storage of data
    - Due to the systems reliance on Firebases free tier as a database, the application is only limited to 1GB of cloud storage
    - Further upscaling of the application would require a paid subscription to Firebase

4. Teamwork

Date: February 19, 2025. From 12:00 to 3:00 p.m.

Purpose: Brainstorm the general Layout of the Application. As well to practice the creation of a second activity inside android studio.

Meeting Minutes:

Table 3: First Meeting Minutes

Team Member	**Previous Task**	**Completion State**	**Next Task**
Brock Young	N/A	N/A	Practice with Android Studio.
Jose Contreras	N/A	N/A	Practice with Android Studio Tools.

Date: February 26, 2025. From 3:00 to 4:00

Purpose: Brainstorm the name of the App as well of the initial functional requirements and non-functional requirements.

Meeting Minutes:

Table 4: Second Meeting Minutes

Team Member	**Previous Task**	**Completion State**	**Next Task**
Brock Young	Practice with Android Studio.	75%	Creation of functional requirements
Jose Contreras	Practice with Android Studio Tools.	75%	Set-up Trello environment and create non-functional requirements

Date: March 1, 2025

Purpose: Gather the necessary documents for the First Partial Delivery.

Meeting Minutes:

Table 5: Third Meeting Minutes

Team Member	Previous Task	Completion State	Next Task
Brock Young	Creation of functional and non-functional requirements	100%	Design small prototype of the app interface
Jose Contreras	Set-up Trello environment and create app name and problem definition	100%	Design small prototype of the app interface

Date: March 2, 2025

Purpose: Work towards finishing Deliverable 2

Meeting Minutes:

Table 6: Fourth Meeting Minutes

Team Member	Previous Task	Completion State	Next Task
Brock Young	Design small prototype of the app interface	50%	Work on finishing section 1 of report
Jose Contreras	Design small prototype of the app interface	50%	Design small prototype of the app interface

Date: March 7, 2025

Purpose: Work towards Deliverable 3

Meeting Minutes:

Team Member	Previous Task	Completion State	Next Task

Brock Young	Work on finishing section 1 of report	100%	Work on creating main functionality of the app
Jose Contreras	Design small prototype of the app interface	100%	Work on creating main functionality of the app

Date: March 15, 2025

Purpose: Work towards finishing Deliverable 3

Meeting Minutes:

Table 7: Fifth Meeting Minutes

Team Member	**Previous Task**	**Completion State**	**Next Task**
Brock Young	Work on finishing section 1 of report	50%	Work on creating main functionality of the app
Jose Contreras	Design small prototype of the app interface	50%	Work on creating main functionality of the app

Date: March 17, 2025

Purpose: Develop the actual app of the project

Meeting Minutes:

Table 8: Sixth Meeting Minutes

Team Member	**Previous Task**	**Completion State**	**Next Task**
Brock Young	Work on creating main functionality of the app	40%	Finish Report
Jose Contreras	Work on creating main functionality of the app	40%	Finish Presentation

Date: March 22, 2025

Purpose: Complete the Report and Presentation

Meeting Minutes:

Table 9: Seventh Meeting Minutes

Team Member	**Previous Task**	**Completion State**	**Next Task**
Brock Young	Work on creating main functionality of the app	100%	Finish Report
Jose Contreras	Work on creating main functionality of the app	100%	Finish Presentation

Date: March 24, 2025

Purpose: Create Poster for the App

Meeting Minutes:

Table 10: Eighth Meeting Minutes

Team Member	**Previous Task**	**Completion State**	**Next Task**
Brock Young	Finish Report	Create poster	Create poster
Jose Contreras	Finish Presentation	Create Poster	Create Poster

5. Conclusion and Future Work

During the development of YourCodex, there were some trade-offs needed to be done in order to implement all the core features into the system to allow a reliable and on time final delivery. Among these, the main one is that the actual algorithm for the recommendation of the book was described as very basic and rudimentary, where there is room for improvement in order to make the app to be more versatile with the actual recommendations for users and not stay as “static” as it currently is.

Nevertheless, the main key features for the app were implemented as they were discussed throughout this document, some changes needed to be done in order to have the actual implementation of them into the system but they are currently working as expected and desired. It is important to consider that it was relatively “easy” to implement this features and have some very “small” maintenance for them because of practices such as Continuous Integration for software as well of taking an Agile approach since the beginning so that deliverables were not as “fixed” as in others frameworks. For each sprint, there were several tasks to be performed and they could be divided into the different screens that YourCodex app has in their implementation, where each one perform or better said, “allow” a certain behavior to be performed so that the user enjoys a fully-functional but “rudimentary” app.

