CS 30700: Tsundoku Design Document



Team number 4: Sarah Luo, Kaley Kwan, Sam Liu, Eric Park

## 

[1. Purpose 3](#_350dzylk6xjn)

[2. Design Outline 4](#_h60tk04d8gp8)

[3. Design Issues 6](#_ql0ulxpb6ep6)

[Functional Issues 6](#_rwp2i697bqs5)

[Non-Functional Issues 7](#_757c4lxwledk)

[4. Design Details 9](#_5g6cgp5a00xk)

[Class Design 9](#_rv0rrwh2ke1a)

[Class Description and Interactions Between the Classes 10](#_54hlmo1z77ea)

[Sequence Diagrams 12](#_1wusp9khopqr)

[Navigation Flow Map 17](#_pvmyg62qwb7g)

[UI Mockup 18](#_5127iz9w2k8j)

## 

### 1. Purpose

Existing book cataloging apps such as Goodreads and StoryGraph are well-known for having boring design and clunky or lacking features–Goodreads’ homepage is overwhelming with lots of text and inconsistent margins; there are many advertisements that take up the page as well. Not only can the user write book reviews and follow other users, Tsundoku is unique from these platforms because 1) the user experience will be more streamlined and 2) the user can customize their own data-input experience however they want, with statistics generated from custom attributes that users can input, such as the number of times a book made them emotional or whether or not the book had a plot twist. These custom attributes will tailor the end-of-year review and data summarization so that it best suits the user, something that other platforms do not yet do.

While Tsundoku is similar to Goodreads and StoryGraph in that users can track their reading journey, with features such as writing reviews, the purpose of our project is to differentiate by providing additional parameters that will provide readers with a more in-depth insight into their reading habits and preferences. By allowing users to define custom parameters that they want to track, we can further diversify and allow the platform to suit the user. These additional metrics cannot be found in any other book-tracking platform and will be the game-changing differentiator that many users have been waiting for. All of the data and metrics will be combined and shown to users and the broader Tsundoku community with the Tsundoku Wrapped feature we are planning, which will show an aggregation of all of the tracked data on the platform given user consent.

### 2. Design Outline

This project will be a mobile application that allows users to write book reviews and follow other users, and is unique from other book-tracking platforms in that the user can customize their own data-input experience however they want, with statistics generated from custom attributes that users can input, such as the number of times a book made them emotional or whether or not the book had a plot twist. Our application will use the client-server model where one server simultaneously handles access from a large number of clients using React Native framework. The server will accept client requests, and fetch and store data in the database.

**1. Client**

1. The client, written with React Native, will be what users use to interact with the Tsundoku platform.
2. The client will communicate with the web server using HTTP requests, in the REST API format (JSON).
3. The response from the server will be used to populate the view shown to the user.

**2. Server**

1. The server will be written in Next JS (or Express) and will facilitate communications between the client and the database/authentication backend.
2. A reverse proxy (nginx) will sit between our backend server and the client to make the installation more secure (with HTTPS encryption with LetsEncrypt certificates) and possibly allow for load balancing in the future if needed.

**3. Database**

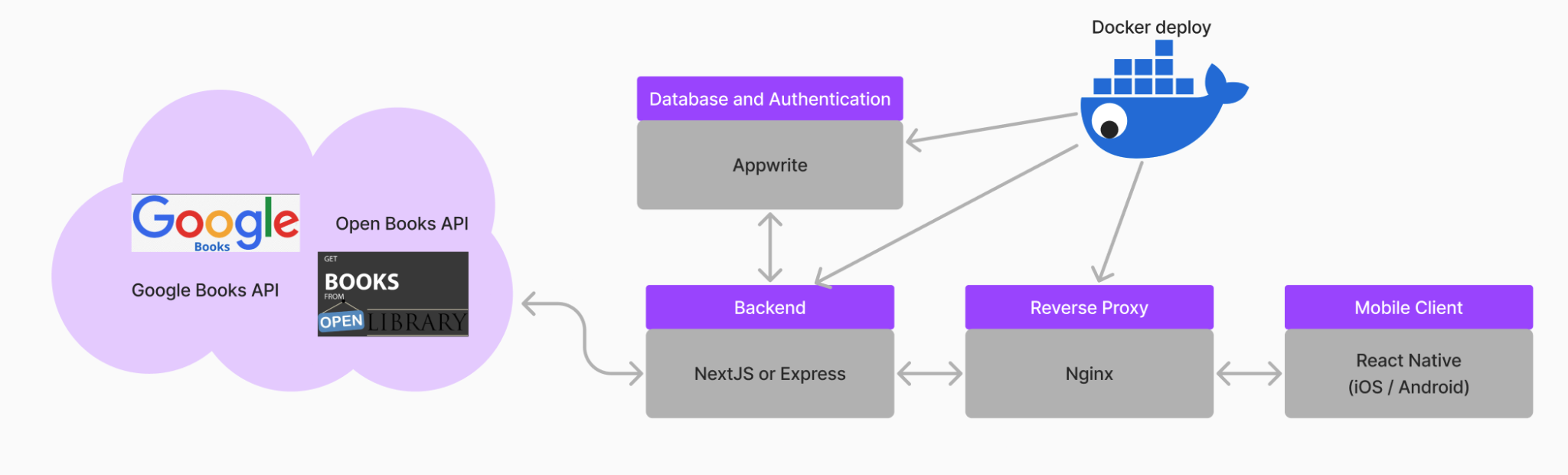
1. The database backend will be provided by Appwrite. Appwrite allows for “databases, collections, and documents”, like MongoDB, to store structured data. It also allows for defining relationships like classic SQL systems (MySQL, PostgreSQL), making it a good fit for Tsundoku with various relations between books, authors, users, reviews, and custom review attributes.
2. Appwrite provides a Javascript SDK that can be hooked into from the backend server (NextJS or Express), which will be used to manage the data being written to the database portion of Appwrite.

