MAD-2 Library Management System Project Report

Author

Name: Tanmay Sharma
Roll Number: 21F3001409
Email: 21f3001409@ds.study.iitm.ac.in

Description

The Library Management System (LMS) is a multi-user web application that allows librarians to manage sections and e-books and general users to request, and access e-books. The system includes features such as role-based access control, e-book management, section management, and asynchronous tasks for daily reminders and monthly reports.

Technologies Used

Flask: Used for developing the backend REST API, providing endpoints for user authentication, section, and e-book management.
SQLite: Employed as the relational database to store user, section, and e-book information.
Flask-SQLAlchemy: Used as the ORM for database operations.
Flask-Security: Integrated for role-based access control, ensuring that only users with specific roles (e.g., librarian) can perform certain actions.
Flask-JWT: Implemented for managing authentication and protecting API endpoints.
Flask-Caching: Used for caching API responses to improve performance.
Celery with Redis: Employed for handling asynchronous tasks such as sending daily reminders and generating monthly activity reports.
Flask-Mail: Utilized for sending email notifications to users.
Flask-Migrate: Used for managing database migrations.

DB Schema Design

Tables:

User: Contains user information including roles, with fields like id, username, email, password, active, roles, max_books, etc.
Role: Stores role data such as id, role_name, and description.
Section: Represents Books sections with fields such as id, name, date_created, and description.
EBook: Contains information about e-books, including id, title, content, authors, isbn, rating, section_id.
BookRequest: Tracks requests made by users for e-books, with fields like id, user_id, ebook_id, date_requested, is_granted, date_granted, and date_returned.
Feedback: Stores feedback and ratings provided by users on e-books, with fields like id, user_id, ebook_id, rating, and date_created.

ER Diagram:

User has a many-to-many relationship with Role through roles_users.

Section has a one-to-many relationship with EBook.

User has a one-to-many relationship with BookRequest and Feedback.

EBook has a one-to-many relationship with BookRequest and Feedback.

API Design

Librarian Endpoints:

POST /librarian/login: Login for librarian role, returns JWT token.

GET /librarian/dashboard: Returns statistics like total users, requests, e-books, and sections.

POST /sections: Create a new section in the library.

PUT /sections/<int:section_id>: Update an existing section.

DELETE /sections/<int:section_id>: Delete a section.

GET /sections/<int:section_id>/ebooks: Get all e-books in a section.

POST /librarian/upload: Upload and add a new e-book.

PUT /librarian/ebooks/<int:ebook_id>: Update e-book details.

DELETE /librarian/ebooks/<int:ebook_id>: Delete an e-book.

GET /requests: View all book requests.

POST /requests/<int:request_id>/grant: Grant a book request.

POST /requests/<int:request_id>/revoke: Revoke a granted book request.

POST /export/csv: Export book requests as a CSV file.

User Endpoints:

POST /register: Register a new user.

POST /login: Login for general users, returns JWT token.

GET /sections_with_books: Get all sections with their respective e-books.

GET /book/<int:book_id>: Get detailed information of an e-book.

POST /book/rate: Submit a rating for an e-book.

POST /book/request: Request to borrow an e-book.

POST /book/return/<int:book_id>: Return a borrowed e-book.

GET /book/request/limit: Check if a user can request more books.

GET /book/request/status/<int:book_id>: Check the request status of an e-book.

GET /search: Search for sections and e-books by keywords.

Video

https://drive.google.com/drive/folders/1MyulTBzeO3ReL8g-y-ulSNyldajM6H0-?usp=sharing