Library Management System

Author

Name : Paul Binu
Roll No. : 22f2001283
Student Email : 22f2001283@ds.study.iitm.ac.in

Description

This project aims to implement an Online Library Management System. The objective is to create a multi-user app that manages e-books and its corresponding sections. Users can request books to read for a limited time. Admins can perform CRUD operations on books and sections. The project is titled “Codexify” and has implemented all the functionalities.

Technologies Used

- **Flask**: It is used as the primary web framework for developing the project due to its simplicity and compatibility with Python Libraries.
- **Jinja2**: It is a modern templating engine integrated with Flask to create reusable templates and maintain a consistent layout across different pages in the project.
- **Bootstrap**: It is a front-end framework for enhancing user interface and providing responsive and visually appealing web pages.
- **Flask-SQLAlchemy**: It integrates SQLAlchemy and ORM library with Flask for database management within the project, allowing easy CRUD operations on the database.
- **APScheduler**: It is a library for scheduling tasks and it is integrated in this project to automatically revoke access to books when users fail to return it before its due date.
- **Plotly**: It is a graphing library for creating interactive data visualizations using charts and graphs. It is enabled for both admins and users to generate visual representations.

DB Schema Design

1. **User Table**: Stores information about users registered in the system. Each user has a unique user ID, username, password, name, and a flag indicating whether they are an admin or not.
2. **Book Table**: Contains details about the books available in the library. Each book has a unique book ID, name, author, content, publisher, publish date, and a foreign key reference to the section it belongs to.
3. **Section Table**: Represents different sections or categories in the library where books are organized. Each section has a unique section ID, name, creation date, and a description.

4. **Request Table**: Records users' requests to borrow books from the library. Each request is identified by a unique request ID and includes the user ID, book ID, section ID, issue date, due date, return date (if returned), and a flag indicating whether the request is revoked.
5. **Feedback Table**: Stores feedback provided by users about specific books. Each feedback entry includes a unique feedback ID, user ID, book ID, and content of the feedback.

Architecture and Features

- The application begins with **app.py**, which serves as the entry point of the Flask application, initializing the app, configuring it, and running the server.
- In **routes.py**, the routes of the application are defined along with their corresponding logic for handling HTTP requests and returning appropriate responses. It imports necessary modules from Flask, the application's models, and other necessary dependencies.
- In **models.py**, database models for the application are defined using Flask-SQLAlchemy. These models represent tables in the database, including relationships between tables.
- The **templates** folder contains all the HTML pages that are rendered by the Flask routes. These HTML templates define the structure, layout, and content of the web pages displayed to users when they visit different routes of the application. Each HTML file within the folder corresponds to a specific route or page of the application.

Key features of the project:

- The login and registration pages are common for both users and admins. Validation is ensured for secure access to the corresponding accounts.
- Users can browse available sections and books, update their profiles, and visualize account activity through charts. Book requests come with automatic revoke if the book is not returned within the due date. Feedbacks can also be provided once users complete and return the book, which is featured in the book information page. The search bar is implemented so that users can search for books or sections.
- Admins can perform CRUD operations for books, sections, requests, and users. Admins can manually revoke or undo revoke on issued books, and can also issue requests for a custom time period. Admins can manage book information pages and feedbacks for each book. They can also view graphs to view user activity and visualize library data.

Video

Presentation video for project demonstration : https://youtu.be/452zahIfA0M