



bibliophilia

by Saransh Saini [22F1001123]

Introduction

Bibliophilia is a role-based library management application that features a Vue.js frontend and a robust Flask backend. It is designed to provide book enthusiasts with a seamless and enjoyable experience. The application is managed by a single librarian who oversees all operations, including accepting requests, adding new books, and managing user accounts.

Technologies Used



Vue.JS



Flask



SQLite



Redis



Celery

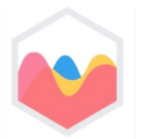
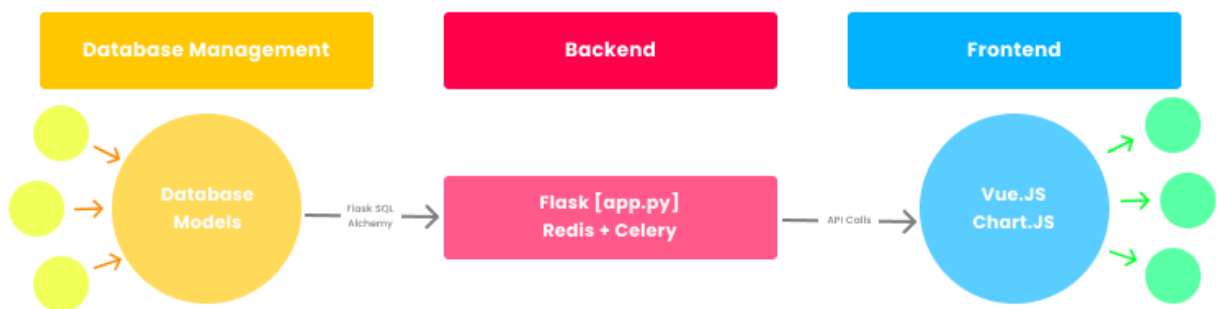


Chart.JS

System Architecture



The **app.py** file is the nexus of the application, handling crucial API endpoints that retrieve data via Flask-SQLAlchemy and transmit it to the Vue.js frontend. Additionally, it encompasses the configuration for Redis caching and batch-scheduled tasks utilizing Celery.

Database Management

The database is powered by SQLite, serving as the foundational software system. The models, crafted through **Flask-SQLAlchemy**, are meticulously designed with appropriate constraints, ensuring the steadfast integrity and consistency of the stored data. Provided below is a succinct code snippet illustrating the structure of the Books model.

```
class Books(db.Model):
    book_id = db.Column(db.String, primary_key=True)
    book_name = db.Column(db.String, nullable=False)
    img = db.Column(db.String, nullable=False)
    author_id = db.Column(db.String, nullable=False)
    author_name = db.Column(db.String, nullable=False)
    section_id = db.Column(db.String, nullable=False)
    genre = db.Column(db.String, nullable=False, default="Fiction")
    date_added = db.Column(db.DateTime, nullable=False, default = datetime.strftime(datetime.today(), "%d-%m-%Y"))
```

API Management

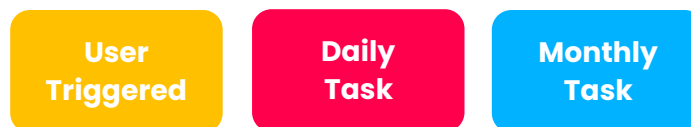
The database's CRUD operations are exclusively executed through APIs, leveraging Flask-Restful to implement essential functionalities such as **GET**, **POST**, **PUT**, and **DELETE** operations at specific endpoints. Flask-Restful facilitates the seamless integration of these operations, allowing for efficient communication between the application and the database. This approach ensures a robust and standardized mechanism for interacting with the database, promoting scalability and maintainability in the overall system.

