ADT Final Project Part 2
Team 24
Project Topic – Readify

Team: Ameya Dalvi, Shefali Luley, Shubham Bhagat

Part 1. Web App Architecture

Design Pattern:
We will be using MVC design pattern for our project.

- **Model:** This will contain classes/objects (entities) which will be mapped to the tables in the database.
- **View:** This will contain all the frontend pages of our application.
- **Controller:** This component will send/receive user-requests and communicate with database using models. It will also decide the format of data which will be sent to the frontend (view).

Data Storage:

- For now, we are using a MySQL database on our local system to test the application.
- When our web application will be hosted on a cloud platform, we will configure a remote MySQL database for storing the data.
- Mostly probably it will be an instance of MySQL database provided by ClearDB addon on Heroku.
- We are also planning on cache intermediate response of the API calls using Redis.

Backend:

- We are using python as the language to design and develop our backend.
- To facilitate our development, we will be using FLASK which is a microframework for using python in web application development.

Frontend:

- We are using HTML5, CSS3, JavaScript ES6 and jQuery to design the frontend pages of our web application.
- HTML and CSS will design the structure, layout, and styling of those web pages.
- JavaScript and jQuery will be used to handle DOM elements and DOM events.
- Also, we will be using Bootstrap 5 which is a CSS framework that allows us to make our web application responsive.
- We are also planning on creating a UI Prototype (Wireframe) using Figma.

Deployment:

- We will be hosting our web application on a cloud platform called Heroku. It will be a free platform to host our web application.
- It also facilitates deploying changes from remote as well as local git repository.
- This will we advantageous for us to integrate changes directly on the Heroku container.

- We will also configure an addon called as ClearDB which will provide an instance of MySQL Database for our application hosted on Heroku.

Connections:

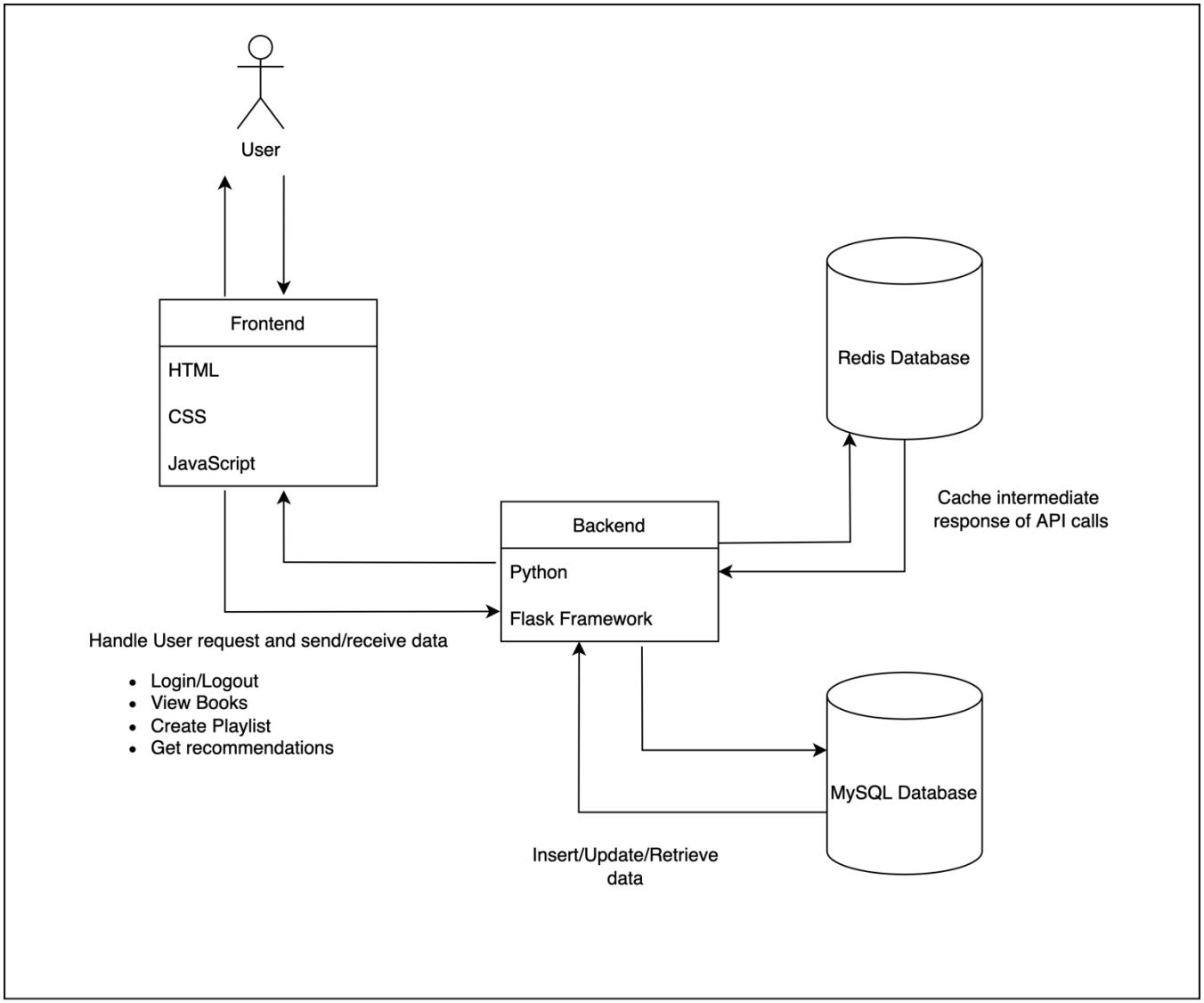
- Initially, even without logging-in the users will be able to explore different books and filter those books using different criteria's such as genre, author, rating.
- However, if a user wants to access his account dashboard, add any book to his booklist/Wishlist or create a new booklist, he will have to be logged-in to our web application.
- We will make effective use of HTTP methods – GET, POST to retrieve and store data in the database. Also, for state management in our web application we will be using 'session variables'.
- The user can send their requests using the frontend. Those requests will be received by the 'Controller' on the backend. Depending on the nature of request (GET or POST) the Controller will communicate with database using Models.
- We will also have an intermediate layer called as DAO (Data Access Object) between Controller and Database. The DAO will contain logic (functions and prepared statements) to query the database.
- Depending on the nature of request and values in query parameters if any, the controller will pass objects of Model to DAO and the DAO will use to objects to insert the data in the database or retrieve records from database.

