Library Management System
(AppDev2 - Project)

Name: Daksh Sharma
Roll No.: 21f1004415
Student Email: 21f1004415@ds.study.iitm.ac.in

Project Overview:
This application is a multi-user platform designed for book borrowing and reading. The development process followed the wireframe and guidelines provided by the AppDev2 instructors, and involved the following key steps:

- **Database Design:** The database schema and tables were created using Flask-SQLAlchemy.
- **App Development:** A Flask app instance was set up along with HTML pages, styled using CSS and Bootstrap.
- **Routing:** All necessary routes were implemented to connect the app with the database, including a secure login system that stores hashed passwords.
- **Styling:** CSS was applied to enhance the visual appeal of the web pages.
- **Scheduled Tasks:** Celery jobs were integrated to handle tasks.

Technologies and Frameworks Utilized:

- **Vue.js:** The frontend of the application was developed using Vue.js.
- **Flask:** The backend was built with Flask.
- **Redis & Celery:** These were employed for scheduled jobs and sending daily reminders via Google Chat and MailHog.
- **Flask Security:** Token-based authentication was implemented for secure access.
- **Smtplib & MIMEMultipart:** Used to send multipart messages through the Simple Mail Transfer Protocol (SMTP).
- **Jinja2:** Employed for generating monthly activity reports on the backend.
- **Bootstrap:** Used for the design of web page templates.
- **SQLite3:** The database structure was created using SQLite3.
- **Flask-SQLAlchemy:** Managed the relational database within the app.
- **Matplotlib:** Used to generate graphs for app statistics on the librarian dashboard.

Database Structure:

- **Database Models:** The app's database was structured using Flask-SQLAlchemy.
- **Tables:** The database consists of five main tables: User, Book, Section, Role, and user_roles.
- **Relationships:** The Book and Section tables have a many-to-one relationship. Similarly, the User and Book tables also follow a many-to-one relationship. User roles are managed through the user_roles table.

System Architecture:

- The design ensures that users are categorized based on their roles, which are managed through the RolesUsers table
  MVC Architecture:
  - **Model (M):** Managed by Flask, which interacts with the database to handle the data model.
  - **View (V):** Implemented using Vue.js, where Vue components deliver an interactive user interface.
  - **Controller (C):** Flask also manages the controller aspect, handling all backend business logic through routing.

- **Folder Structure:**
    - **Instance Folder:** Contains the app's database.
    - **Static Folder:** Houses all graph and image files.
    - **Main.py:** Contains the code to initialize the Flask app instance, Celery instance, and database setup.
    - **Sample_data.py:** Holds preloaded data for the app.
    - **Models.py:** Contains the code for defining database tables.
    - **Worker.py, Celeryconfig.py, Task.py:** Include the code for configuring Celery, setting up scheduled jobs, and managing daily reminders.
    - **Views.py:** Contains the routes and endpoints code.

Features Implemented:

- **User Authentication:**
  Separate login forms for users and librarians, with appropriate alerts for task completions.
- **Librarian Dashboard:**
  Includes detailed statistics on users, books, sections, and visual graphs like Book vs. Rating and Section vs. Number of Books.
- **Book and Section Management:**
  Librarians can manage, create, update, and delete books and sections, as well as control user access to books.
- **Overdue Book Management:**
  Librarians can revoke access to books that have passed their due date via a designated route.
- **Search Functionality:**
  Both librarians and users can search for books by name or author.
- **User Capabilities:**
  Users can request books, read approved content, like or dislike books, and have a limit of requesting up to 5 books at a time.
- **Monthly and Daily Notifications:**
    - A monthly activity report is automatically sent to the creator's email on the first day of each month.
    - Users receive daily notifications via Google Chat to revisit the app if they've been inactive for 24 hours.

Running the App:

- To start the application, run the `main.py` file.

Presentation Video Link:

- https://drive.google.com/file/d/1Jg0zFbk6nz42nRJs-wLXGggWcqpVJ4DR/view?usp=sharing