Finally, YourCodex app, as a final product for this project, presented a few limitations based on functionalities; already discussed with the recommendation algorithm, and based on the User Interface; where some elements, such as buttons might be changed for future iterations in order to make them more to the liking of users, for example: The current App has buttons to go back to home page but it would be more appealable to users to make a clickable image of a go back arrow so it is more “relatable” to the current trends of the user.

6. Appendix

Video Demo: https://www.youtube.com/watch?v=r-IsVNwIEr0&feature=youtu.be

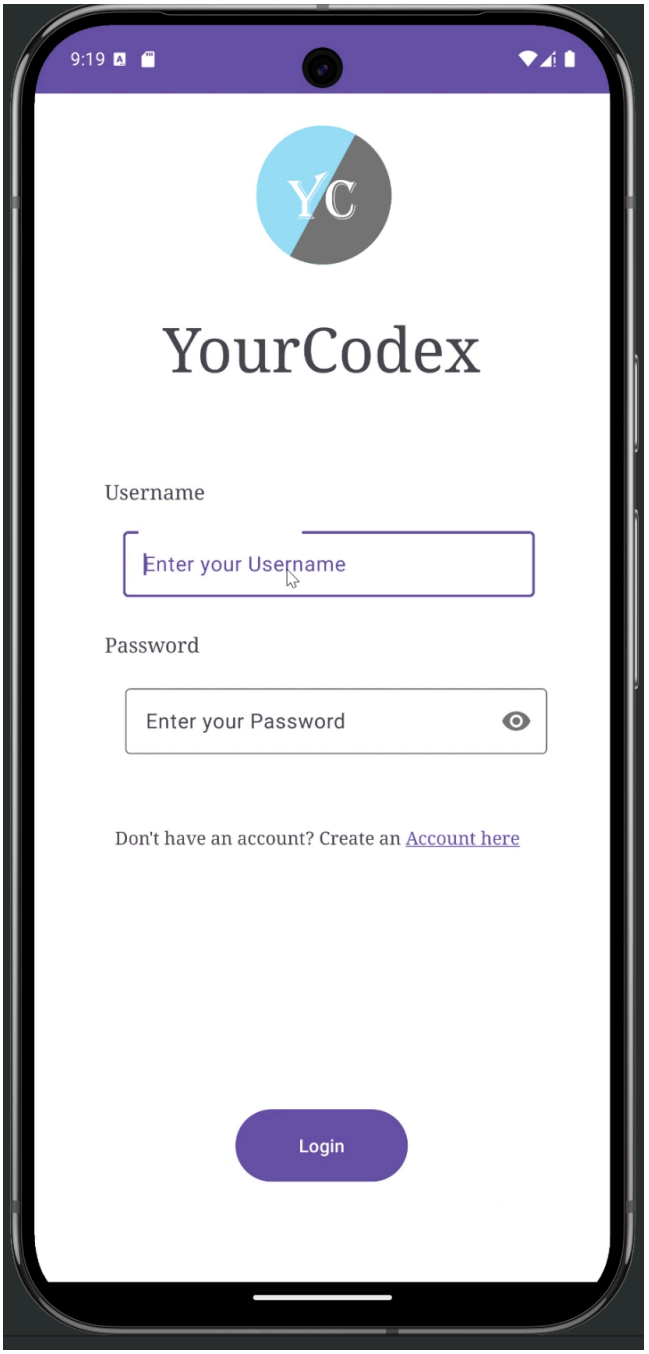