User Interaction:

- Users will be able to interact with an intuitive UI having different components such as Navigation Bar or menu bar, Tab, Forms, buttons, modals, accordions, etc.
- The web application will also have different views such -
    - dashboard for exploring books,
    - accounts page for displaying user profile, Wishlist, book lists,
    - recommendation view to display book recommendations.
- The user can create booklists, add books to these lists also get recommendations based on these booklists.
- Furthermore, while exploring books, the user can apply filters using the filters panel which will consists of different UI components such as checkboxes, dropdown menus, range selector, etc.
- All books displayed would act as clickable links. Clicking on any book would redirect the user to a book details page which would have additional (detailed) information pertaining to that book.
- The user will also be dealing with forms such as –
    - Signup Form
    - Login Form
    - Create Booklist Form
- Additionally, we will also try to provide a graphical view providing an analysis of books using different attributes such genres of books, ratings of books, author of the book. This information will be helpful for the user to get a broader view of which books is being rating positively by different users.

Web Architecture:

- As mentioned earlier, the user will interact with the application using the frontend UI pages. Through these pages the user can send requests/data to the backend.
- The backend designed using Flask (python) will interact with the database to store, update as well as retrieve information requested by the user.

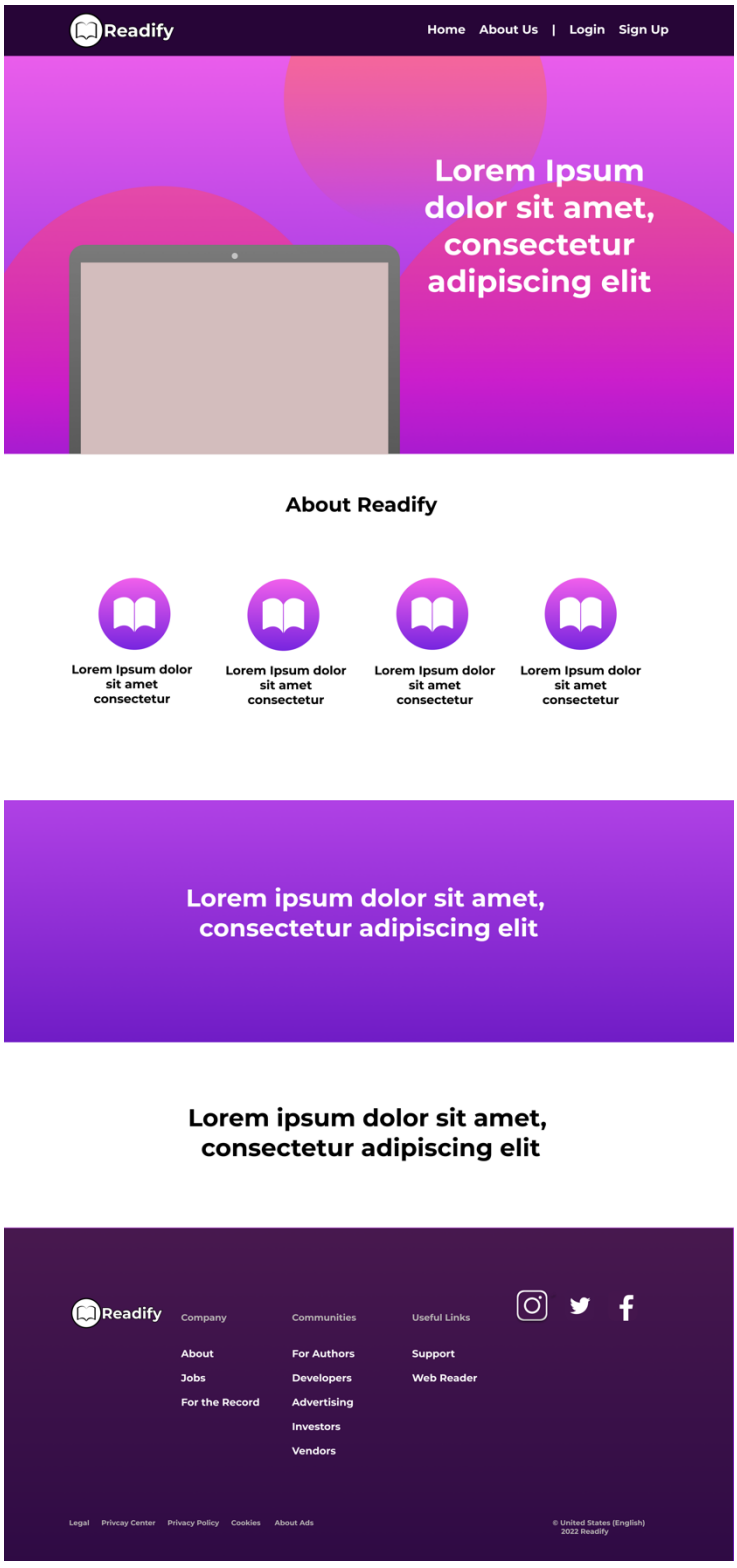


Readify Web App Architecture

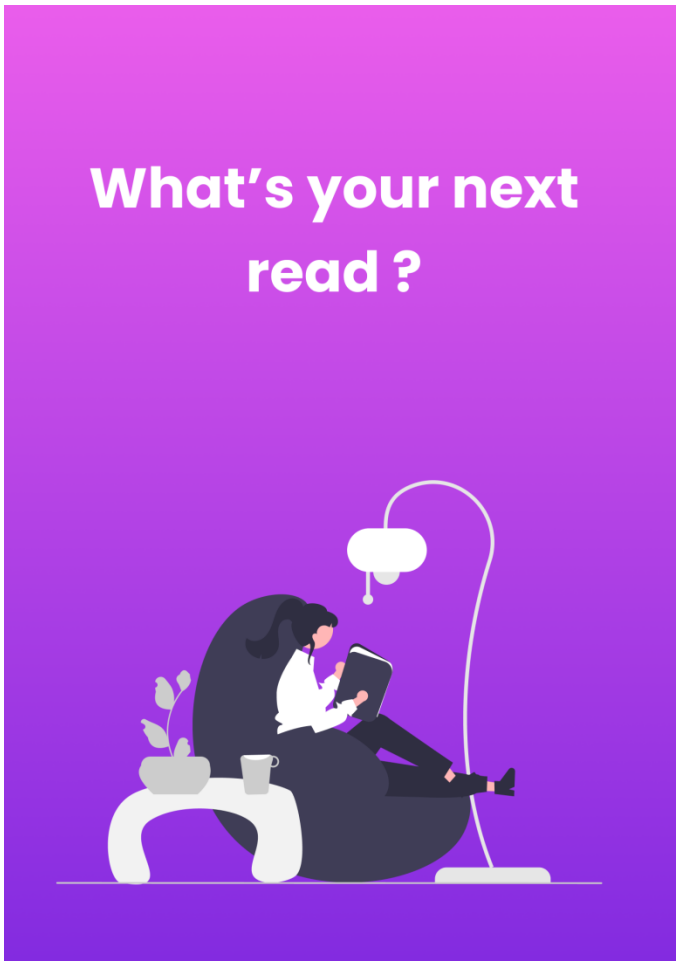
Part 2. Web App Layout

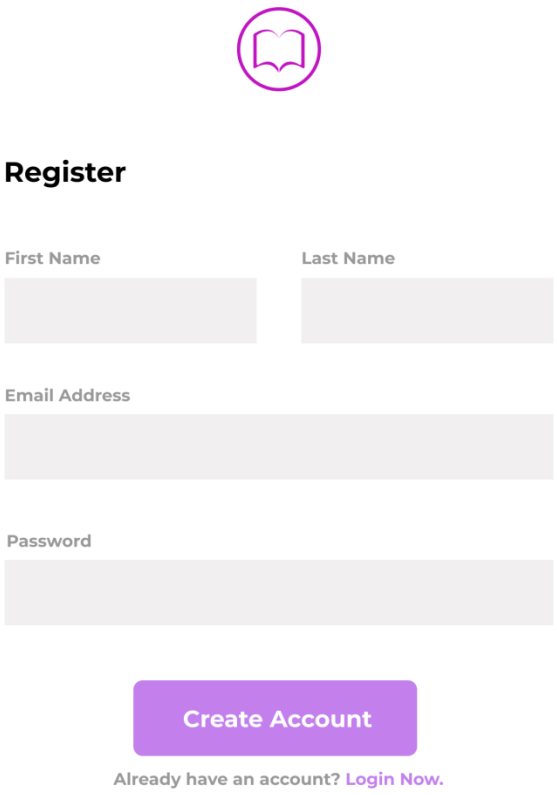
What is the initial layout (when a user sees your app first)?

When the user enters the website, the landing page is the first page that they see:



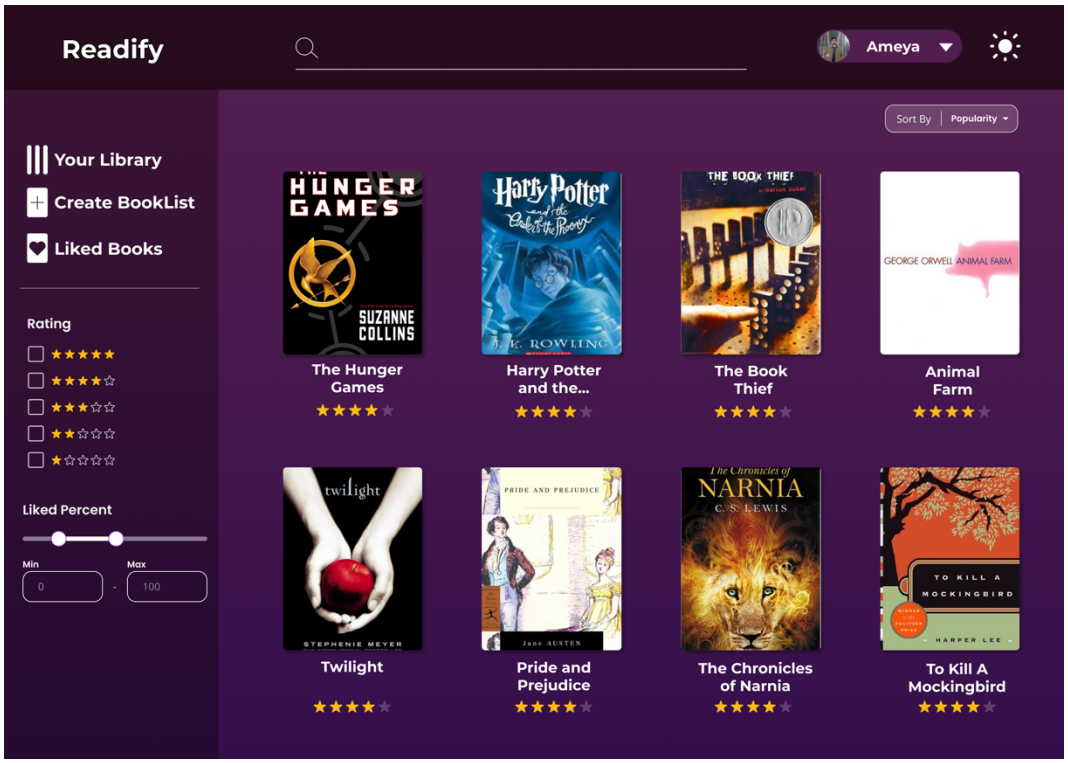
After clicking on Login or signup:





