DESIGN DOCUMENT

Team 22

Logesh Roshan Ramadoss

Myeongsu Kim

Piyush Juneja

Shobhit Makhija

TABLE OF CONTENTS

Purpose

Along the evolutionary course of technology, most forms of audio-visual based media have constantly pushed and grabbed a larger portion of our hobbies year after year. The platforms for movies and TV shows are numerous, with streaming services, movie rentals, discussion forums and several others. The sheer number and extensive nature of this form of media has pushed out more traditional counterparts like literature.

The primary goal of PickaBook is to provide a platform for book enthusiasts and newbie readers alike, in an attempt to close the divide between the traditional and digital forms of media. PickaBook is a web-based service that will provide a review and discussion platform for users to participate in a proactive community where they can talk about their favorite books, as well as explore new and popular works. As icing on the cake, not only will users be part of the larger community, but will also receive personalized recommendations based on their preferences - making for the ultimate book-lover experience.

Functional Requirements

1. User account.

   As a user,

   a. I would like to be able to Register and Create a new account.
   b. I would like to be able to link my Google or Facebook account.
   c. I would like to be able to reset my password.
   d. I would like to login and logout from my account.
   e. I would like to set a profile picture.
   f. I would like to be able to set my profile's privacy to control if other users can see my profile.

2. User Profile

   As a user,

   a. I would like to keep track of my profile.
   b. I would like to see a collection of my favorite books.
   c. I would like to see my favorite genres.
   d. I would like to see all the authors I'm following.
   e. I would like to see my profile and information.
   f. I would like to view any public users' profile.
   g. I would like to be able to set my profile private.
   h. I would like to be given recommendations on what to read in the future.

3. Viewing Content

   As a user,

   a. I would like to search for any book.
   b. I would like to see a page with the information and ratings + reviews for the book.

c. I would like to add a book to my favorite collection.
d. I would like to search for an author.
e. I would like to get the author of a book.
f. I would like to be able to follow the author to get notifications for any new releases.
g. I would like to see all the releases from an author.

4. Book Reviews

   As a user,

   a. I would like to be able to write reviews for books.
   b. I would like to see all the reviews of a book.
   c. I would like to give ratings for books.
   d. I would like to see the ratings of a book.
   e. I would like to see all the reviews I have given.
   f. I would like to like or dislike reviews given by other people.

5. Exploration System

   As a user,

   a. I would like to be able to explore different titles.
   b. I would like to browse through new releases.
   c. I would like to browse through the top rated books.
   d. I would like to filter by the year of publication.
   e. I would like to filter by genre of books.

6. Back End

   As a developer,

   a. (If time allows) I would like the server to crawl the web for new books.
   b. I would like to keep track of user preferences in the database.

c. I would like to recommend new books based on user preferences.

Non-Functional Requirements

1. Performance
    a. I would like the application to support at least 1,000 users.
    b. I would like the application support about 50,000 book records.
    c. I would like the application to respond in 500 ms for any get request.
    d. I want the server and client to gracefully handle any error or failure.
    e. I want the design to scale well without requirements to modify the design for excess load.

2. Server Requirements
    a. I would like the server to support real time server client communication.
    b. I would like the server to be able to store user data into the database.
    c. I would like the server to follow RESTful API to integrate with a front end.
    d. I would like the server to store static files.

3. Architecture & Usability
    a. I would like the app to require minimal effort from users to understand and navigate the app.
    b. I would like to adhere to principles of modularity to be able to update and improve the app as need be.

4. Security
    a. I would like to allow users to set their profiles' privacy setting.

b. I would expect all requests to the API will need to be authenticated.
c. I would like to ensure data protection within the database by only letting the users view information that they need to.
d. I would like to maintain only encrypted passwords for account protection.

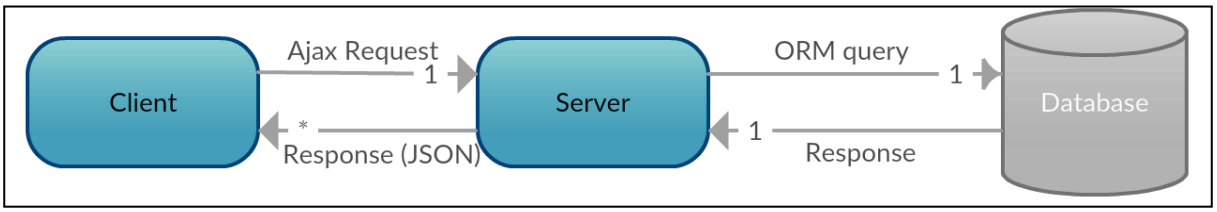
Design Outline

This project will be a web application that allows users to discuss and review books, as well as find new books to read. The application will use a client-server model, with a single server dealing with multiple clients. The project will be implement the Model-View-Controlled (MVC) design pattern. The server will accept requests from the clients, and access, store or modify data in the database as per the request.

1. Client
    a. The client provides the user with an interface through which they can interact with our system.
    b. The client will send promise based, ajax requests to the server.
    c. Client receives ajax responses in the Representational State Transfer (REST) format and renders the data in the user interface.
2. Server
    a. The server will deal with all interactions between the application and the database.
    b. The server will receive and handle ajax requests from clients.
    c. Based on client requests, the server will validate a request and run the appropriate query to modify the database or access the database to return a response.
3. Database
    a. A relational database will be used to store all the data relating to the application, such as book information, user information, reviews, etc.
    b. Database will interact with the server through queries. Updates and data retrieval will be carried out by the server on the database through such queries.