Figure 2: Main login page

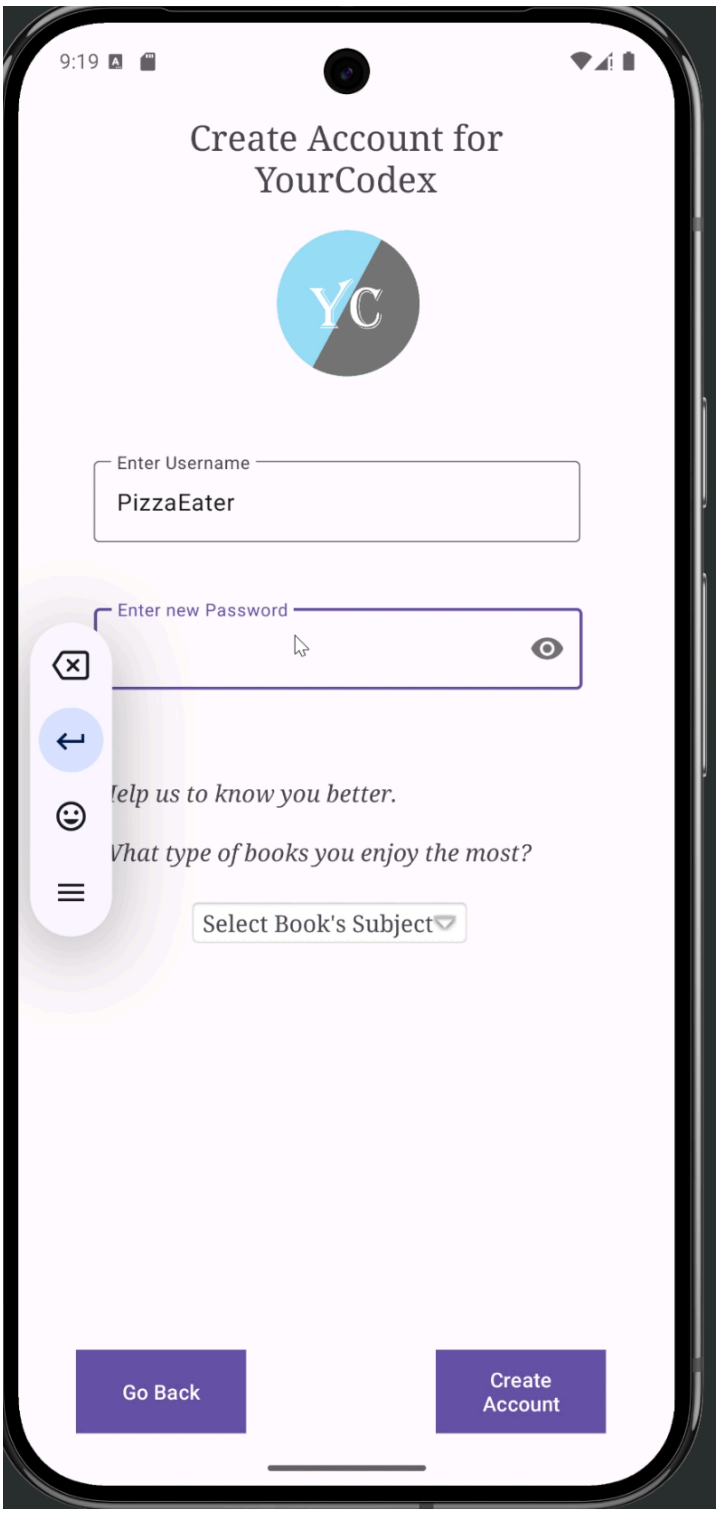

Figure 3: Sign up page

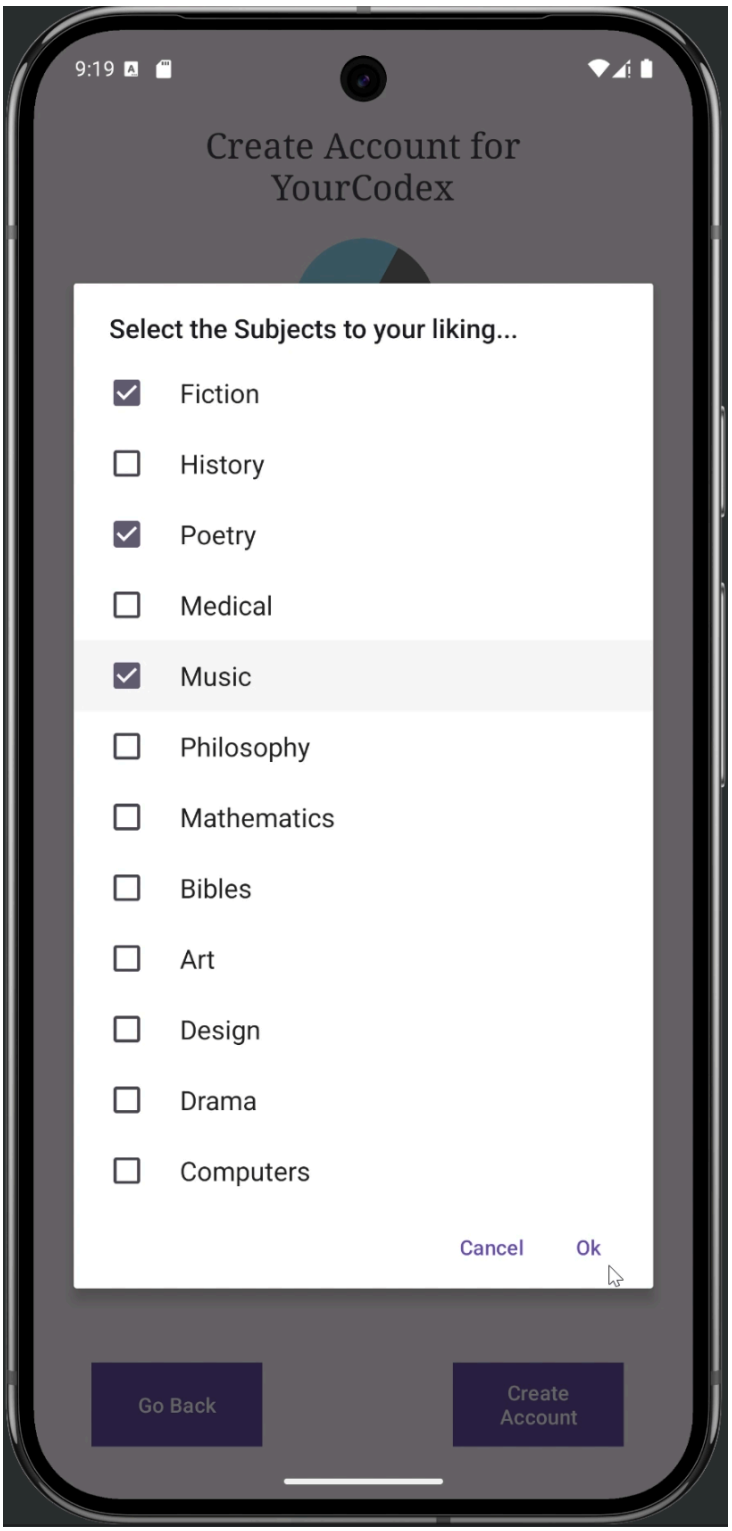

Figure 4: Select subjects and interests

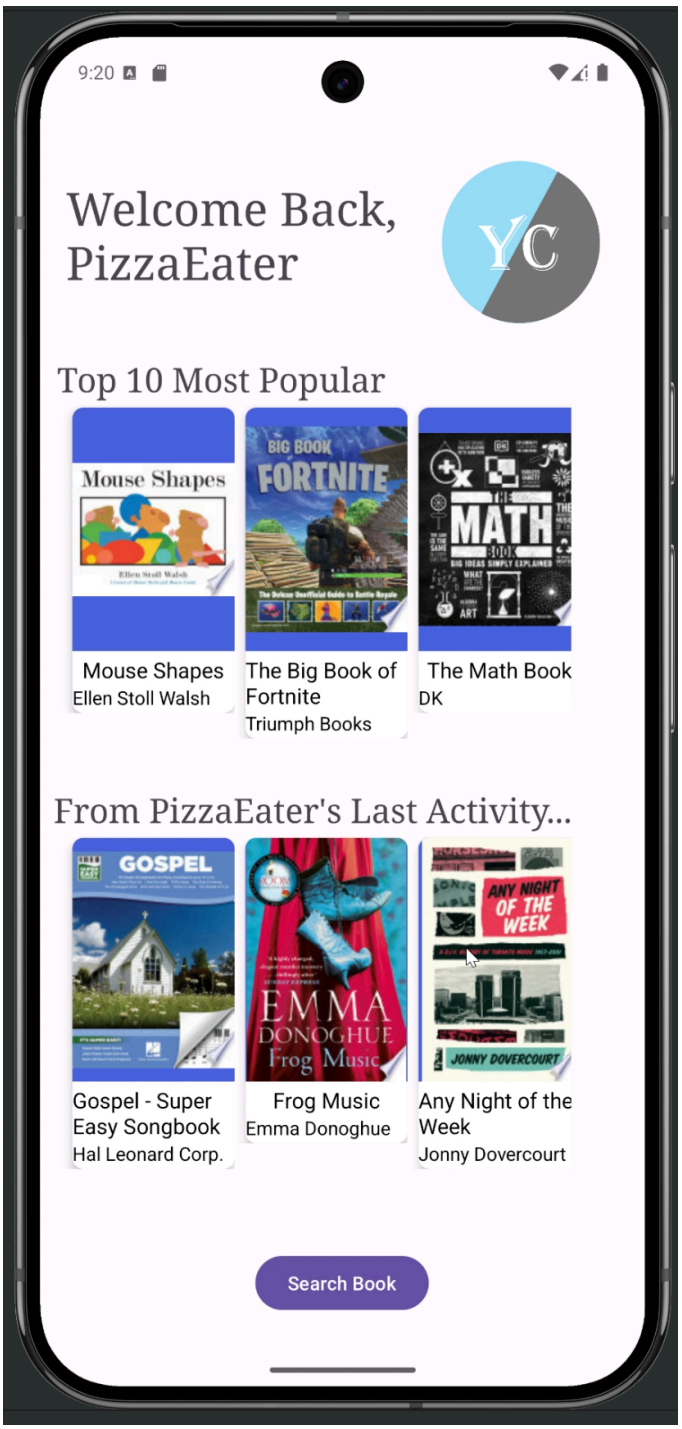

Figure 5: Home page which displays popular books and recommendations

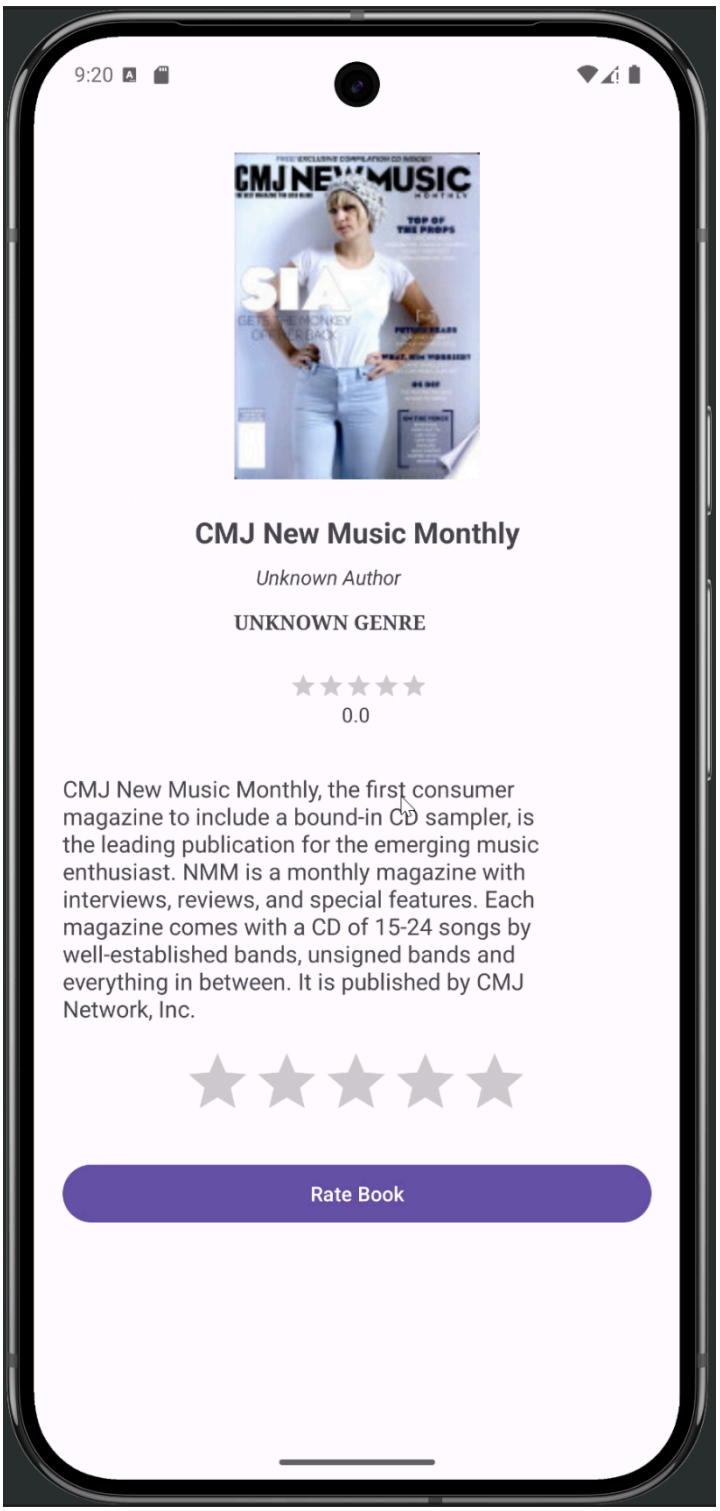