**4. Authentication**

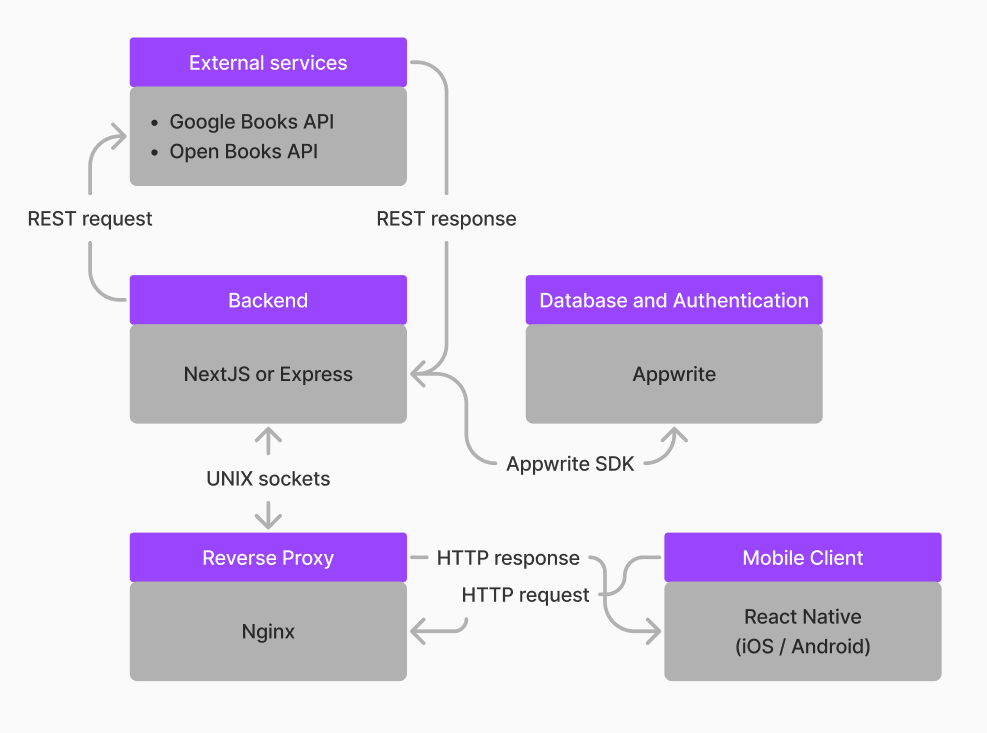
1. The authentication backend will be provided by Appwrite. The authentication implementation of Appwrite has been tested on many different projects utilizing Appwrite, which means we can save time and not make common mistakes re-implementing our own solution for authentication.
2. Appwrite provides a Javascript SDK that can be hooked into from the backend server (NextJS or Express), which will be used to manage the users saved in the authentication module of Appwrite.
3. The same SDK can be used in the React Native app to allow client applications to sign in to the Tsundoku platform.
4. The authentication module of Appwrite offers extensibility, allowing us to add more authentication methods and security features in the future, such as TOTP and OAuth sign-in.

**5. External services**

1. The Google Books API will allow the Tsundoku platform to query for and fetch information regarding new books. This will allow readers to quickly find and add information about the books they are currently reading, without having to stay on top of all of the books that are being published in real-time.
2. Google Books provides a REST API that the backend server can hook into and make requests to. The response will contain data about the book being queried.
3. As a fallback, the Open Books API will serve a similar function as the Google Books API. The Open Books API is provided by the Open Library project of Internet Archive, and also provides a REST API similar to the Google Books API.
4. Diversifying our book API endpoints should allow us to fetch information about more books, in case a particular API doesn’t have information regarding a given book, and should make the platform more resilient as a whole as we have a fallback to rely upon if one of the APIs experience unexpected downtime.
5. By developing a generic interface to communicate with those APIs, we can add on more book API providers in the future if necessary, allowing us to rapidly scale to fetch more book information (perhaps books from different countries that are not present in either of the APIs, etc.)



Protocol



### 3. Design Issues

#### Functional Issues

1. What user information do we need when signing up for an account?

● Option 1: Username and password only

● Option 2: Username, password, email address

● Option 3: Username, password, email address, phone number

Choice: Option 2

Justification: When a user sets up an account on our application, username and password are necessary to identify the user and provide them security on their account. Providing an email will allow for verification of the user when they sign up, and can prevent users from spam-making accounts. When the user wants to change their password, we will use the email to reset the user’s password. A phone number will not be necessary because it is a higher level of security that is not necessary for our app, as our app does not collect sensitive information such as birthday or age or location. In addition, using the phone number to send a verification code with SMS adds an extra monetary cost that we do not want to have. So, we will only use email, username, and password for our application’s sign up.

2. What kind of review properties do we allow for the user’s book reviews?

● Option 1: Set of default predefined properties

● Option 2: Customizable properties made by users

● Option 3: Set of default predefined properties in addition to custom properties made by users

Choice: Option 3

Justification: The default predefined properties for reviewing books are a star rating and an option of writing a text review, and we will let the user decide what custom properties they want to track. Having the default properties will streamline and simplify the process for users to review their books, as well as making it simpler to implement a filtering system that will include filters such as highest rating. Allowing the user to also create their own custom properties (such as “character lovability”, whether this reading session made me emotional, etc.) to track for books so the user can track what matters to them specifically. These properties would be either numerical, categorical, or boolean data. The user can customize their reviews depending on the kind of book they read. For example, they could create a “romance novel review template” with properties such as “how many times I cried.” Thus, we will allow the user to create custom review properties in addition to the default predefined properties.

