MAD-I Project Report

Name: Sai Ashwin Kumar C
Email ID: 21f2000619@ds.study.iitm.ac.in

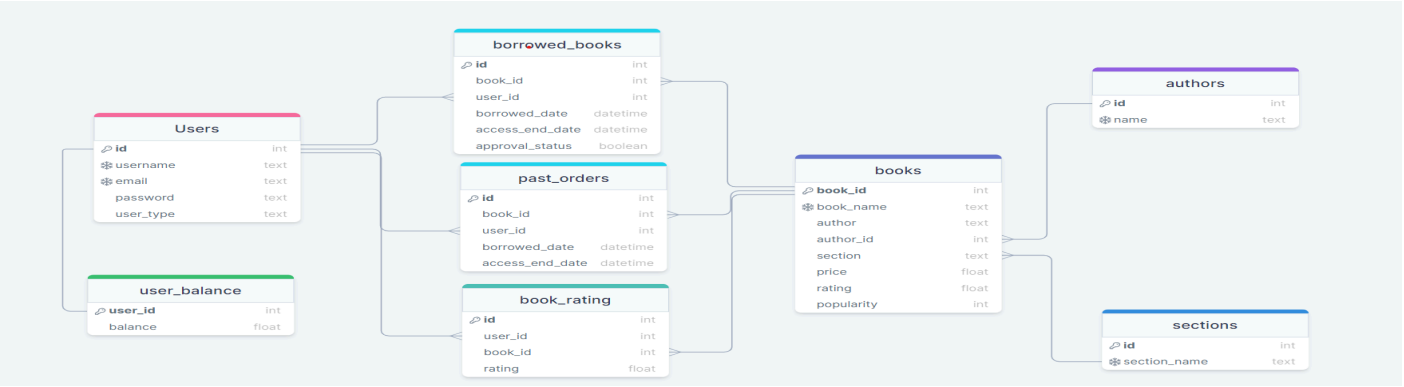
Description

This project report details the development of a library management web application using Flask, a Python web framework. The system is designed to manage library operations, encompassing user authentication, session management, and dynamic content delivery. Utilizing tools like SQLAlchemy for database interactions, Flask-Login for user sessions, and Jinja2 for HTML content rendering, the platform supports complex functionalities such as book borrowing, search and filter capabilities, and administrative oversight. The application ensures robust security measures, seamless data handling, and provides interactive user interfaces that enhance both administrative tasks and user experiences.

Tools Used

- **Flask:** A compact web framework in Python designed for the creation of web applications.
- **SQLAlchemy:** This library implements Object-Relational Mapping (ORM) to facilitate database management.
- **Flask-Login:** This component manages user authentication and session handling.
- **Flask-Session:** Dedicated to managing server-side user sessions.
- **Flask-CORS:** Allows for Cross-Origin Resource Sharing (CORS), essential for processing requests across different domains.
- **Plotly:** A sophisticated library for creating interactive data visualizations.
- **Jinja2:** A templating engine used to generate dynamic HTML content.
- **SQLite:** A streamlined relational database management system.
- **Werkzeug:** Employs hashing and verification techniques to enhance password security.

Database Schema Design



Constraints

1. All the columns in the tables of the database are set to have non-nullable entries to avoid empty records.
2. User name and email in Users table, author name in authors table, section name in sections table are set to have unique values to avoid duplicate entries.

Architecture and Features

This project employs Flask, a Python web framework, for building a web application to manage library operations efficiently. It features comprehensive user management, book handling, and dynamic content generation.

1. **Database Management:** SQLAlchemy, an ORM, is used to streamline database operations with SQLite, enhancing data handling and scalability through well-structured tables for books, authors, and transactions.
2. **User Authentication and Session Management:** Flask-Login manages user authentication, enhancing security and user experience by differentiating roles and storing sessions on the filesystem.
3. **Security Features:** Security is bolstered by Werkzeug for password security and Flask-CORS to safely handle cross-origin requests.
4. **Dynamic Content Rendering:** Jinja2 templates render HTML dynamically, integrating seamlessly with Flask's routing to pass SQL data to the frontend.
5. **Interactive User Interface:** Forms for adding and managing books, user registration, and book borrowings update the database in real time, facilitated by SQLAlchemy.
6. **API Endpoints:** The application supports custom API endpoints for book-related actions like borrowing, extending, and returning, ensuring smooth integration with other systems or frontends.
7. **Administrator Features:**
   - Administrators can efficiently manage the library's catalog with options to add, update, or delete books and sections. This includes the ability to add new authors to the database.
   - Book borrowing requests require administrative approval, giving admins control over user access to book PDFs. Admins can approve or deny these requests effectively.
   - Administrators can monitor active borrows and have the authority to revoke access at any time, automatically transferring the book to the user's past orders.
   - Admin can also view "Library Statistics", which provides visualization for various metrics that can be tracked from the database exploiting libraries like plotly and matplotlib.
8. **User Capabilities:**
   - Users can borrow books, with an interactive dialog box asking them to specify the duration of the borrow period. A maximum of five active borrows is allowed per user.
   - The search functionality allows users to filter and sort books by author, section, name, price, rating, or popularity.
   - Users have the capability to extend or return borrowed books, with each transaction adjusting their pre-allocated account balance based on the duration of the book being borrowed. The system also provides a dashboard displaying remaining balances and the past orders of the user with the value of each order.

Overall, this project illustrates a well-rounded approach to building a Flask-based web application with robust database interactions, secure user authentication, and dynamic content management, making it suitable for scalable web applications that require detailed record-keeping and user interaction.