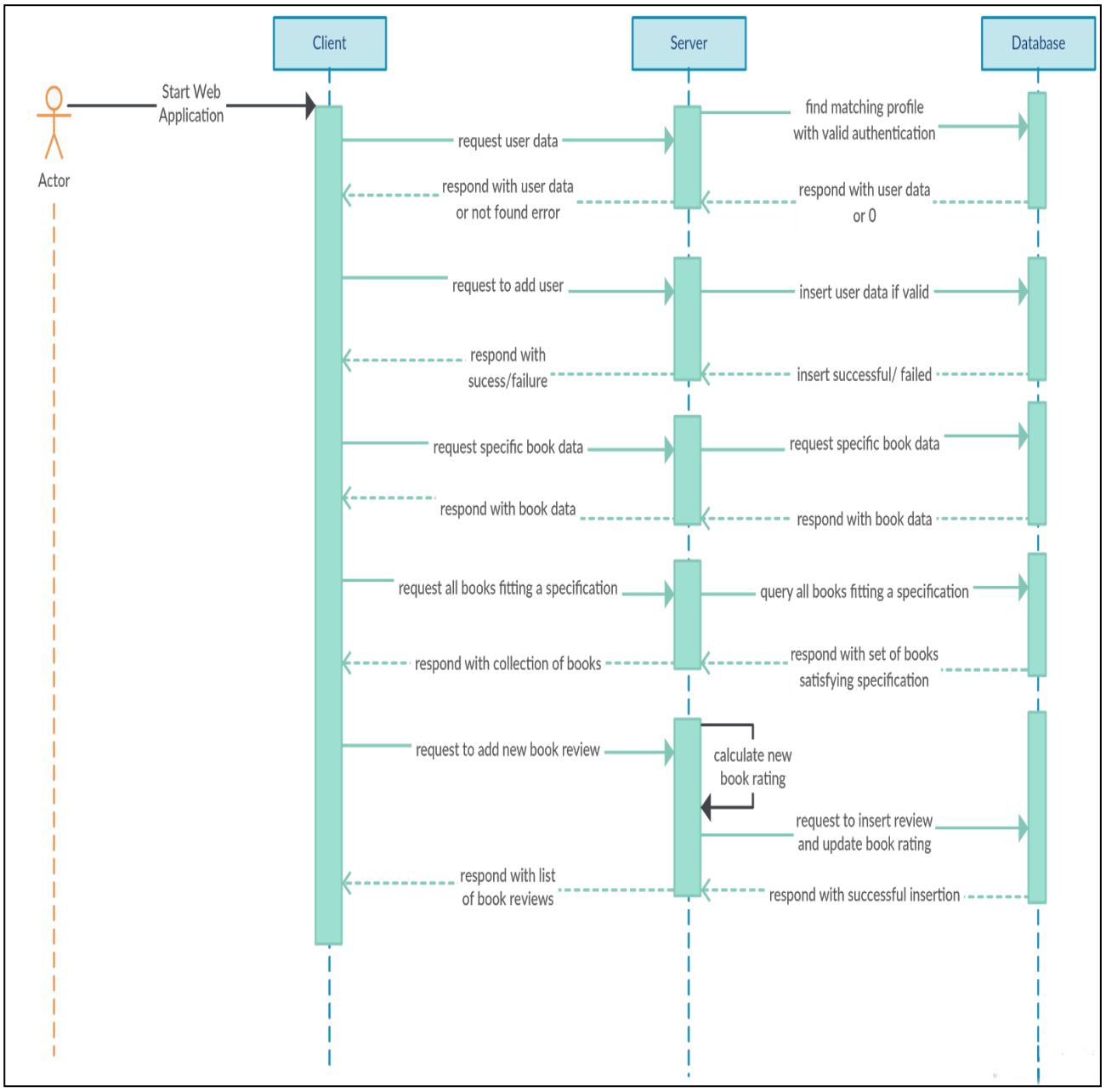
High Level Overview

The client and server will communicate using ajax requests and responses. Client will send GET requests to access any data from the database through the server, and will send POST requests to update the same. The server will wrap the data from the database into a JSON format in order to follow the REST API, this JSON will then be sent to the client.



Sequence of Events Overview

The sequence diagram below shows the general flow of interaction in our Model View Controller framework. The initial start of the sequence will be from the client using the web app. Whenever the user interacts through the client side, requests will be sent to the server. The server will handle the requests by sending queries to the database through its ORM and receiving responses from the database, if any. User actions like logging in, looking up profiles, books or reviews, exploring new titles will involve requests to the server, where the server will access the database and return the requested data. Other actions like posting reviews, updating user info, asking questions, favoriting a book will send post requests to the server, which will result in the server modifying or updating the database. The server will then respond with the updated data from the database back to the client.


Client
Server
Database
Actor
Start Web Application
request user data
find matching profile with valid authentication
respond with user data or not found error
respond with user data or 0
request to add user
insert user data if valid
respond with sucess/failure
insert successful/ failed
request specific book data
request specific book data
respond with book data
respond with book data
request all books fitting a specification
query all books fitting a specification
respond with collection of books
respond with set of books satisfying specification
request to add new book review
calculate new book rating
request to insert review and update book rating
respond with list of book reviews
respond with successful insertion

Design Issues

Functional Issues

- Do we require users to login to our service, and if so, then how do we implement it?

  Option 1 - Use unique username and password.

  Option 2 - Username and password along with email.

  Option 3 - Username and password along with email or link your facebook account.

  Choice: Option 3
  Reason: Requiring a username and password for creating an account is mandatory. The addition of an email address will allow for options like resetting one's password and preventing a single user from making multiple accounts. Linking users to their facebook account will allow for easier registration, since the user does not need to remember their details.

- How should we calculate the ratings for Books?

  Option 1 - Calculate average of all ratings when new rating is added.

  Option 2 - Use a pseudo running average calculator.

  Choice: Option 1
  Reason: The main reason for the choice is simplicity. Calculating the new average will be easier to execute through a database query. The pseudo running average will require keeping track of other information and running the calculations ourselves, which could

result in errors. Option 1, however, is not very scalable.

- How do we allow discussions relating to a book?

  Option 1 - Real Time Chat Room for every book.

  Option 2 - Review Board for each book.

  Option 3 - Review Board + Q&A for each book.

  Choice: Option 3

  Reason: The rationale for choosing this option is two-pronged: to allow for a platform that enables an open discussion amongst different users, but at the same time, also pin answers to certain important questions (asked by users on the platform) above the reviews to allow for better readability. Furthermore, our application is not a social network, so a real time chat application does not align with our platform.

- How do we Recommend Books to users?

  Option 1 - Use user's history of read books.

  Option 2 - Use user's favorite book list.

  Option 3 - Use a combination of favorite books, authors, book ratings and user's genre preferences.

  Choice: Option 3
  Reason: Using a single source of information for a user will be easy to implement but will result in poor recommendations. Combining all of the users preferences will allow us to get a holistic image of their likes, allowing for better recommendation.

- Should there be an option to create a personalized Top 10 for each user?

  Option 1 - Yes

  Option 2 - No

  Choice: Option 2

  Reason: Given the fact that we would also like to implement a Favorites list as well as an Already Read list for each user, making a separate list for each user's top 10 choices seems redundant. For this reason, we decided to give priority to the first two elements over this one.

- Should we allow users to add new books?

  Option 1 - Yes

  Option 2 - No

  Choice: Option 2
  Reason: We want to prevent users from uploading their own personal works as books or add books that don't currently exist. Furthermore, this would require us to screen any new addition for questionable content, which is not part of the scope of this project.

- How do we go about implementing user reviews?

  Option 1 - A binary LIKE DISLIKE system.

  Option 2 - Allow users to rate out of 5.

  Option 3 - Allow ratings in a scale of 100.

Choice: Option 2
Reason: A like/dislike regimen does not provide a reasonable rating estimate of the quality of the work due to its binary nature. On the flip side, a score out of 100 covers too broad of a range, which can skew ratings due to outliers in user preferences.

Non-Functional Issues

- How will we host our Web App?

  Option 1 - Amazon Web Services

  Option 2 - Heroku

  Option 3 - Microsoft Azure

  Option 4 - Digital Ocean

  Choice: Option 2
  Reason: Heroku is a hosting provider that supports Django as part of a Platform as a Service (PaaS) offering. In such hostings, one does not need to worry about the production environment, such as web server and balancers, as the platform takes care of them. This allows for easy deployment, enabling us to focus on the application itself.
  Furthermore, Heroku's free package is more attractive than a few of the other options.

- What type of database will we be using for our data?