3. How do we track a book that is not available in the app’s database?

● Option 1: Allow user to add the book that is not available in the database

● Option 2: Do not add it until it is included by our book API to prevent spam

Choice: Option 1

Justification: If a book is not available in our app’s database using the book API that receives book information, we will allow the user to manually add a book that is not available in our database. Despite the possibility of misuse, such as spam/fake books, by the user when adding books, it is important to allow for inclusivity and flexibility in our book database to cater to our user audience. We will also include measures to prevent misuse by having the user be prompted to provide verifying details of the book, which includes title, author, genre, and ISBN. Other users can also report the book that is manually added if it is actually nonexistent, which will encourage responsible usage of adding books manually to our database. Our app values a user-centric experience so it is important to accommodate user-added books to include a wider range of books and niche books. Thus, we will allow users to manually add books that are not already in our database.

4. ​​How should we allow the users to communicate with each other?

● Option 1: Private Messaging

● Option 2: Public Forum

● Option 3: Both

Choice: Option 2

Justification: The main purpose of allowing the users to communicate with each other in our app is to provide a sense of community and connection for readers. Private messaging can be dangerous for some users, as there can be incidents of spam or scammers. Thus, we will only allow for public forums to provide protection for users, and so administrators and supervise these public discussion forums. We will eliminate the choice of having both public discussion in forums and private messaging. This will protect our users from phishing or users with malicious intentions in private messaging. We want to provide users with a safe social experience while still connecting with other book readers.

#### Non-Functional Issues

1. What platform should we develop the app for?

● Option 1: Web app

● Option 2: Mobile app

● Option 3: Both

Choice: Option 2

Justification: We have a lot of goals we want to meet for this app, and we are worried that if we try to develop for multiple platforms, we will not be able to meet our requirements in time. For that reason, we have decided to work on just a mobile app for now. We will first satisfy our goals for the mobile app before deciding whether or not it is too ambitious to try and replicate our efforts on a web app. This will ensure a high quality mobile application to provide the user with a quality experience.

2. What framework should we use?

● Option 1: React Native

● Option 2: MERN stack

● Option 3: NativeScript

Choice: Option 1

Justification: We will use React Native as our framework for our application because of its cross-platform compatibility that can run on both iOS and Android platforms. This will allow for more accessibility for our users, and we do not need to have two separate codebases for each respective platform. Furthermore, our project team is more familiar with React Native and so we can utilize our existing skills in JavaScript and React.js. Without having to deal with the learning curve of learning a new framework, we can ensure a more streamlined and efficient code development for our application.

3. What database software should we use?

● Option 1: MongoDB

● Option 2: Appwrite

● Option 3: Supabase

Choice: Option 2

Justification: We will use Appwrite as our database software because not only does it offer a database functionality, it also provides authentication so we don’t need separate software for authentication, unlike Supabase and MongoDB. Appwrite also provides a user-friendly interface which will make it easier for us to manage our database. Furthermore, Appwrite is very flexible in that it supports both SQL and NoSQL databases. It also has authentication functionality which will ensure security and data encryption for our user’s data. Appwrite will provide a simple and secure way for us to build our mobile application. Thus, Appwrite is the best database software to use for our application and our development needs.

