Library Management System Project Report

Pranshu Jaiswal
21f3001310
MAD2 Project

Problem Statement:
Develop a multi-user Library Management System (LMS) for managing e-books across various sections, similar to an online library. The system should support two user roles: librarian and general user. The librarian maintains sections and e-books, while general users can access e-books from the library.

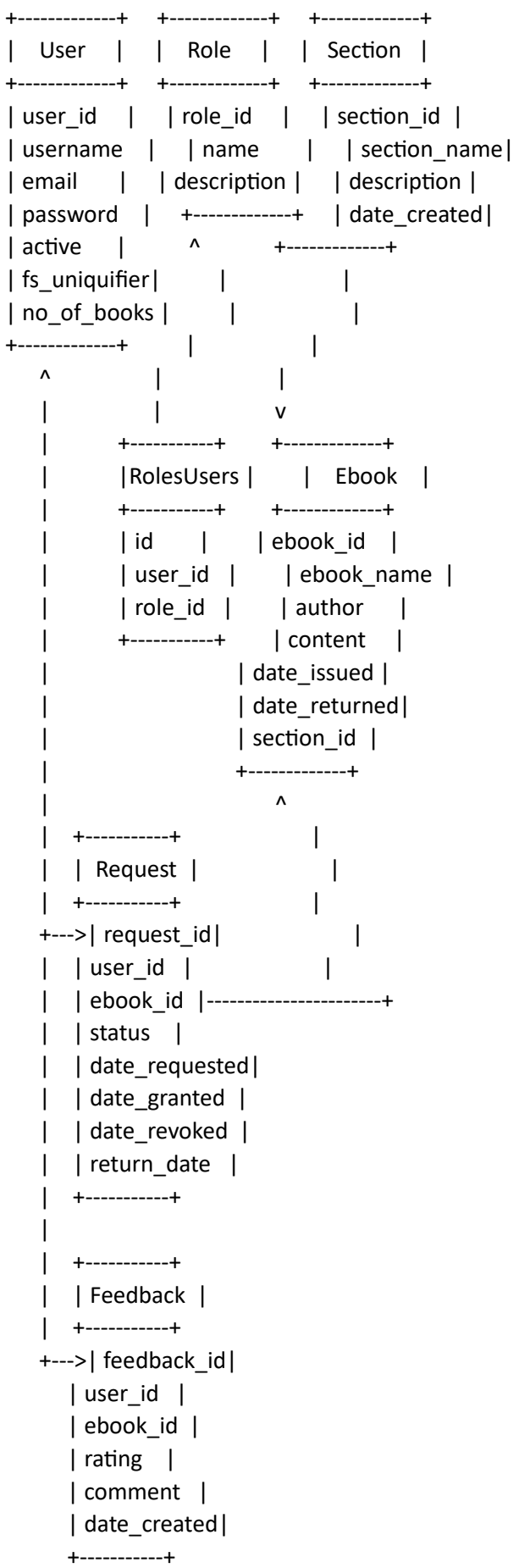
Approach:

1. Backend Development:
    a. Implemented a Flask-based RESTful API to handle user authentication, e-book management, and request processing.
    b. Utilized SQLAlchemy ORM for database operations and model definitions.
    c. Implemented JWT-based authentication for secure API access.
2. Frontend Development:
    a. Created a Vue.js-based single-page application (SPA) with separate dashboards for librarians and general users.
    b. Implemented responsive UI components for managing sections, e-books, and user requests.
3. Database Design:
    a. Designed a relational database schema to store user information, e-books, sections, requests, and feedback.
4. API Documentation:
    a. Created a Swagger/OpenAPI specification to document all API endpoints and their usage.
5. Asynchronous Tasks:
    a. Implemented Celery for handling background tasks such as sending daily reminders and monthly reports.

Frameworks and Libraries Used:

1. Flask
2. SQLAlchemy
3. Flask-JWT-Extended
4. Celery
5. Vue.js
6. Fetch API
7. Bootstrap
8. SQLite: Lightweight relational database
9. Swagger/OpenAPI: API documentation

ER Diagram:


User
user_id
username
email
password
active
fs_uniquifier
no_of_books
Role
role_id
name
description
Section
section_id
section_name
description
date_created
RolesUsers
id
user_id
role_id
Ebook
ebook_id
ebook_name
author
content
date_issued
date_returned
section_id
Request
request_id
user_id
ebook_id
status
date_requested
date_granted
date_revoked
return_date
Feedback
feedback_id
user_id
ebook_id
rating
comment
date_created

API Resource Endpoints:

1. Authentication:
    a. POST /api/login: User login
    b. POST /api/register: User registration
2. Users:
    c. GET /api/users: Get all users (Librarian only)
    d. GET /api/user/profile: Get user profile
    e. PUT /api/user/profile: Update user profile
    f. GET /api/user/stats: Get user statistics
3. Sections:
    g. GET /api/section: Get all sections
    h. POST /api/section: Create a new section (Librarian only)
    i. PUT /api/section/{section_id}: Update a section (Librarian only)
    j. DELETE /api/section/{section_id}: Delete a section (Librarian only)
4. Ebooks:
    k. GET /api/ebook: Get all ebooks
    l. POST /api/ebook: Create a new ebook (Librarian only)
    m. PUT /api/ebook/{ebook_id}: Update an ebook (Librarian only)
    n. DELETE /api/ebook/{ebook_id}: Delete an ebook (Librarian only)
5. Requests:
    o. GET /api/request: Get all requests
    p. POST /api/request: Create a new request
    q. PUT /api/request/{request_id}: Update a request status (Librarian only)
6. Returns:
    r. POST /api/return/{request_id}: Return an ebook
    s. POST /api/auto-return: Automatically return overdue ebooks
7. Feedback:
    t. GET /api/feedback: Get all feedback (Librarian only)
    u. POST /api/feedback: Submit feedback for an ebook
    v. DELETE /api/feedback/{feedback_id}: Delete feedback (Librarian only)
8. Librarian Dashboard:
    w. GET /api/librarian/dashboard: Get librarian dashboard statistics