  Option 1 - MySQL

  Option 2 - MongoDB

  Option 3 - PostGreSQL

Option 4 - Cassandra

Choice: Option 3
Reason: NoSQL is usually more favored for cases where the data in unstructured. Given that our data is fairly well structured, we decided to go with a Relational Database. For the most parts, MySQL and PostGreSQL are fairly similar. They are both open source, follow ACID properties, and have great resources. The final choice was simply one of preference and familiarity, which led us to PostGreSQL.

- What languages/libraries should we use for implementing the front-end of the app?

  Option 1 - ReactJS

  Option 2 - AngularJS

  Option 3 - VueJS

  Option 4 - None, just use basic HTML/CSS

  Choice: Option 1
  Reason: React is a well established library that was released in 2013. It is very newbie friendly as it is easy to learn and start developing. The language also has a lot of well documented packages that are easy to integrate and use with any project. React is component based, where each component is built as an encapsulated unit, with its own JSX components and props. Furthermore, React ensures better performance with the use of virtual DOM, which is a lightweight copy of a real DOM. The virtual DOM keeps track of updates and only the components that were updated in the virtual DOM will be reflected in the real DOM.

- What web-frameworks could we use to develop our web app?

Option 1 - PHP

Option 2 - Django

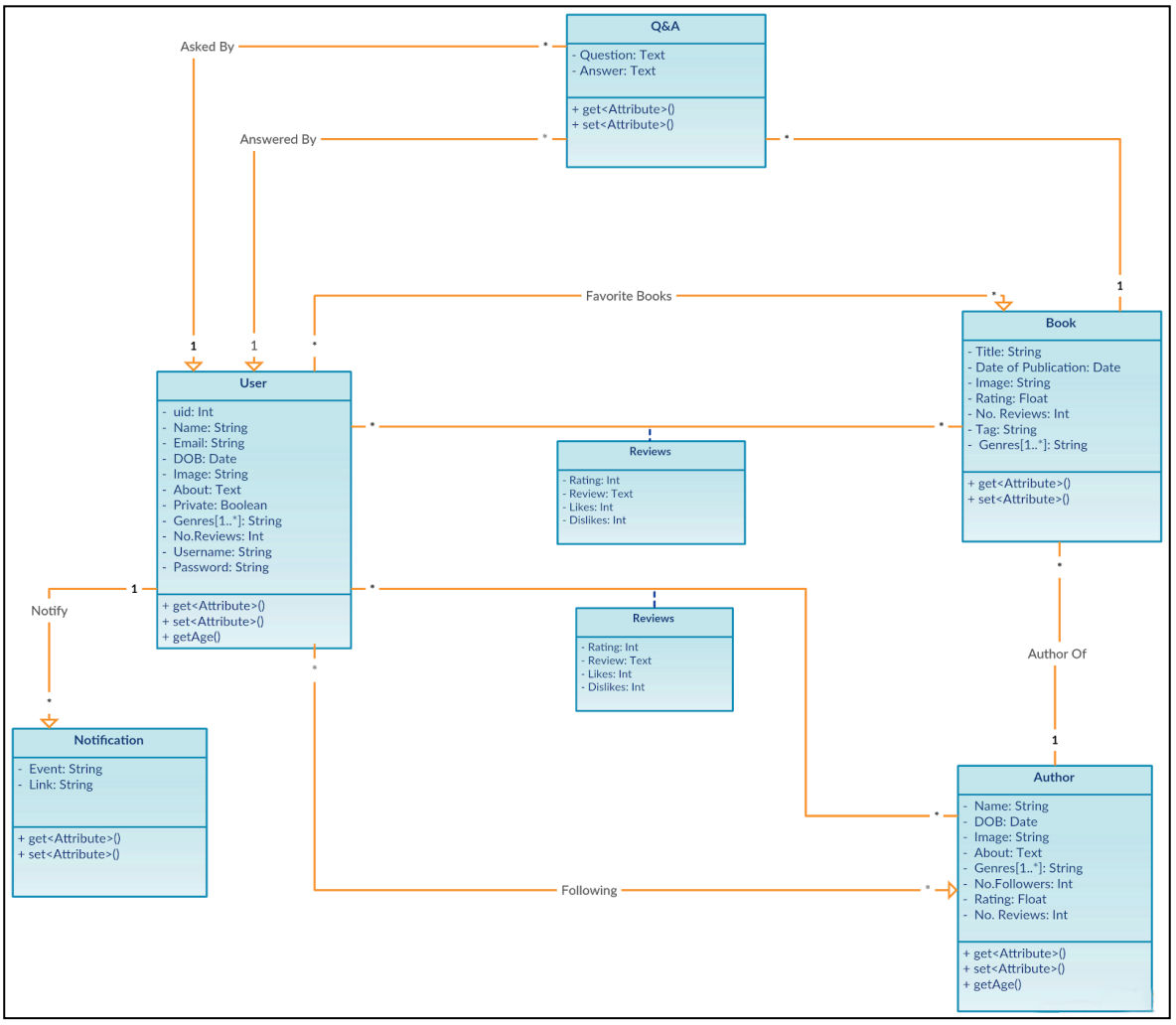
Option 3 - Node.js

Option 4 - Ruby on Rails

Choice: Option 2
Reason: We chose Django since it is a Python based framework that we're all comfortable with. Django follows the DRY (Don't Repeat Yourself) development philosophy, allowing us to carry out rapid development of the project. It comes with its own Object Relational Mapper and also has great out-of-the-box security systems that deal with SQL injections, cross-site scripting and other issues. Finally, the framework has a large community and forum that will be highly useful in finding packages and resolving any developmental issues that we many run into.

Design Details

Class Design



Descriptions of Classes and Interaction between Classes

Each of the class reflects an object of our application. Each of these classes hold a list of attributes which determines the features of an object of that class.

- User
    - A user object will be created every time a viewer creates an account.