4. What API should we use to receive book information?

● Option 1: Google Books API

● Option 2: Open Books API

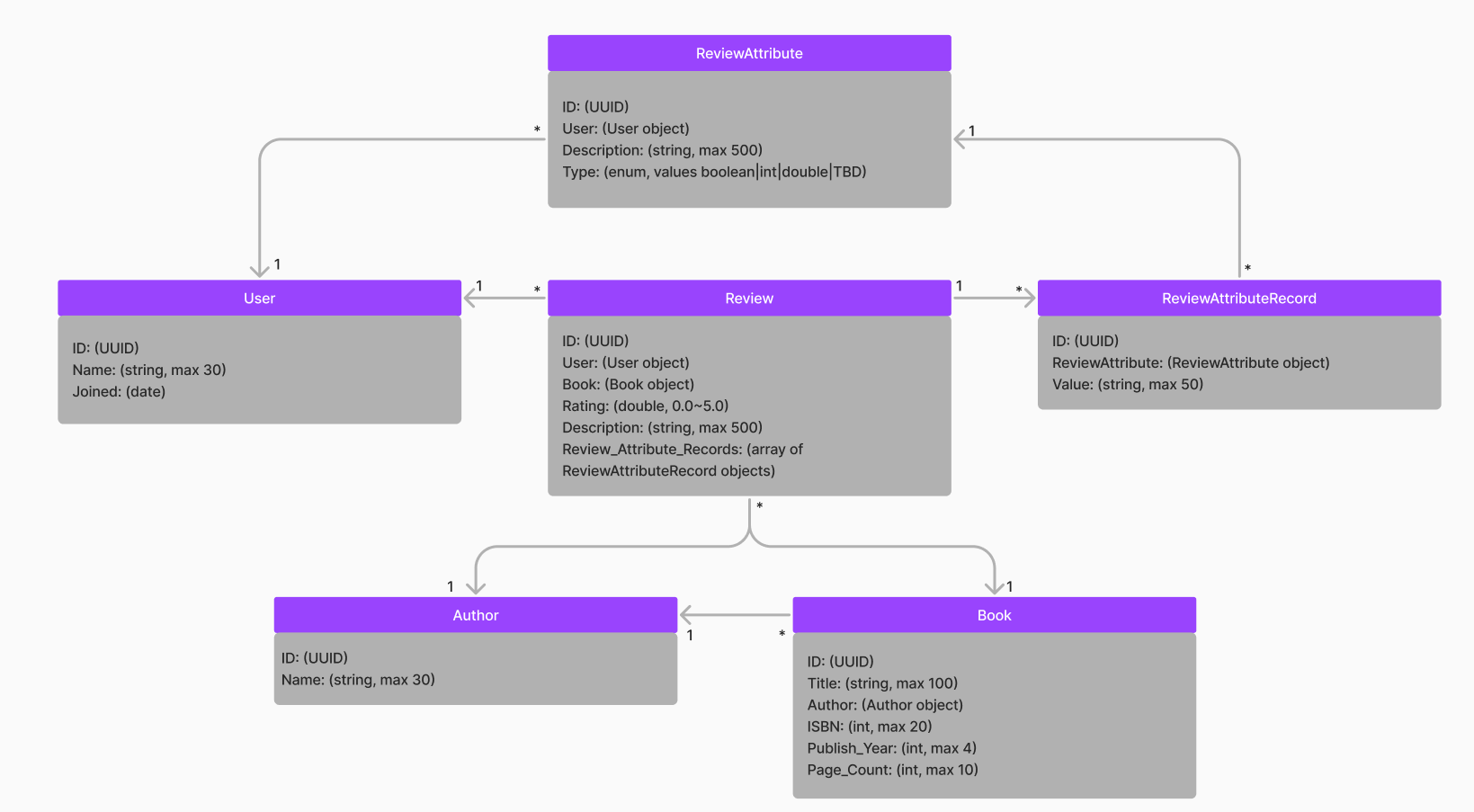
● Option 3: Both

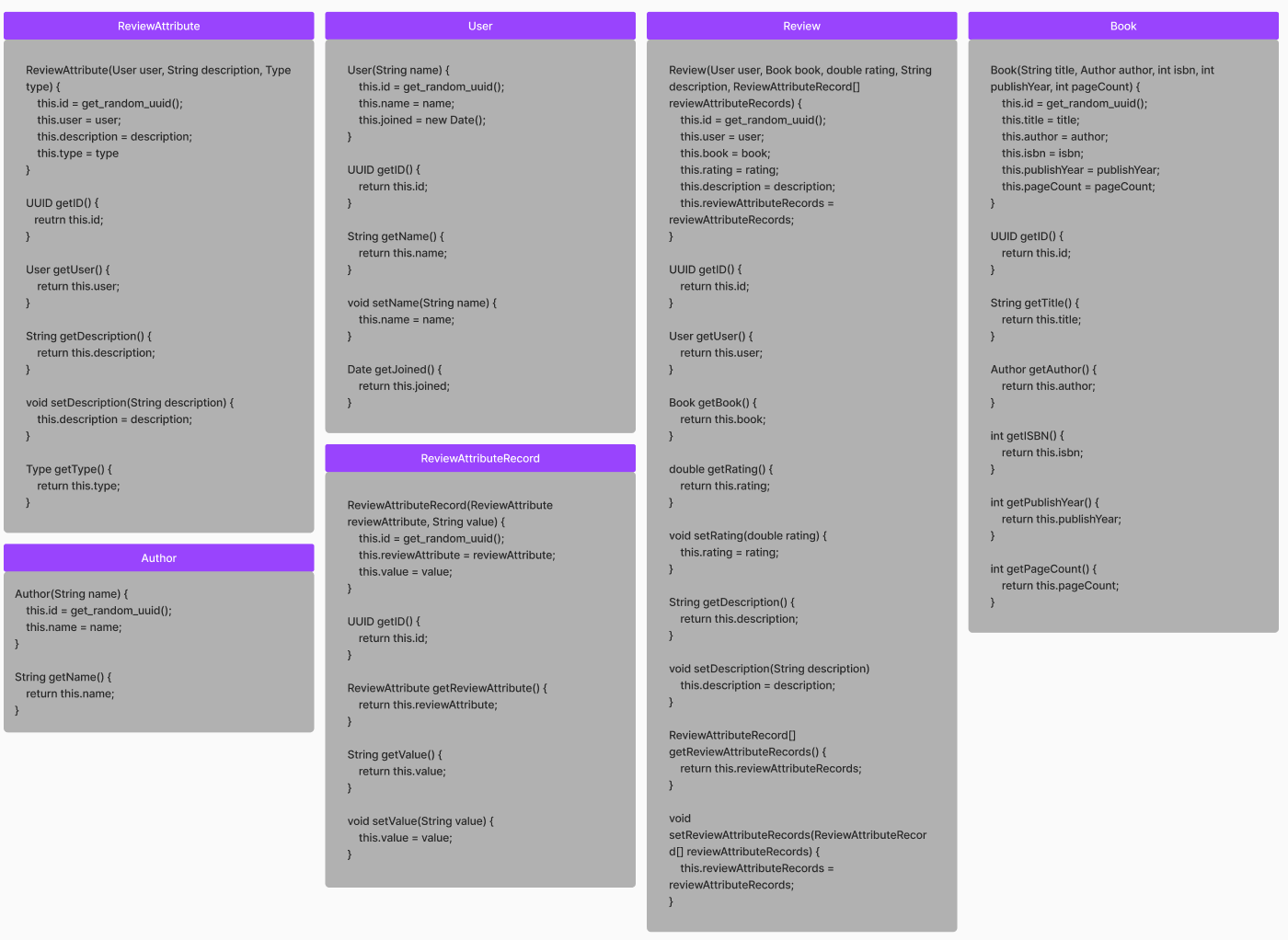
Choice: Option 3

Justification: By using both the Google Books API and the Open Books API, we can maximize the amount of books available in the database for our users. Google Books API is widely used and has a large library of books that includes descriptions and metadata (title, author, summary, publication date, and more). Open Books API offers features unique from Google Books API such as book datasets. By utilizing both these APIs, we can mitigate the limitations of using only a single API. This will allow our application to have more accurate and up-to-date information for a more reliable book library for the users. Thus, we will use both the Google Books API and Open Books API to leverage each of their respective strengths and allow for a broad range of books for our application’s database.

### 4. Design Details

#### Class Design





#### Class Description and Interactions Between the Classes

The following class design is based on the objects in our application. Each class has a list

of properties that are the characteristics of each object. There is also another diagram that directly corresponds with the class diagram that describes the constructors and methods for each class.

**User**

○ User objects are created when someone signs up in our application.

○ Each user will be assigned a unique user ID.

○ Each user will have a username, password and email.

○ Each user will have a date for when they joined.

**Author**

○ Author objects are created when books with authors not added to the author table are added to the book table.

○ Each author will be assigned a unique author ID.

○ Each author will have a string name with a max length of 30.

