Author

Ranjeet Sharma
21f2001119
21f2001119@ds.study.iitm.ac.in
Student at IIT-Madras pursuing Bachelor's degree in Data Science and Application, with a fervent interest in development, and programming.

Description

This project is about an Online Library Management System. There will be one admin and many users. Users can sign up/register and can start reading books or issue e-books. Admin can perform CRUD Operations on Sections and Books and handle incoming book requests.

Technologies used

- **Flask**: For application code, to handle user requests, manage routing, and creating APIs.
- **Vue.js**: For building a dynamic and responsive user interface.
- **Flask-SQLAlchemy**: For interaction with the database.
- **Flask-Bcrypt**: For hashing passwords.
- **Bootstrap**: For quick CSS styling and aesthetics.
- **SQLite**: For data storage.
- **Redis**: For caching.
- **Redis and Celery**: For batch jobs and task queue management.

DB Schema Design

- **Book Table:** Stores details of the book, having columns as `isbn`, `name`, `content`, `author_name`, `section_id`, `date_added`, `language`, and `rating`. Primary Key being `id` that stores a unique id for each book. It also has a column name `enrollments` that is in a relationship with the enrollments table.
- **User Table**: Stores details of each user, having columns as `id`, `name`, `email`, and `password`. It also has a column name `enrollments` that is in a relationship with the enrollments table, making a many-to-many relationship.
- **Enrollments Table**: `book_id` and `user_id` columns are Foreign Keys to Book and User tables respectively, also these two making a unique constraint such that no duplicate records are there. `issue_date` and `return_date` columns keep track of book issuing and returning dates. The table creates a many-to-many relationship with User and Book tables.
- **Sections Table**: Represents different sections in the library. Each section has a unique `id`, `name`, `creation_date`, and `description`.
- **Book_req Table: Tracks requests made by users to issue books. Includes `user_name` and `id`, `book_name` and `id`, `req_days`, `issue_date`, and `return_date`.**

➢ Rest Tables namely ***Feedback, Rating, Status,*** stores feedbacks, ratings for each book received. Status to keep track of every user's completed Book.

Architecture and Features

app.py: Contains the main code to run the Flask application. It initializes Flask and Flask-SQLAlchemy objects and contains necessary imports from controllers.

Controllers: Contains all the routing for the project.

models.py: Contains the schema for database design using Flask-SQLAlchemy. It includes classes that represent tables in the database, including columns and relationships between tables.

Static and Templates: The static folder contains global.css along with a few images, and all the HTML files are kept in the templates folder.

- **CRUD** Operations for **Sections** and **Books**:
  - ➔ `section.py` and `admin.py` contain routes specifically for admin only. Only logged-in admins can perform CRUD operations on sections and books.
- For **Securing** ***admin-specific*** routes:
  - ➔ The `@app.before_request` decorator is used to validate each request before executing and ensuring that only logged-in admins can access those routes.
- **Admin** Features:
  - ➔ Routes to view all books, sections, book requests, and all registered users.
  - ➔ Dashboard route to access analytics and insights.
  - ➔ Routes to accept/deny book requests and revoke book access for users.
- **User** Features:
  - ➔ Routes to request books, download, view issued books, return, and manage currently issued books and user profiles.
  - ➔ Search functionality for both admin and regular users to search based on book name, author, and sections.
  - ➔ Bar Charts on the admin's dashboard for better tracking of books and sections.
- **Search Functionality**
  - ➔ Both admin and regular users can search based on book name, author, sections.
- **Bar Charts** also visible on the admin's dashboard for better track of books and sections.
- **Batch Jobs and Task Management**
  - ➔ **Redis and Celery**: Used for managing background tasks and scheduling periodic jobs like sending reminders and generating reports.

API

➔ **Book Management API**, **Section Management API, Enrollments Management API, Profile Management API, Search, User Authentication/Management API** and **Admin** relevant endpoints with **GET**, **POST**, **DELETE**, **PUT** methods.

Video

https://drive.google.com/file/d/1ssPTRdVz-XE7gYkFoi3Z3nJXKW55J46G/view?usp=sharing