Library Management System V2.0

Author

Mohak Khatri
21f1000865
21f1000865@ds.study.iitm.ac.in

An aspiring software engineer.

Description

Library Management System V2.0 is a multi-user application to issue e-books. Users can read, request and return e-books. A librarian can perform CRUD operations on sections (container for books) and books. The librarian can also approve or deny requests made by the users. The librarian also has the ability to revoke an already issued book.

Features

- The application allows for 2 kinds of users, a general user and a librarian.
- Separate login for both kinds of users has been provided.
- General users are also provided with a “register” functionality to register as a new user.
- Librarian features: CRUD on sections and e-books. Approve/deny incoming user requests, revoke e-book access, view overall summary.
- General user features: Request for e-books, view/return/provide feedback for issued e-books, view all requests sent, view all books issued to them, search for e-books and sections based on various parameters.
- Backend features: Caching for performance, RBAC, daily visit reminder emails, monthly activity report emails and Auto-revoke functionality.

Technologies Used

- Flask for APIs
- Vue.js (CLI) for Frontend
- Bootstrap for UI
- SQLite for primary database
- Redis as a cache database, message broker and task-queue result store
- Celery for asynchronous backend jobs
- Mailhog as a sudo SMTP server
- JWT based authentication

DB Schema Design

Schema Link

Relationships:

- User-Request: One to many
- Book-Request: One to many
- Section-Book: One to many

API Design

All API endpoints are developed using Flask. Each API follows required JWT based authentication, returns valid responses along with valid http codes (even for errors), restricts the calls to pre-defined request methods and ensures consistency throughout the application.

Frontend/UI Design

Entire frontend is developed using Vue.js. Bootstrap is used for styling. Asynchronous fetch methods are used to make calls to the backend APIs. Vue-CLI is used to instantiate vue-webpack. Vue-router is used for handling routing.

Backend Features

- Caching using redis for cache store and flask-caching module
- Scheduled asynchronous jobs: Daily visit reminder emails, Monthly activity report emails, Auto-revoke functionality
- APIs for interacting with the data

Architecture

The root directory contains 2 folders and a readme file. The contents of the folder named “backend” are as follows:

- .py extension files which contain the application code
- A folder named “templates” which contains the html templates that can be used without a vue frontend, directly with the flask application
- A folder named “static” which contains the static files required by these templates
- A folder named “database_files” containing the SQLite database.

The folder named “frontend” contains all the files required by vue-webpack to function correctly. Custom defined vue-components are in “frontend/src/components”. Routing information can be found in “frontend/src/main.js”.

Video

LMS_V2.0_Presentation.mp4