**Book**

○ Book objects are created from third-party API data. If users look up a book and it does not exist in the database, it queries a third-party API and adds it to our database.

○ Each book will be assigned a unique ID.

○ Each book will have a string title with a max length of 100 characters.

○ Each book will have an author object.

○ Each book will have an integer describing the book’s ISBN (International Standard Book Number) with a max length of 20 digits.

○ Each book will have an integer describing the year it was published with a max length of 4.

○ Each book will have an integer describing the page count with a max length of 10 digits.

**Review**

○ Review objects are created when a user creates a review.

○ Each review will be assigned a unique ID.

○ Each review will have a user object assigned to it.

○ Each review will have a book object assigned to it.

○ Each review will have a double ranging from 0.0 to 5.0 for the user’s rating of the book.

○ Each review will have a string description with a max length of 500 characters.

○ Each review will have an array of ReviewAttributeRecord objects for the review’s custom attributes created by the user.

**ReviewAttributeRecord**

○ ReviewAttributeRecord objects are created when the user creates a new custom attribute for reviews.

○ Each review attribute record will be assigned a unique ID.

○ Each review attribute record will have a ReviewAttribute object to describe it.

○ Each review attribute record will have a string value with a max length of 50 characters.

**ReviewAttribute**

○ ReviewAttribute objects are created when the user creates them.

○ Each review attribute will be assigned a unique ID.

○ Each review attribute will have a user object assigned to it.

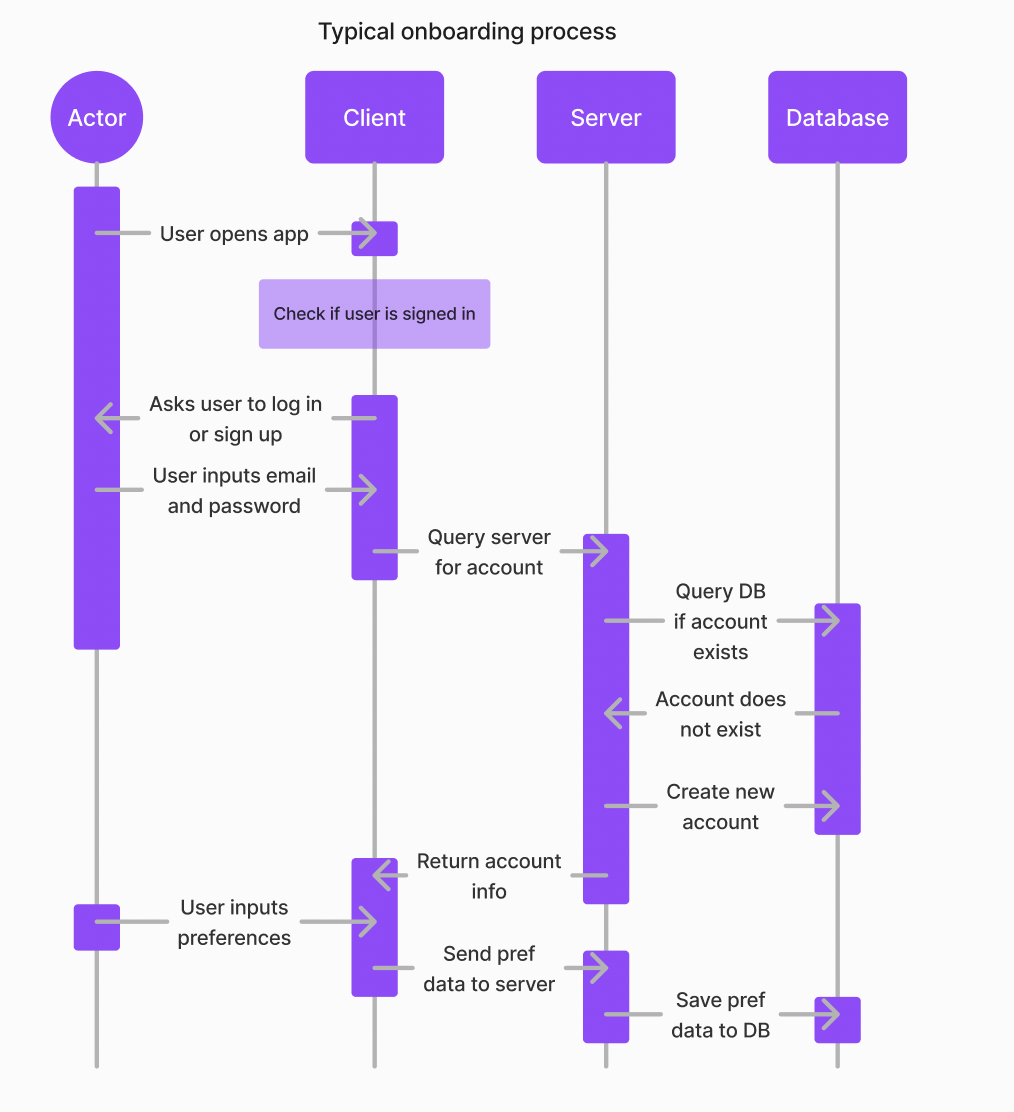
○ Each review attribute will have a string description with a max length of 500 characters.