After logging into the account:

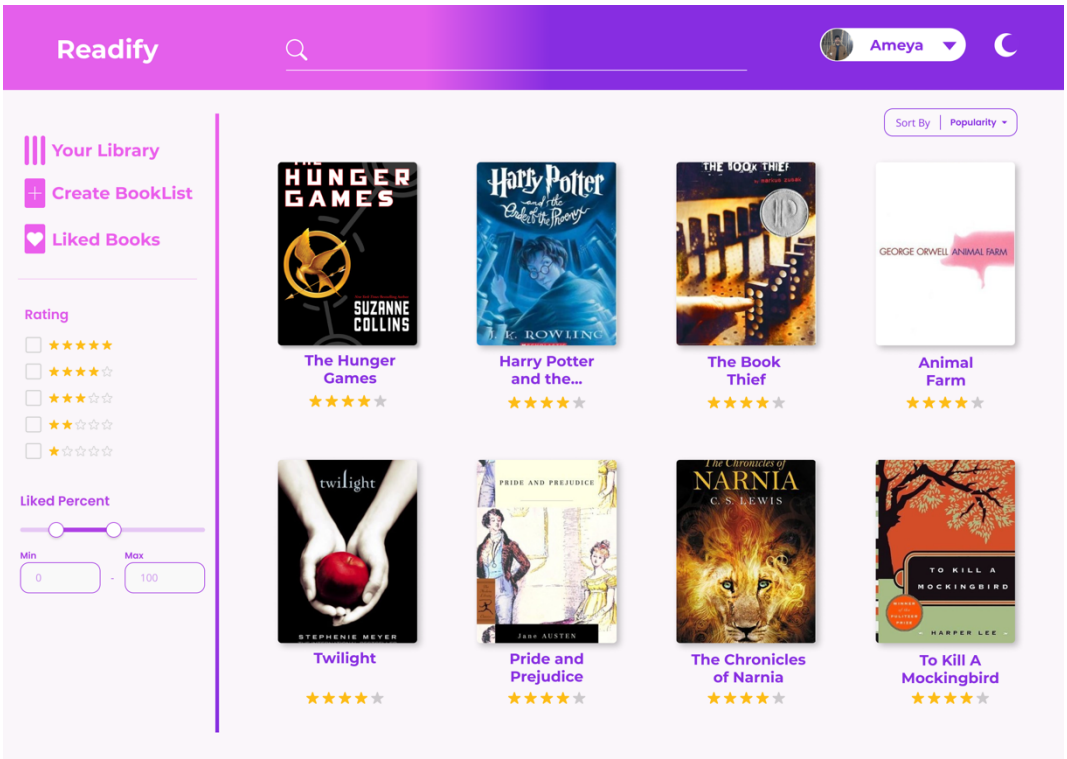
Dark Mode:



Where is the menu panel?

The menu panel is present on the left hand side at all times. User has the option to apply filters to generate list of books that interest them.

Light Mode:

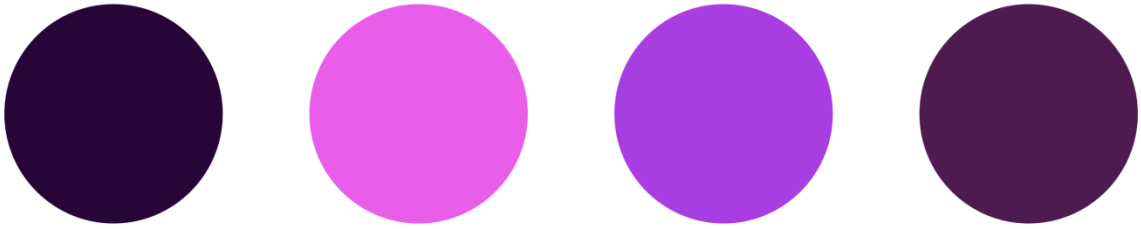


How many pages do you need? Or will you be using Tabs?

We would need 3 pages,

1. Landing Page
2. Login/Signup
3. Book Listing/ Account View

What is the color schema?
The color palette that we aim to use for our project.



#270537 #E95DEB #A83DE1 #4D1B4F

Part 3. Team Work

Ameya – Figma, Flask Backend, Data Cleaning.
Shefali – Flask Backend and Documentation, SQL Database setup.
Shubham – HTML CSS JavaScript Frontend, Website hosting.