

PROJECT REPORT

Author

Abhishek

22f3000978@ds.study.iitm.ac.in

Description

This project involves developing a web application for a library management system. Key features include user authentication, allowing users to register and login, with role-based access control for librarians. Librarians can manage books, issue and return books, view statistics, and generate charts for the dashboard. The application also handles file uploads, such as book covers and documents

Technologies used

1. Flask: Lightweight web framework for Python.
2. SQLAlchemy: ORM library for database interactions.
3. Flask-Login: User session management for Flask.
4. Flask-CORS: Cross-Origin Resource Sharing for Flask.
5. Vue.js: JavaScript framework for frontend.
6. Vuex: State management for Vue.js.
7. Redis: In-memory data structure store, used for caching.
8. Celery: Distributed task queue, used for scheduled jobs.
9. Flask-Caching: Flask extension for caching with Redis.

DB Schema Design

1. User: Stores user information such as name, username, email, password hash, role, and image.
2. Book: Contains details about books including title, description, author, file path, book cover image, and a foreign key reference to the section it belongs to.
3. BookIssuance: Records instances of book loans, including the user who borrowed the book, the book itself, the date it was issued, the return date, whether it has been read, and if it has been returned.
4. Section: Contains information about book sections, such as name and description.
5. Section-Book relationship: A many-to-many relationship table connecting sections and books.


API Design

APIs are created using Flask, SQLAlchemy, and Flask-Login. They cover user authentication (signup, login, logout), book management (issue, return, request), and librarian dashboard. Routes are organized within Flask Blueprints for modularization. Authentication APIs handle form data, while book management APIs manage book operations. Librarian dashboard API retrieves statistics and generates charts using Matplotlib. Feedback submission API enables users to provide feedback on books. Each route supports specific HTTP methods with error handling for robustness.

Architecture and Features

The project is structured around Flask for the backend and Vue.js for the frontend. Flask follows a conventional MVC architecture with routes handling requests, models interacting with the database, and templates rendering dynamic content. Vue.js is integrated into the frontend to create interactive user interfaces and handle client-side routing. Controllers are defined in the routes directory, templates are stored in the templates directory, and static assets are kept in the static directory. Configuration settings are centralized in config.py for easy management. Default features like user authentication and book management are implemented using Flask-Login and Flask-SQLAlchemy, while additional features such as user feedback and librarian dashboards for book management are also included. Vue.js enhances the frontend with reactive components and client-side routing, providing a seamless user experience. The combination of Flask and Vue.js enables efficient development of a modern web application with both server-side and client-side functionality.

Video

 Appdev2.mkv

<https://drive.google.com/file/d/1NdghqpngSkMTCufkw9IAmLbHAEayHSmg/view?usp=sharing>