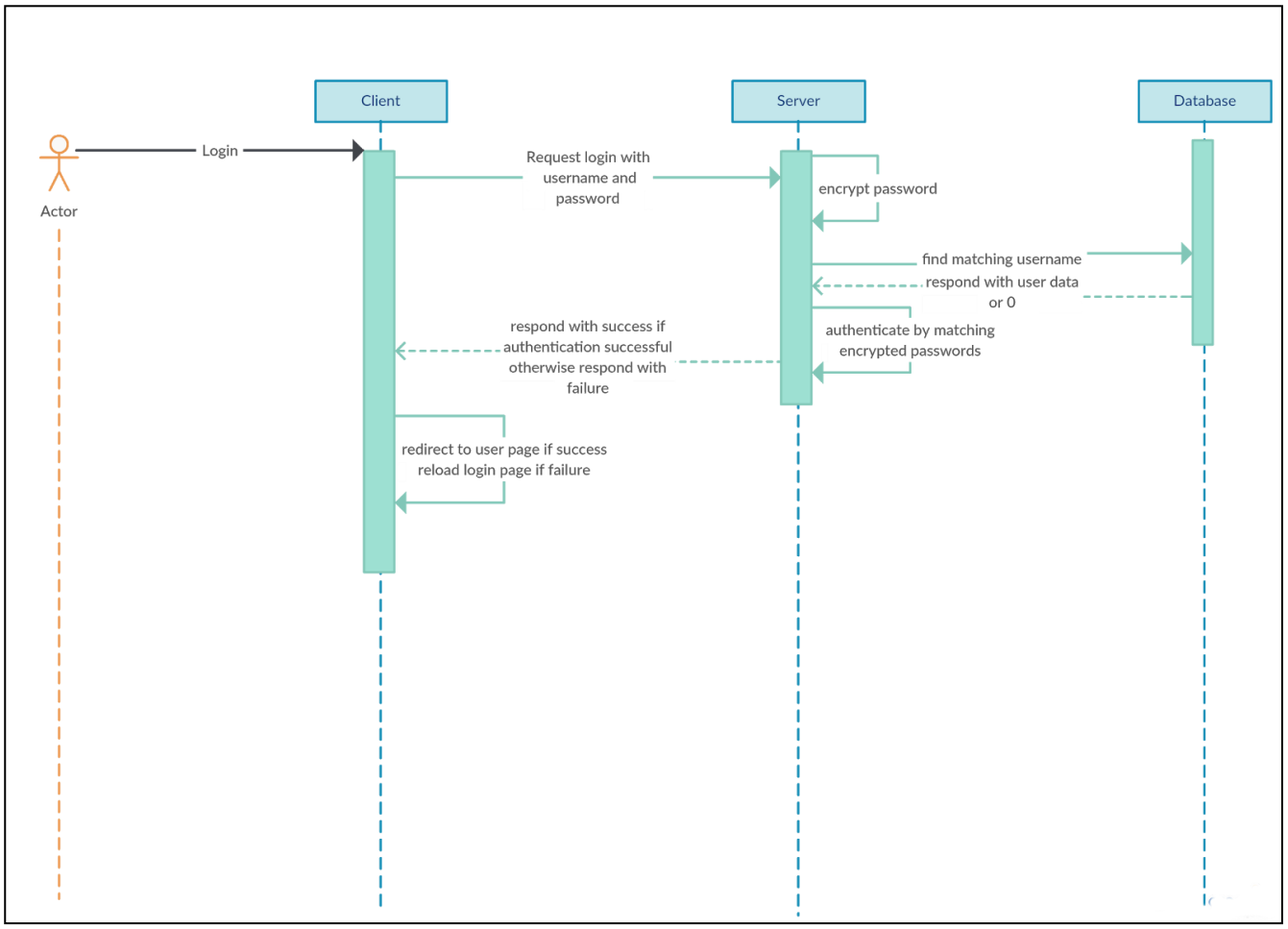
    - Each user will have a unique username.
    - Each user will also have an email and a password for security purposes.
    - Each user will also have their name, Date of Birth, a profile picture, and a description section about themselves.
    - Each user will also have a list of their favorite genres that they enjoy, provided during registration.
    - Each user will have a list of favorite books shown by its relation to the books class.
    - Each user will have a list of their following Author shown by its relation to the books class.
- Book
    - Each book object will be created by an admin or through crawling.
    - Each book will have a unique name.
    - Each book will have an author, date of publication, genre, image and description.
    - Each book will also have a rating score based on the ratings given to it by users.
    - The genre and rating will be used to display the books in the explore section.
    - Each book will have a list of reviews and question and answers created by users for discussions.
- Author
    - Each author will be created either by an admin or when one of their books are added.
    - Each author will have a unique name.
    - Each author will have a Date of Birth, Description, and image.
    - Each author will have a rating score from users.
    - Each author will also have a list of genres that they worked on.
    - Each author will have a list of their books.
    - Each author will have a list of reviews from users.

- Reviews
    - Each review will have a title and review content.
    - Each review will have a rating score [0-5]
    - Each review will also have a like, dislike score, so as to order them.
    - Each review will have a user and a book or author that it relates to, i.e. they will be foreign keys.
- Q&A
    - Each object will have a Question.
    - An object may or may not have an answer.
    - Each instance will also have a like dislike score, so as to order them.
    - Each object will have a user and a book that it relates to, i.e. they will be foreign keys.
- Notification
    - Each notification will have an event string stating what the event is.
    - Each notification will have a link to the notification location (book or Q&A).
    - Each notification will have a userID as a foreign key.