○ Each review attribute will have an enum describing the type of data it represents (boolean, int, string, etc.).

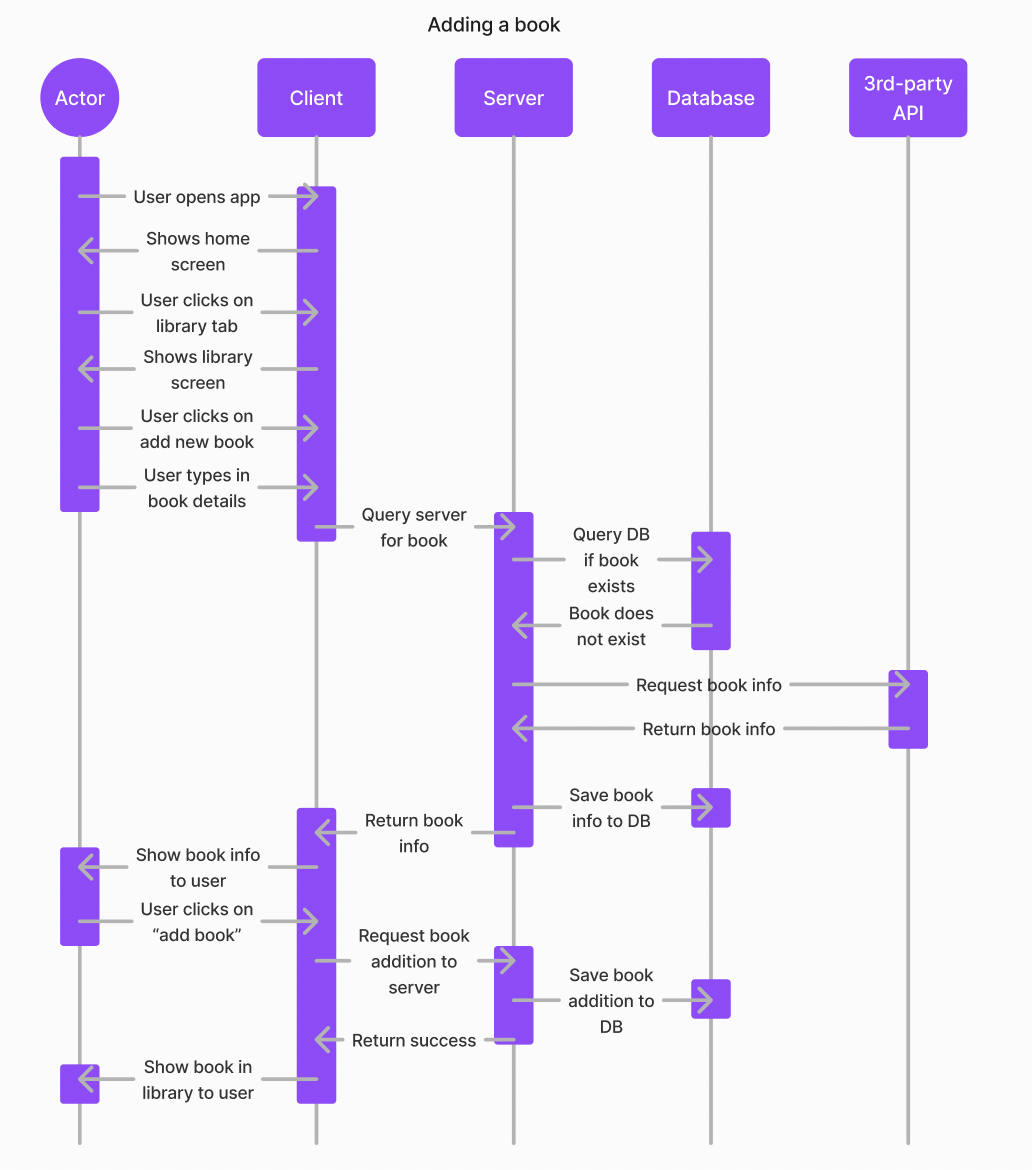
#### Sequence Diagrams

The following diagrams depict the sequence of the major events in this application in a client-server model, which includes user sign-in, adding a book, creating a review, searching for books, and creating a forum post.