User and Librarian Functionalities

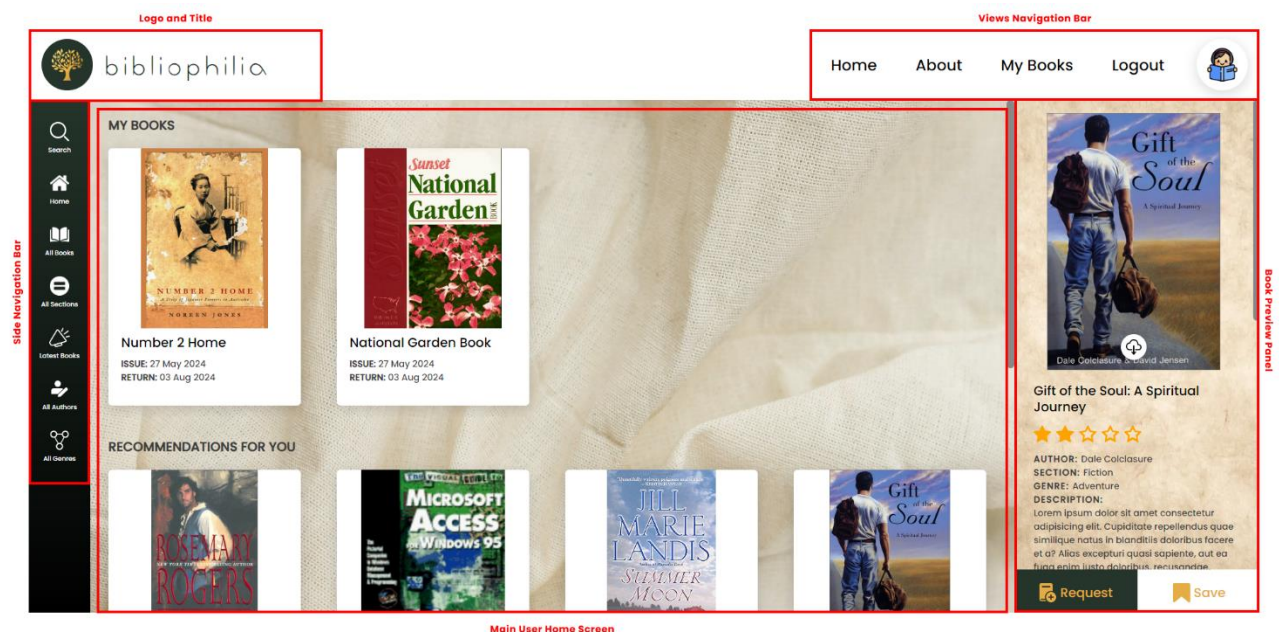


Celery Batch Jobs

Three types of Asynchronous batch jobs have been implemented

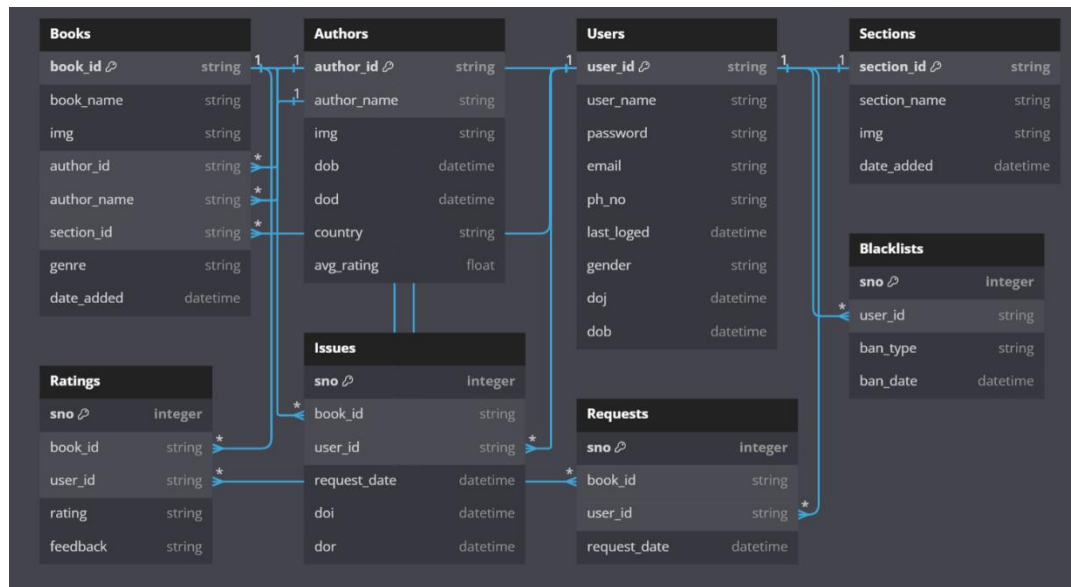


Frontend



The frontend has been meticulously designed to offer a seamless and user-friendly experience for both patrons and librarians. Styling is implemented using **basic CSS** to ensure simplicity and efficiency. Data is dynamically populated via API calls, providing real-time updates. The **View Navigation Bar** allows users to navigate between different pages, while the **Side Navigation Bar** facilitates smooth transitions between various tabs on the same page. Additionally, the **Preview Panel** enables users to view comprehensive details of the selected book and offers options to either request the book for issue or save it for later.

Database Schema



Additional Features

I have implemented a league/ranking system for users, determined by their activity level using a straightforward formula: $100 \times \text{Number of Books Issued} + 250 \times \text{Number of Books Rated}$

Users are categorized into four leagues in ascending order of rank: **Reader**, **Literati**, **Scholar**, and **Sage**. Advancement to higher leagues requires meeting specific criteria. All relevant information is displayed on the user's dashboard.



Conclusion and Extra Information

Overall, developing this application was an enjoyable and rewarding experience. Having prior experience with **ReactJS**, transitioning to **VueJS** was relatively straightforward. However, working with **Redis** and **Celery** was new territory for me, providing valuable hands-on experience and enhancing my skill set in these technologies.

- ❖ The sign-in process for the librarian involves two steps. Initially, a specific username and password are required to access the Librarian Sign-In Page.
- ❖ **How to start the app** (in the terminal execute these commands):
 - o python app.py [In backend directory]
 - o yarn run serve [In frontend director]
 - o celery -A app.celery_app worker --loglevel INFO -P gevent [In backend directory]
 - o celery -A app.celery_app beat --loglevel INFO [In backend directory]

- ❖ **Presentation Video Link:**

<https://drive.google.com/file/d/1b1nlhAl2gbPuQvSye2skowuVp0q8gY5r/view?usp=sharing>