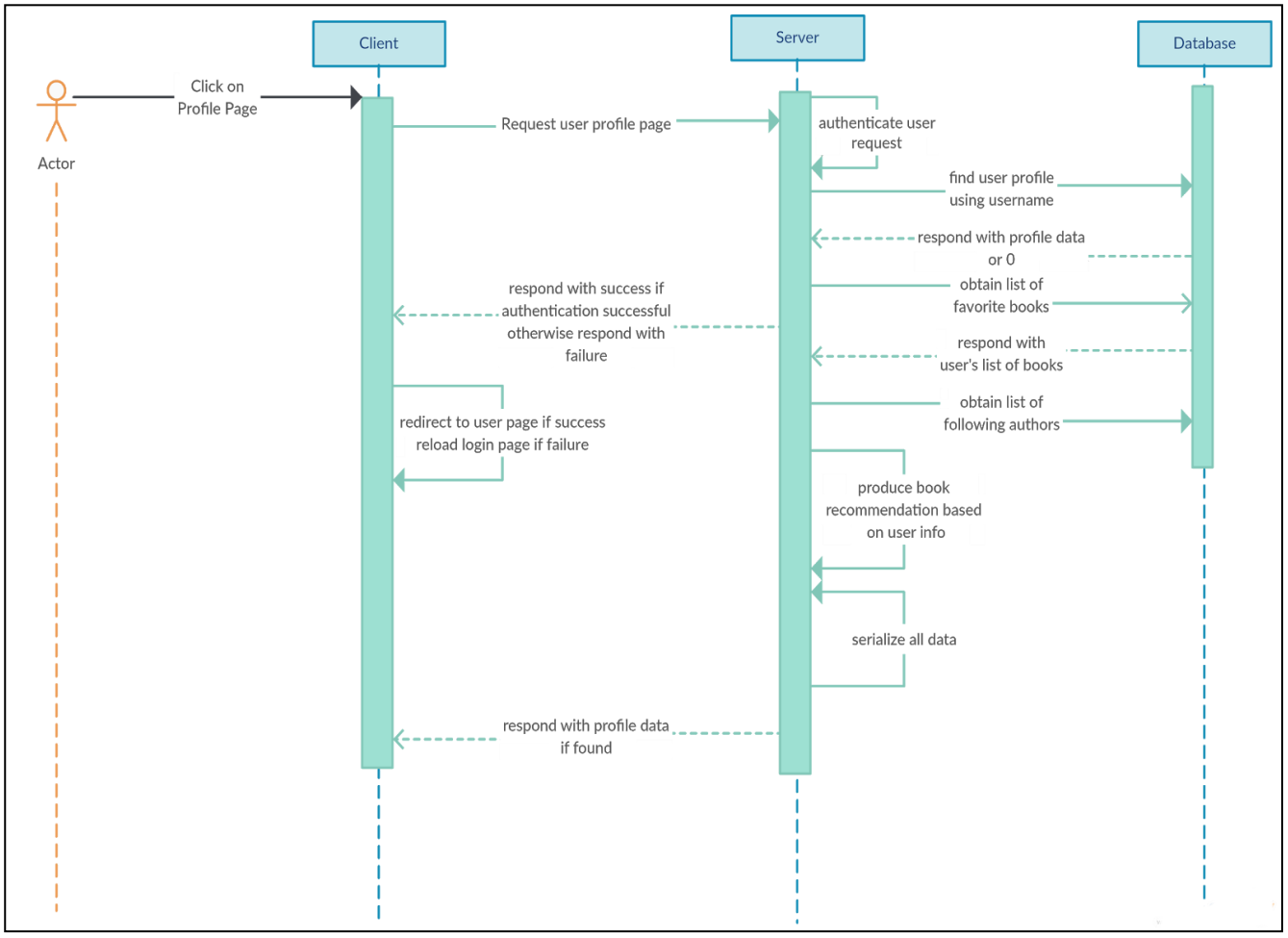
Sequence Diagrams

The major MVC events of the application are represented in the following sequence diagrams. This includes the act of logging in, signing up, accessing the exploration section, looking up a book, writing a review, writing a question and accessing user profile. Each sequence depicts how the message interactions will take place starting from the client all the way to the database. The general flow is that the client will request some data, the server will then grab this data from the database, process and serialize it and then respond back with this data. The frontend will then get updated with the response and produce a new UI.

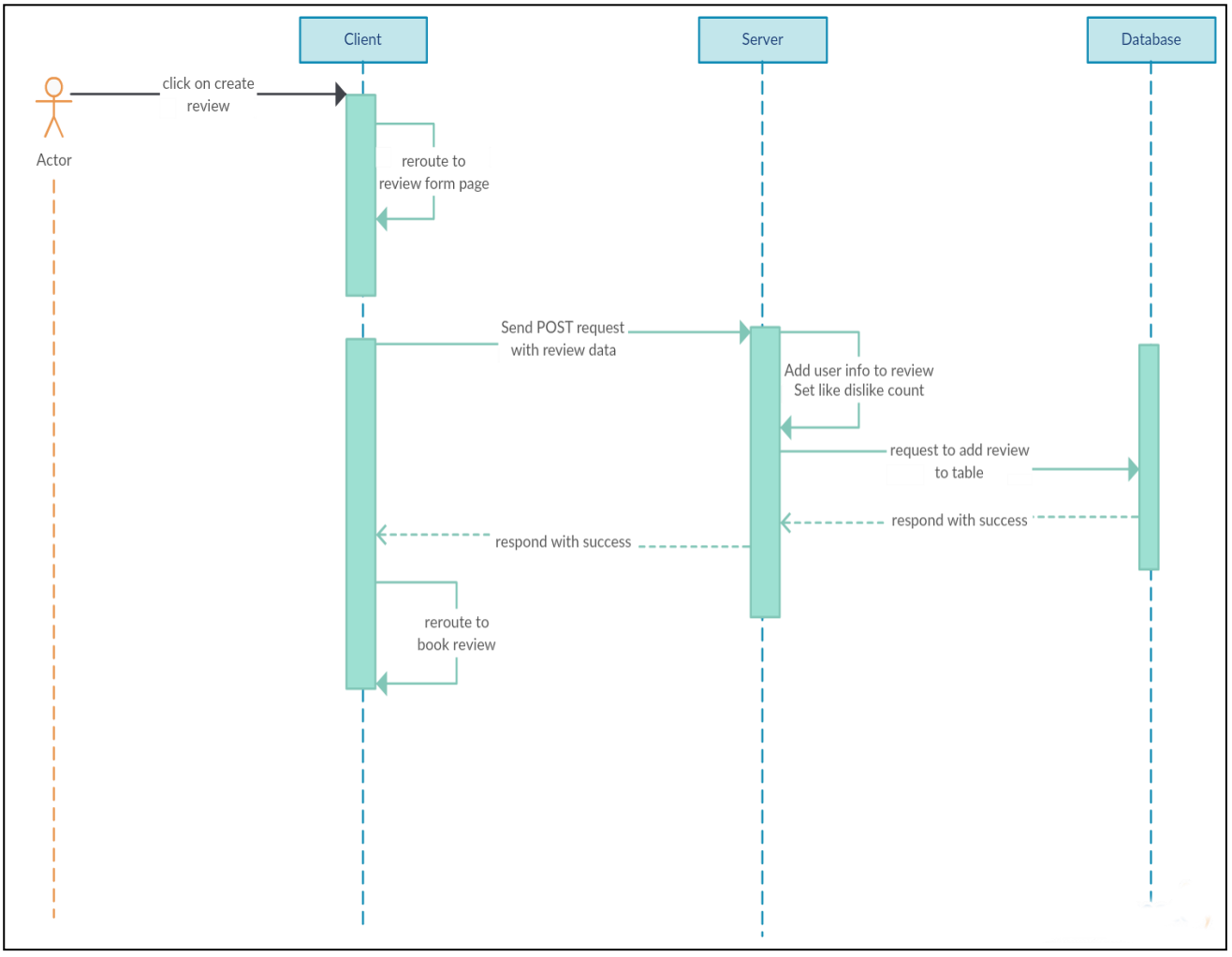
1. Sequence Diagram of User Login



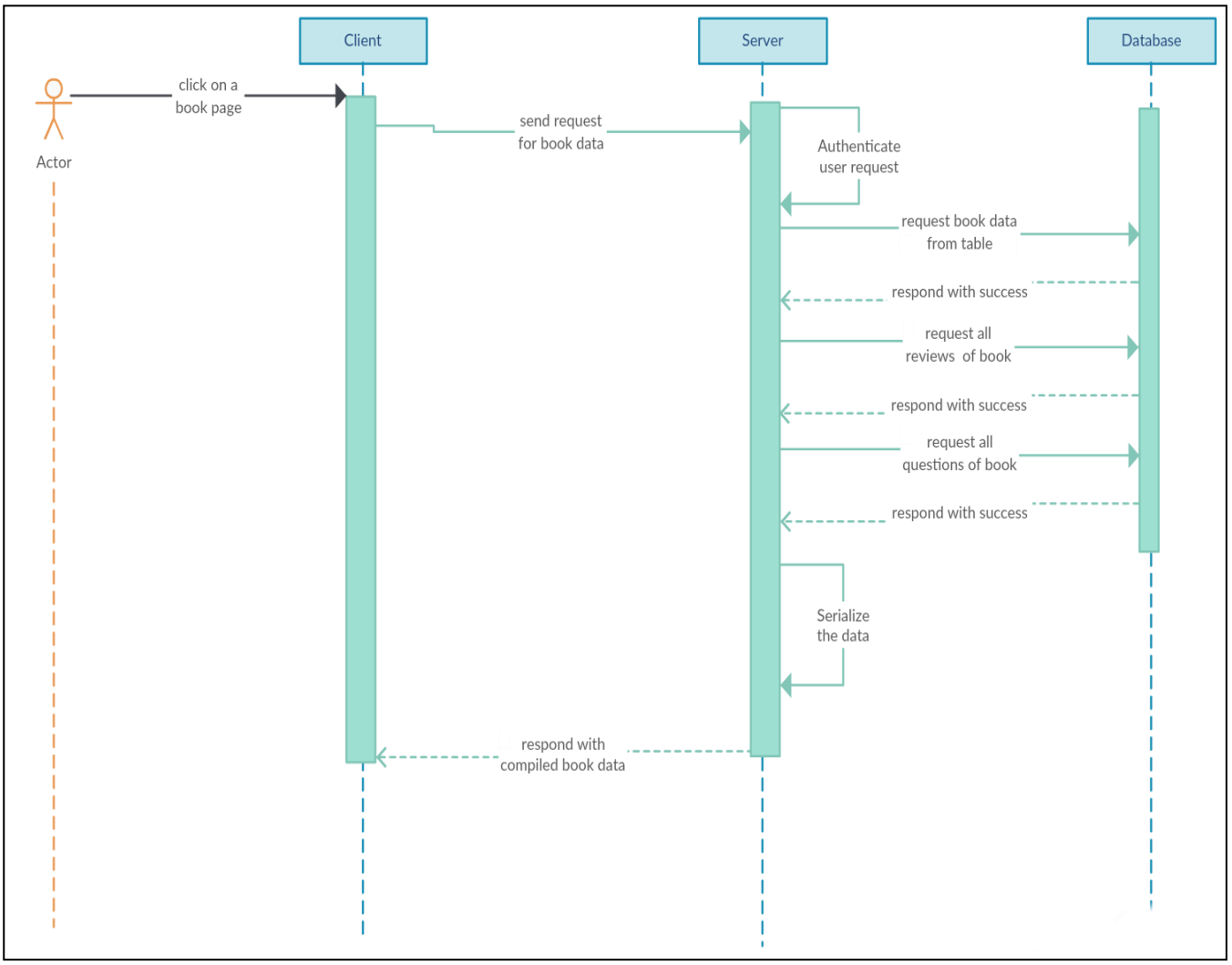
2. Sequence Diagram of loading user page