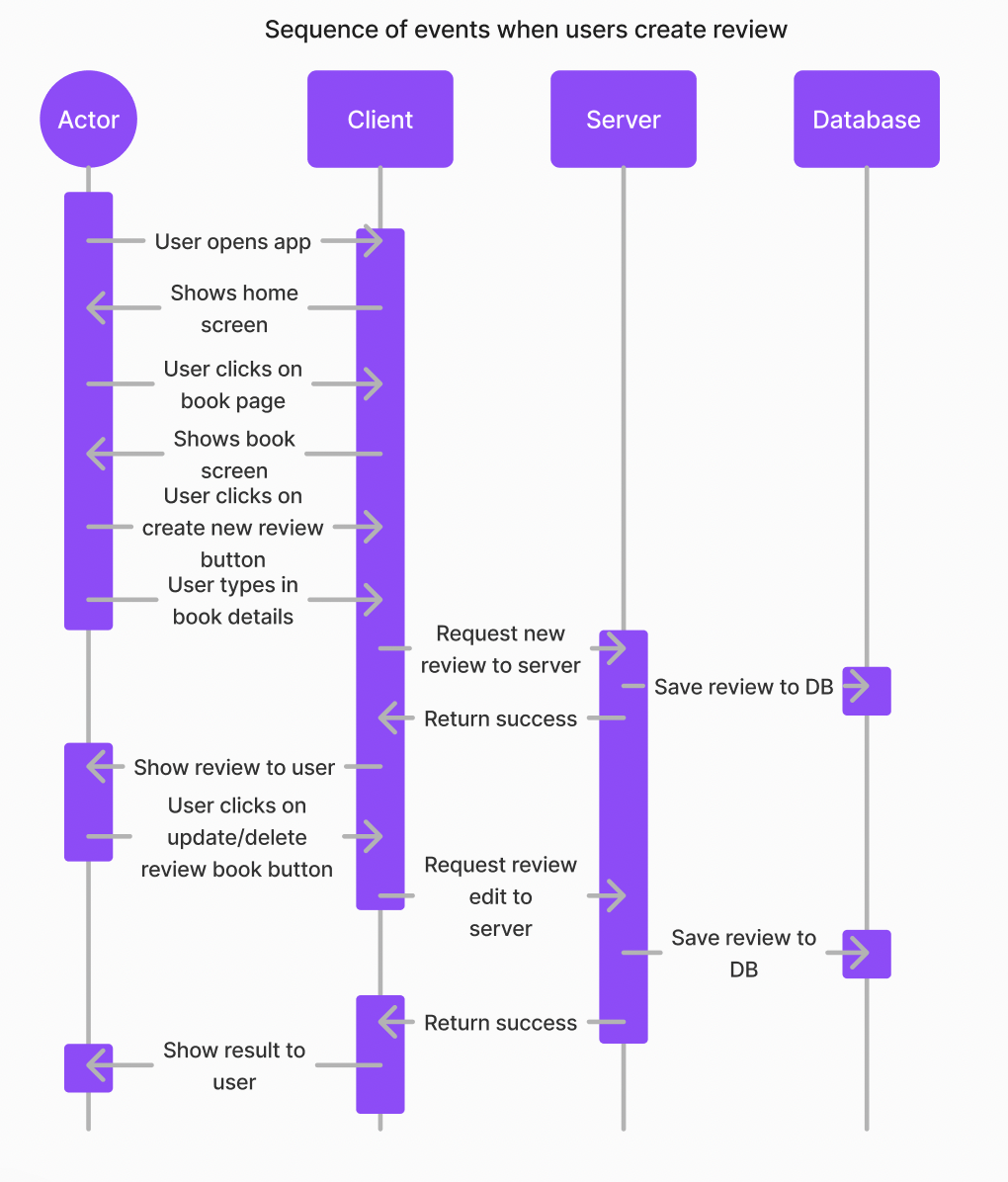
1. Sequence of events when user signs in



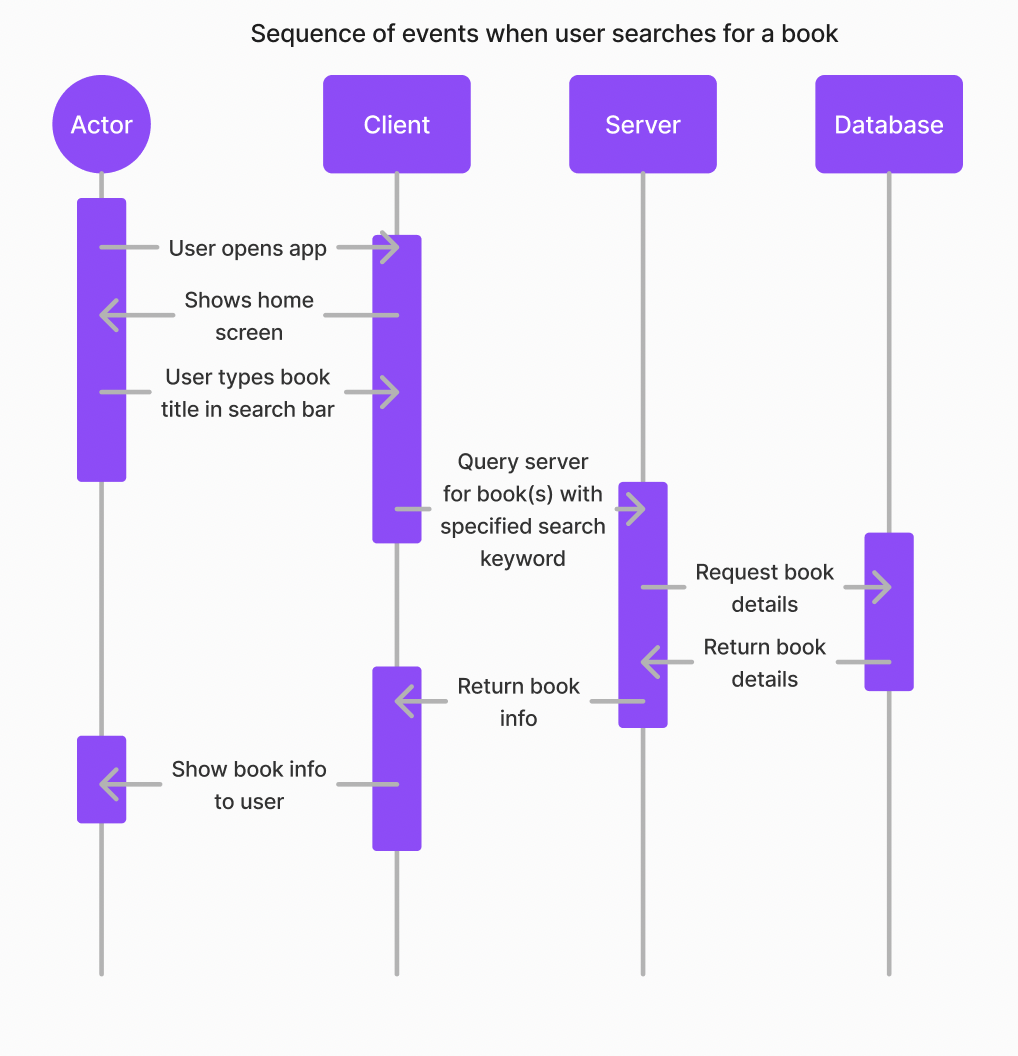
1. Sequence of events when user adds a book



