MAD II - Library Management System Project Report

Author

Krishna Chandra Shekar
21f1006336
21f1006336@ds.study.iitm.ac.in
I am a working professional currently in my final semester of Diploma.

Description:

The LMS (Library Management System) project is designed to facilitate the management of library resources, including ebooks and user requests. The system supports user and librarian roles, enabling functionalities like user registration, login, ebook management, and feedback handling.

Technologies Used:

- **Flask:** A lightweight WSGI web application framework used to build the backend APIs. It is used for its simplicity and ability to easily integrate with various libraries for building a RESTful API. (Flask-Caching, Flask-Cors, Flask-JWT-Extended)
- **SQLite:** It is a simple yet effective Database used for storing data. It is chosen due to its lightweight nature and ease of integration with Flask
- **VueJS:** A progressive JavaScript framework used to build the frontend UI, allowing for a reactive and dynamic user interface. It is used for its ability to build a modern and responsive frontend.(Pinia, Axios)
- **Redis:** Used for caching to improve performance by reducing the load on the SQLite database, especially for frequently accessed data through API.
- **Bootstrap:** Used for HTML generation and styling to ensure a responsive and user-friendly design. It ensures the frontend design is responsive and visually appealing

DB Schema Design:

- **Users Table:** Stores user credentials and roles.
  Columns: id (Primary Key), username, password, role

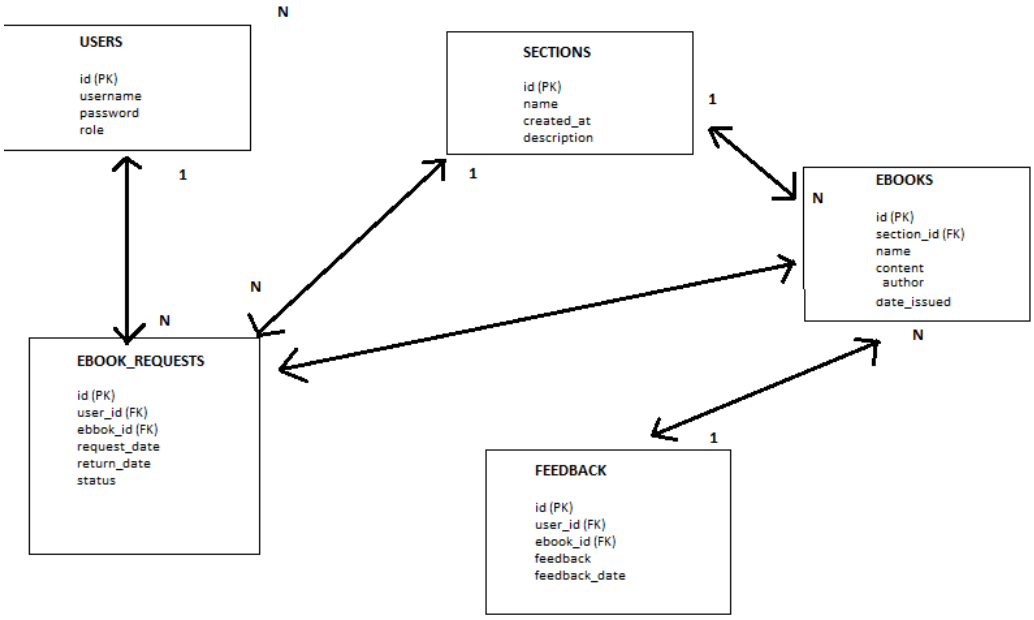
- **Sections Table:** Manages e-book sections.
  Columns: id (Primary Key), name, created_at (Default: current timestamp), description

- **Ebooks Table:** Stores e-book details.
  Columns: id (Primary Key), section_id (Foreign Key), name, content, author, date_issued

- **Ebook Requests Table:** Manages user e-book requests.
  Columns: id (Primary Key), user_id (Foreign Key), ebook_id (Foreign Key), request_date, return_date, status

- **Feedback Table:** Stores user feedback on e-books.
  Columns: id (Primary Key), user_id (Foreign Key), ebook_id (Foreign Key), feedback, feedback_date

ER Diagram:



API Design

The APIs are designed to handle various functionalities such as authentication, e-book management, section management, and feedback management. They are:

- **auth.ts:** Handles user authentication, including login and registration.
- **ebook.ts:** Manages e-book creation, editing, deletion, and retrieval.
- **ebook_requests.ts:** Handles user requests for e-books, including requesting, returning, and viewing the status of requests.
- **feedback.ts:** Manages the submission and retrieval of feedback for e-books.
- **section.ts:** Handles section creation, editing, and deletion.
- **stats.ts**: Handles the statistics of the user/librarian details in charts and plots.

Implementation:

Using extensions like Flask-JWT-Extended for authentication, Flask-Caching with Redis for performance, and SQLlite for database interactions, the backend is constructed using Flask. The frontend uses Vue.js with Axios for API interaction, Vue Router for navigation, and Pinia for state management. Real-time notifications, role-based access control, and a modular architecture that guarantees scalability and maintainability present in this project. With seperate frontend and backend connections, the system is built to manage user and librarian interactions, such as authentication, ebook management, and feedback, effectively.

Architecture and Features

Architecture:
The project consists of a Flask backend and a Vue.js frontend, which are linked through RESTful APIs. The backend is responsible for managing data and implementing business logic, and it includes models for users, sections, e-books, requests, and feedback. Data operations are managed by Flask controllers, while the frontend utilizes Vue.js for creating interactive user interfaces and Vue Router for navigation. State management is handled using Pinia, and API requests are managed by Axios. Styling is provided by Bootstrap, and user experience is enhanced with extra libraries such as Notivue and Oh-Vue Icons for notifications and icons. This approach ensures that the application is modular, maintainable, and scalable.

Features Implemented:

- **User Authentication:** Role-based access control differentiating between librarian and user.
- **Librarian Dashboard:** Displays statistics such as active users, e-book requests, issued and returned e-books.
- **Section Management:** Librarians can create, update, and delete sections.
- **E-book Management:** Librarians can manage e-books within sections, including adding, editing, and deleting e-books.
- **Search Functionality:** Users can search for e-books and sections by title, author, or genre.
- **Request Management:** Users can request and return e-books, with manual revocation if the return is delayed.
- **Feedback System:** Users can provide feedback on e-books, which is stored in the system for librarian review.
- **Caching with Redis:** Frequently accessed data is cached to reduce database load and improve performance.
- **RBAC:** Implemented using JWT to ensure secure access control.
- **Statistics :** Generate statistics for either a librarian or a user, using data from the database while ensuring the user is authenticated and authorized via JWT

Video:

LMS Video Demonstration