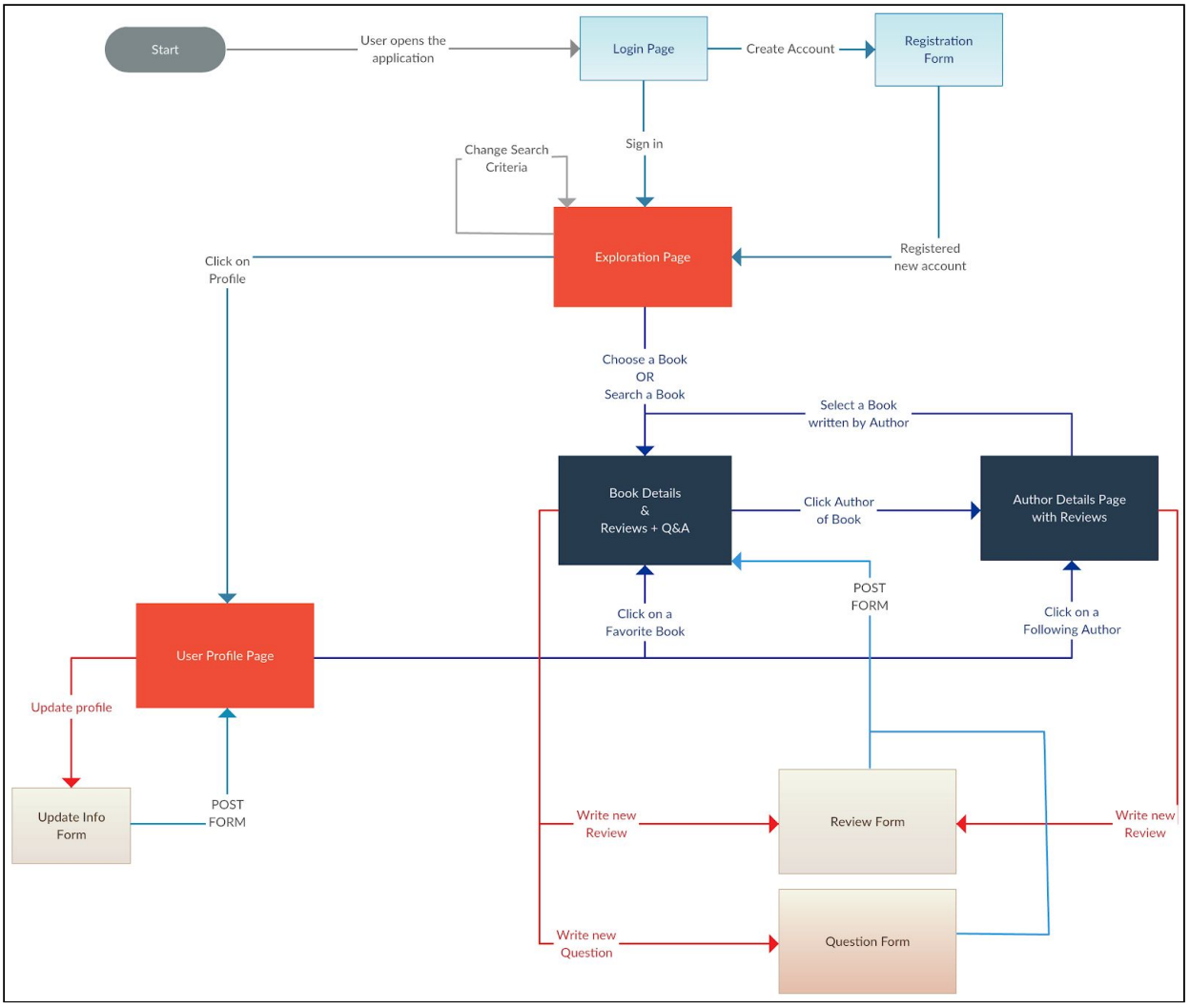
3. Sequence diagram of adding a review



4. Sequence diagram of loading a book



Activity/ State Diagram

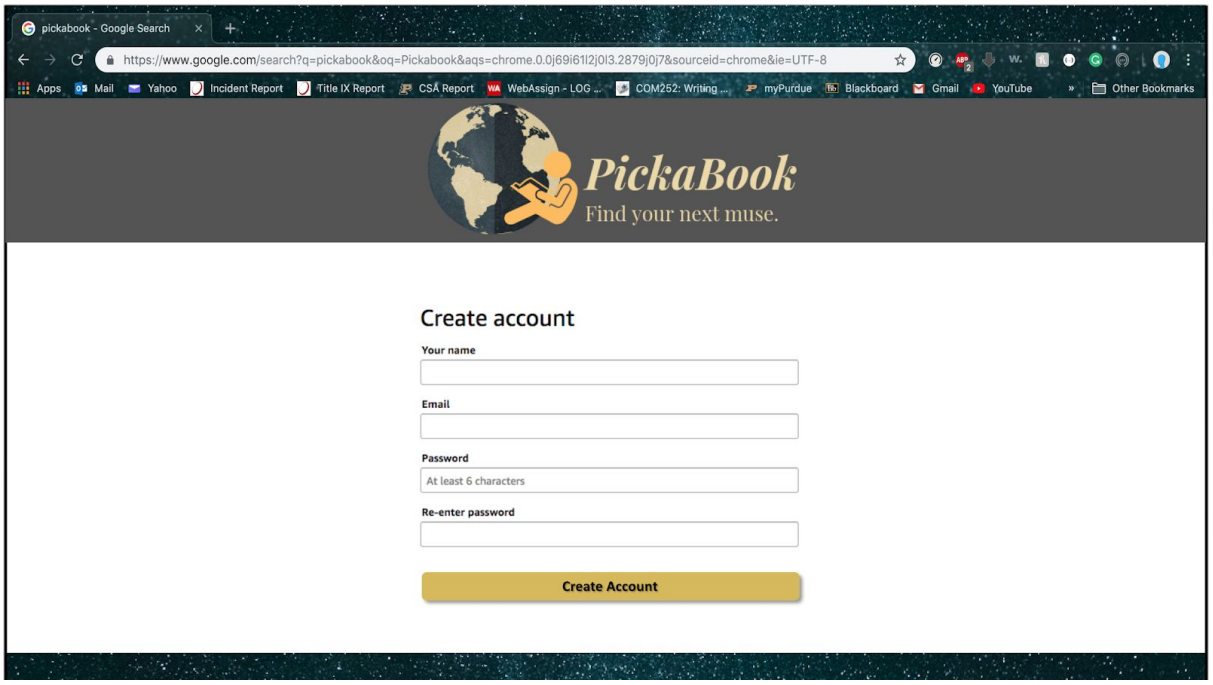


The design of our state diagram allows for a minimalistic UI design. It will enable users to go to any page from any other, thereby, providing an easy method to navigate our application. All pages will have a central theme, with a navbar header for easier navigation. The landing page of the application will be the login, enforcing anyone who would like to use the app to create an account. Each page will authenticate the user to ensure that they are logged in. Finally, there will also be a search bar to search for books, authors and other users.

