MAD-2 Project Report

LMS V2

Author

Name: Ashutosh Utsav
Email: 22f2001659@ds.study.iitm.ac.in
ID: 22f2001659
About me: I'm currently pursuing a BS in Data Science and Applications and I am currently at Diploma level. I have gained a lot of Programming experience while doing my project.

Description:

Developed a multi-user web application for Online Library management. Implemented functionalities based on VueJS(CLI) for frontend and Flask for API creation. The features also include scheduled jobs and reminders using redis and celery.

Technologies Used:

VueJS(CLI): Frontend of the application.
Flask: Backend/API development.
Flask-Restful: To develop the Restful API for the app.
Flask-Security: For Token based authentication.
Sqlite3 Flask-SQLAlchemy: Simplified database interaction with ORM.
Redis and Celery are used for scheduled jobs/daily reminders via Google Chat and Mail
Flask-mail: For sending monthly report emails.
Flask-CORS: To enable CORS for the app.
Jinja2: For generating monthly activity reports at backend.
SQLite: Easy-to-use SQL database engine for data storage.

Database Schema Design:

The database is created using SQLite and Flask-SQLALchemy.
7 Tables used in the database: `User`, `Role`, `RolesUsers`, `Section`, `Books`, `BookRequest`, `UserLibrary`.

- **Roles of Users** are stored in the `RolesUsers` table.
- **UserLibrary** and **Books** have a many-to-one relationship.
- **BookRequest** tracks requests by users for books, linking to both `User` and `Books`.

Constrains:

- **Primary keys** ensure unique identification of records in each table.
- **Foreign keys** establish relationships between tables, ensuring data integrity.

Reasoning:
The structure efficiently manages the library data with relationships between users, roles, sections, and books. The use of foreign keys helps maintain referential integrity, ensuring books, sections, and requests are linked appropriately to users and their roles.

API Design

CRUD on Books/Section:
Description: Books/Section API allows performing CRUD operations on Section/Books. It supports HTTP methods like GET, POST, PUT, and DELETE to manage Section/Books data in the database.

- API for User Login/Signup
- API for User Library where user can view the book and return the book
- API for Admin to manage the user's of the app which allow admin to GET and DELETE the user and Books in user's Library

These API were implemented using Flask to define resources.SQLAlchemy was used to interact with the database and the data is returned as JSON responses by Flask's jsonify.

Architecture and Features

The application follows the MVC structure:
Model(M) is handled by flask - It interacts with databases and manages data.
View(V) is handled by vue.js. Vue components are responsible for interactive user interface.
Controller(C) is handled by flask. Flask Resources handle all the business logic at the Backend.

Project Features Implemented:

●Role based Login for Librarian and User's
●Admin has special privilege to see the app statistics and manage users, Books and Section.
●Users can search Books/Section.
●Users can Read Books, Explore a particular Section and request any Books.
●Admin can create Section, Upload Books, Edit Books/Section and delete them.
●Monthly Activity report of Admin is sent to Admin email.
●Daily notifications on google chat to users to visit the app if inactive for 24 hours.

Video

For viewing the video : Video

To run the app please Read Read.md in Root Folder.