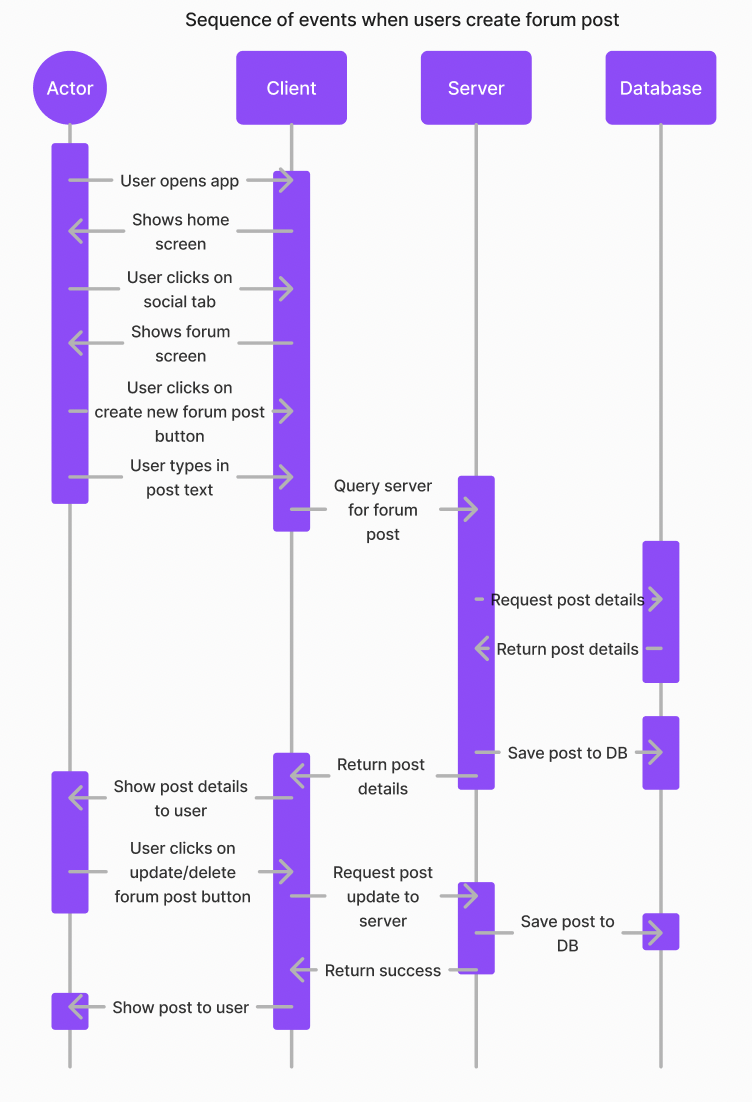
1. Sequence of events when user creates a book review



1. Sequence of events when user searches for a book



1. Sequence of events when user creates a forum post



#### **Navigation Flow Map**

Our navigation design emphasizes simplicity and ease of access, as shown in our diagram. All modules after sign-in are accessible from the navigation bar at the bottom of our mobile app. We encourage the user to sign in to the homepage with their account. If the user does not already have an account, they must create one by entering their user information. The homepage contains the book lists of the user. The navigation bar on every page contains all the

accessible pages in the mobile application that includes a social tab, profile page, library, and homepage. The homepage contains the search bar at the top which allows the user to find their desired books. Whenever the user searches for a book, their recommended book list is updated based on this search query which is displayed in the application homepage. If a user selects a book, they will be directed to the bookpage which displays the book details. From the book page, the user can add it to a list or create a review. The user can also engage with social features which are accessible in the navigation bar, such as creating a forum discussion post.



#### UI Mockup

| Library | Discover |
| --- | --- |
| Library after clicking track (+) button | Community - friends tab |
| Community - book clubs tab | Community - challenges tab |
| Book list | Book info page |
| Profile - activity tab | Review |