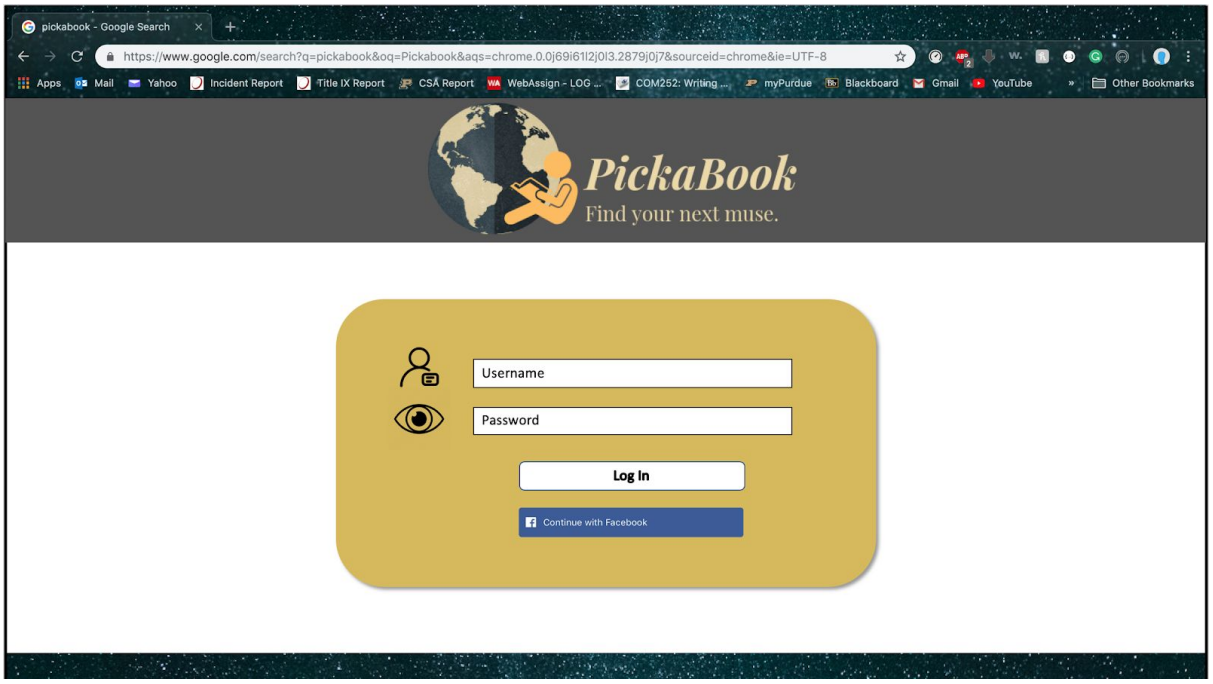
UI Mockup

The images below provide a glimpse our UI. Specifically, the 5 images cover each of the following: the Sign Up page, the Login page, the Home/Explore page, the User Profile page & the Book Info page. For each of these pages, there is our application's logo at the very top. We would also like to implement a color scheme that is consistent across the board, i.e. in all of the pages.

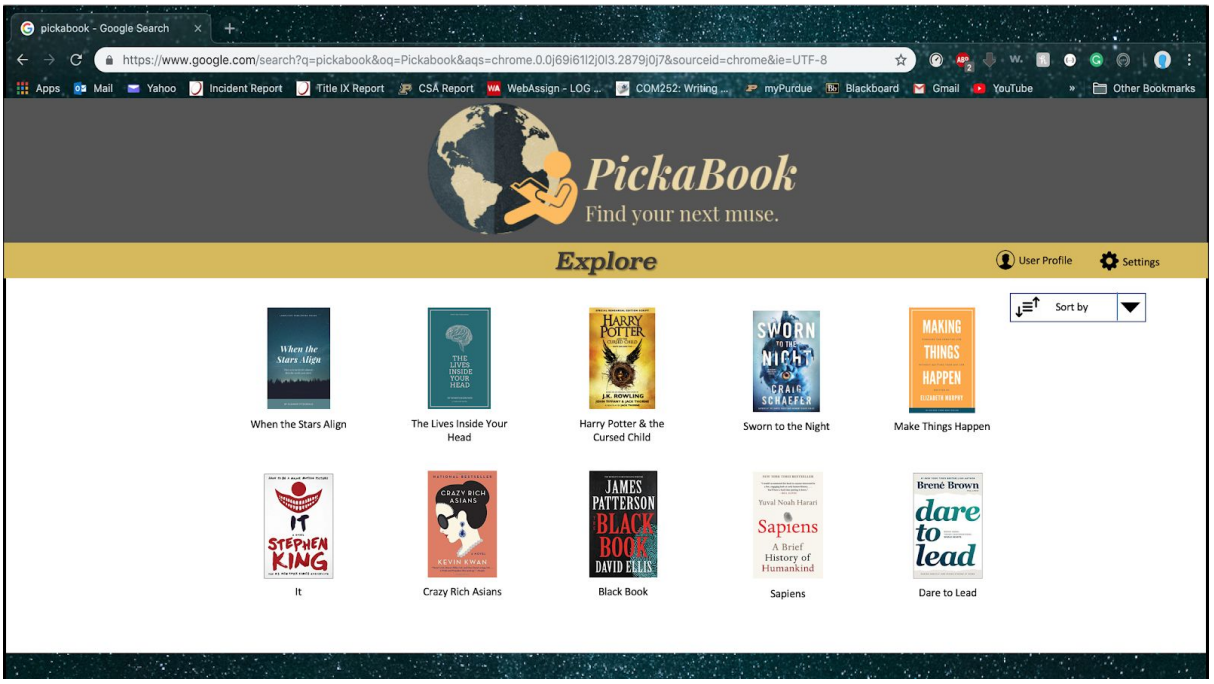
We aim to create a UI that is simplistic, minimal and well-organized. Furthermore, by organizing different functions such as creating an account, viewing your profile, exploring new books, etc. into different pages, we hope to simplify and streamline the user experience.