Figure 6: Display more information about book

Figure 7: Search books menu

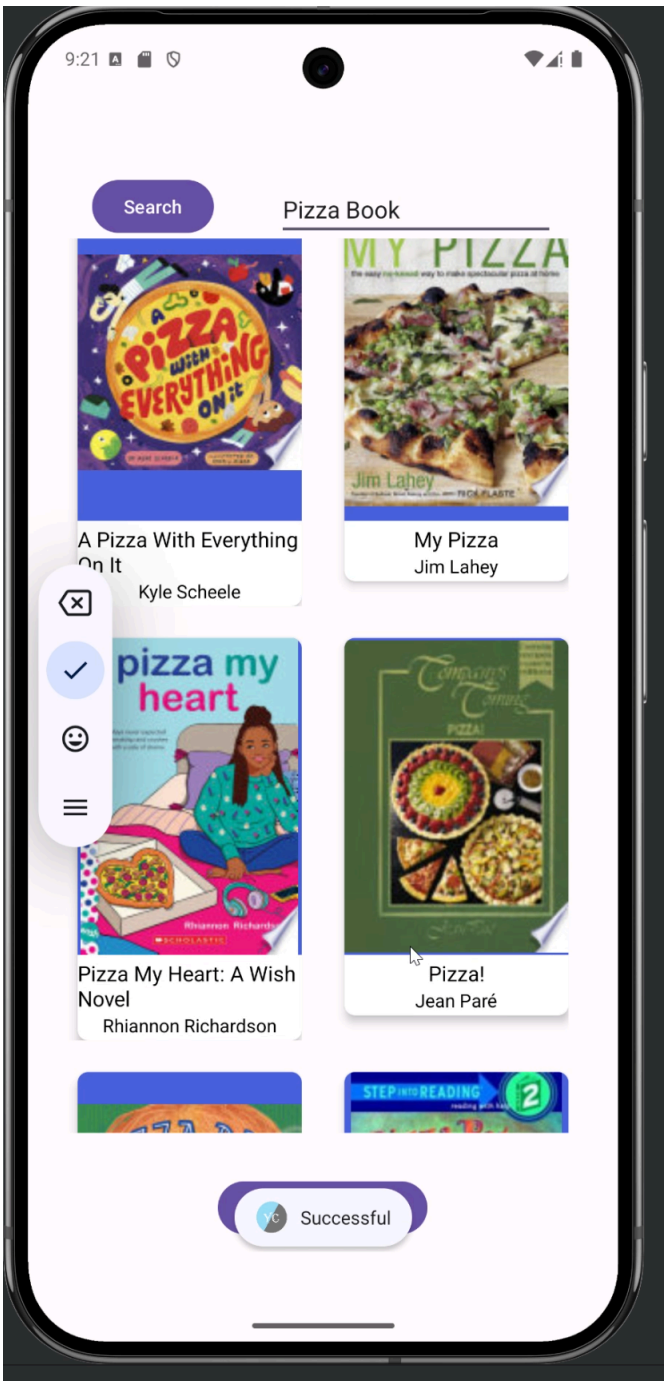

Figure 8: Display book search results

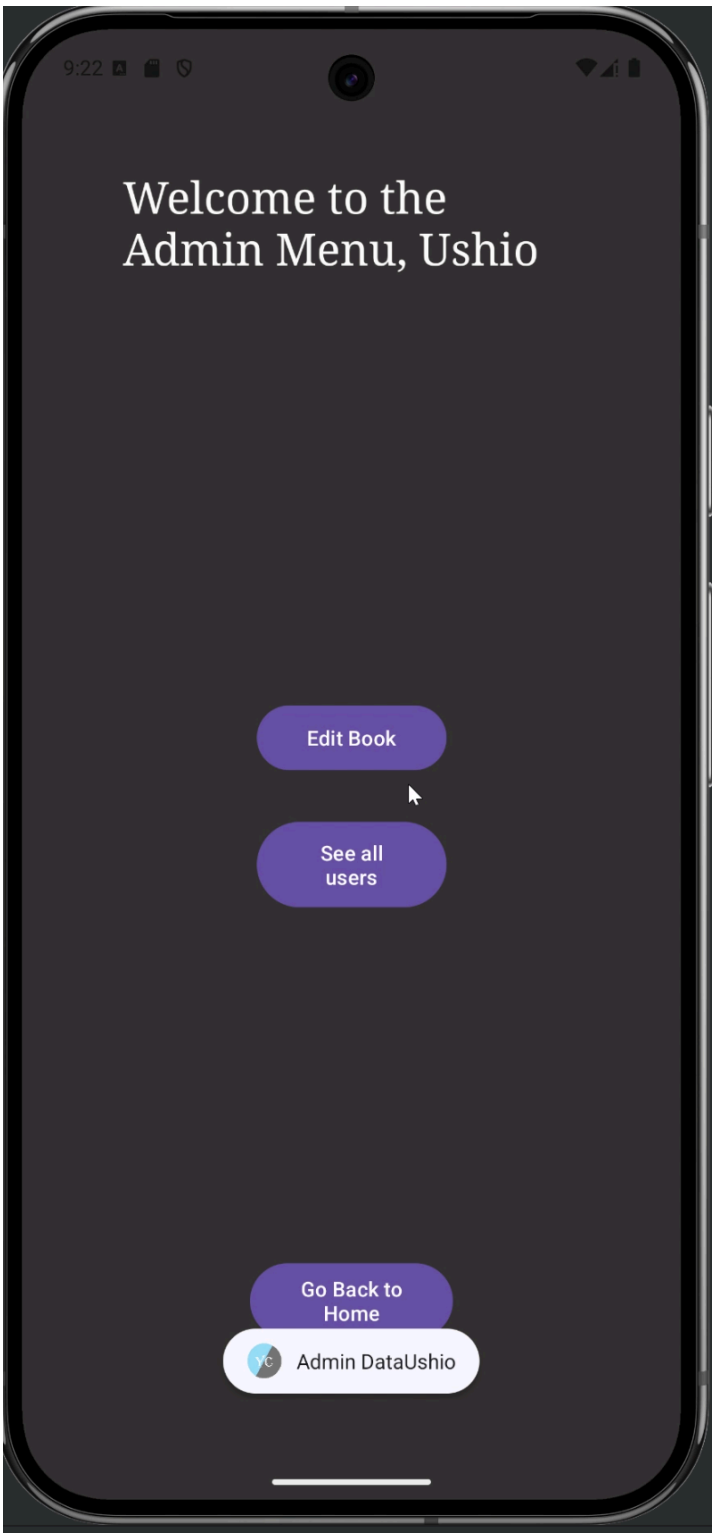

Figure 9: Administrator menu

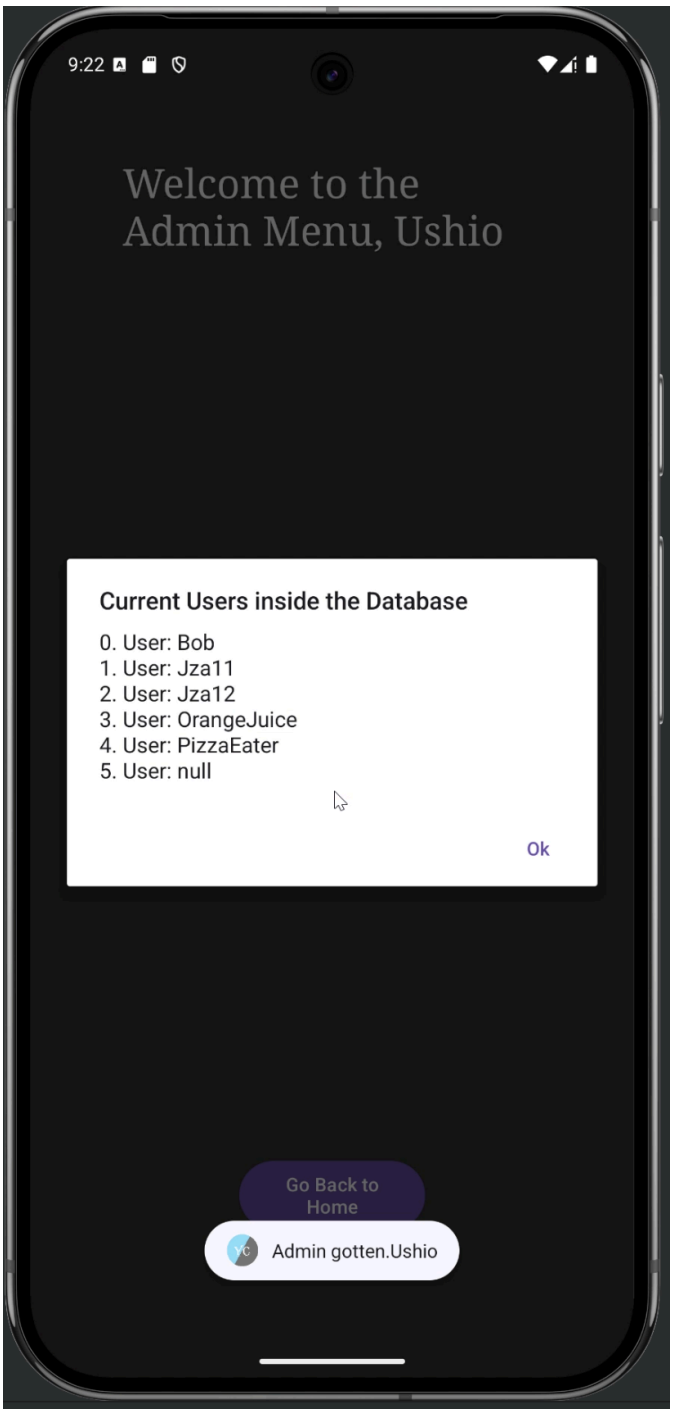

Figure 10: Display current users

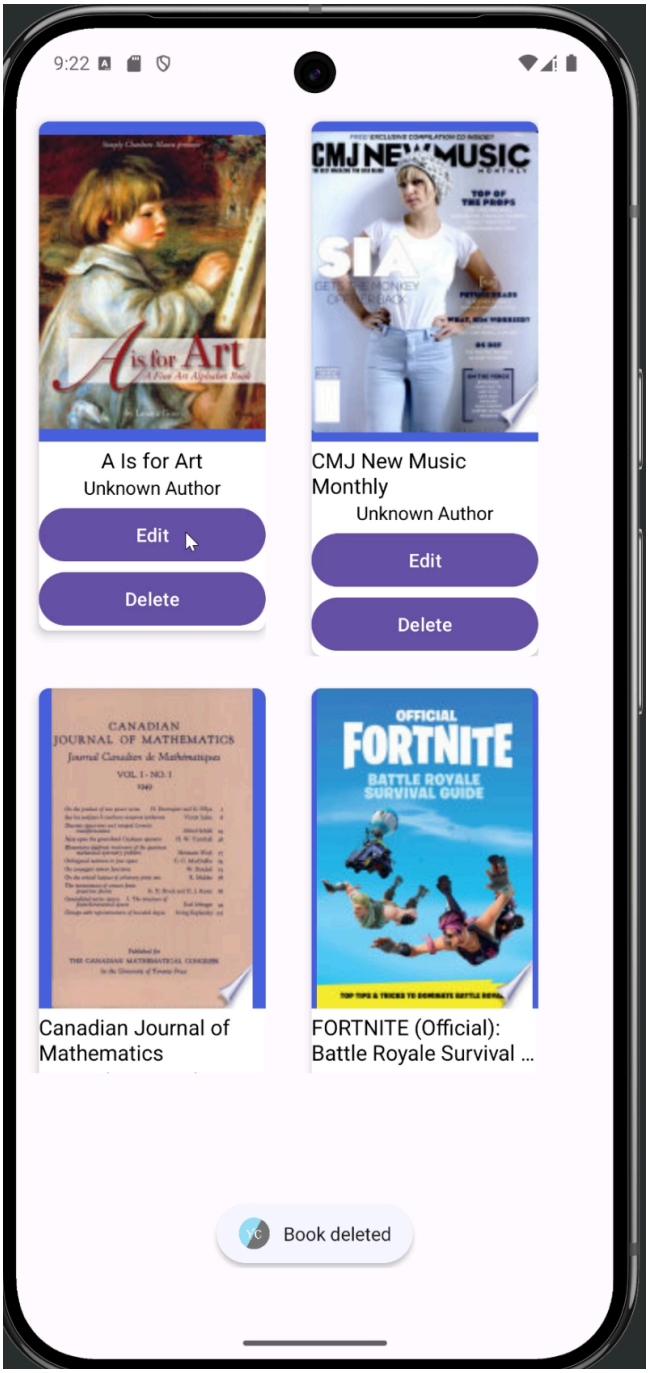

Figure 11: Edit or delete books

7. References

For this project, there were not cited references, nevertheless, in order to build the code for the project, documentation was needed in order to understand how to actually use the built in functions for the Database, Retrofit and Glide. Therefore, the documentation is as follows:

[1] Google for Developers, “Firebase Documentation”, *Firebase.* Accessed: March 10, 2025. [Online]. Available in: Firebase Documentation

[2] Retrofit, “Retrofit, A type-safe HTTP client for Android and Java”, Square. Accessed: March 12, 2025. [Online]. Available in: Retrofit

[3] BumpTech, “**Glide v4** Fast and efficient image loading for Android”, bumptech.github. Accessed: March 13, 2025. [Online]. Available in: Glide v4 : Getting Started

[4] Geek for Geeks, “How to Build a Book Library App using Google Books API in Android?”, Geek for Geeks. Accessed: March 13, 2025. [Online]. Available in : How to Build a Book Library App using Google Books API in Android? - GeeksforGeeks