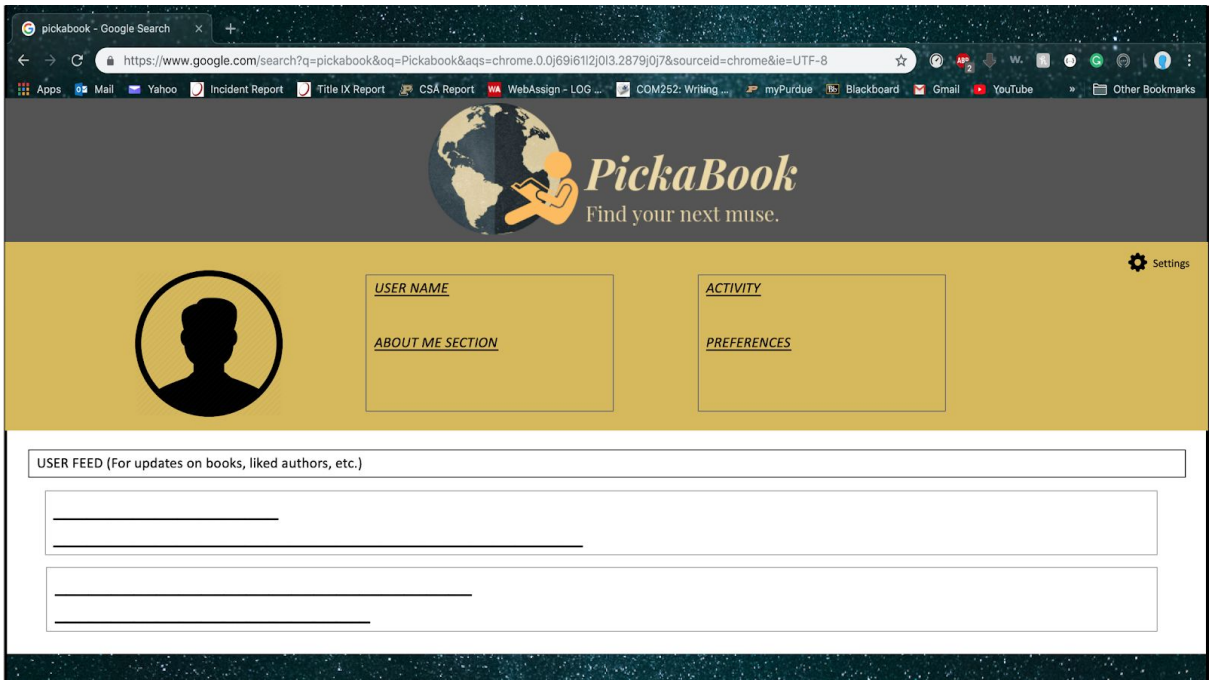
This is the sign-up page. Here, users can enter their preferred name, email address and create a password to set up an account with our application.



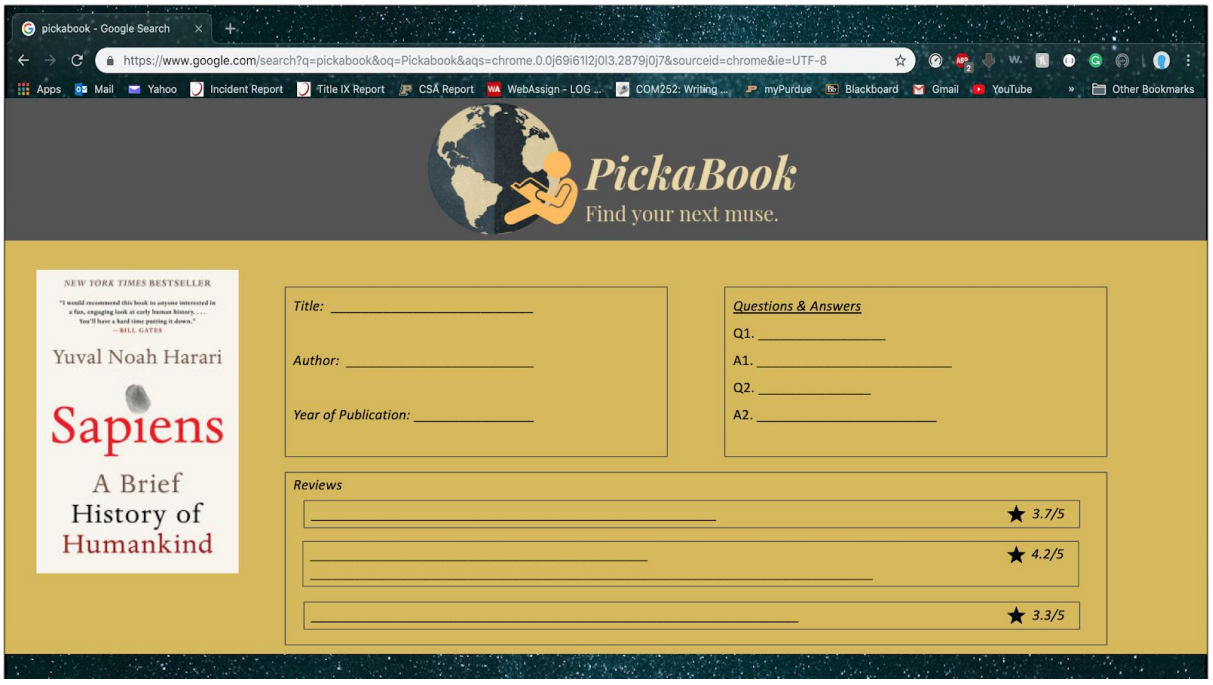
This is the login page. This page allows users to log in with an account they created previously, or link the application to their Facebook to login directly with that.



This is the Home/Explore page, which will allow users to browse through the latest books in the market. Users will also have the capability of sorting the books displayed based on different factors - name, author, popularity, etc.



This is the user profile. Here, users can view their personal information, including recent activity, preferences. They can also add an About Me section to tell other users a little bit about themselves. Below all of this information, they can their personalized user feed, comprising of updates about books they added to favorites, authors they liked, etc.



This is the Book Info page that will pop up when a user clicks on a book title. On this page, users will find an enlarged image of the cover, along with the book title, author, year of publication, etc. This page will also feature a Q&A section that will contain relevant questions about the book asked by other users, and their answers. Furthermore, right below this would be a Reviews section, where users can view the ongoing discussion about the book on the platform.