Library Management System

Name: CHEEDELLA UDAY

Admission No.: 23F1002528

Description:

Library Management System is a project focussed on taking Library online. It is a web application serving as a platform for users to read/download books upon request/purchase. It is a multi-user platform with one administrator (Librarian).

Technology Stack:

- **Flask –** is a web framework that allowed me to build lightweight web applications quickly and easily with Flask Libraries, it helped me manage routing, create web pages, handle user/admin requests.
- **Flask-SQLAlchemy** to work with databases in my Flask application.
- **Jinja2** as the templating engine to generate dynamic HTML content. It allows me to combine Python code with HTML templates, making it easier to display data from my application.
- **CSS –** used this to style my html templates. Used internal and inline CSS for my project.
- **Reportlab -** used this to generate PDFs for the added books for user to download them later.
- **flask_apscheduler –** used to implement auto-revoke functionality which invokes auto_revoke_books method after every scheduled no. of hours.

DB Schema:

- **Admin:**

  Stores information about administrators, including their username and password.

- **Users:**

  Contains details about library users, including their name, password, and other relevant information. It also maintains relationships with other tables for various user-related operations using a unique user_id.

- **Sections:**

  Represents different sections or categories in the library.

- **Books:**

  Stores information about books, including their name, content, author, section ID, price, and creation date.

- **BookIssue:**

  Tracks issued books, including the user ID, book ID, issue date, and return date.

- **BookRequest:**

  Records book requests made by users, along with the user ID, book ID, and number of days requested.

- **UserNotifications:**

  Stores notifications for users, including the user ID, notification message, and date.

- **UserBookRating:**

  Manages user ratings for books, including the user ID, book ID, and rating.

- **BookRating:**

  Tracks the average rating and count of ratings for each book.

- **BookPurchase:**

  Records book purchases made by users, including the user ID and book ID.

- **AdminTools:**

  Contains configuration settings for administrators, such as the maximum number of book requests allowed per user.

Architecture and Functionalities:

A python executable file **app.py** contains:

- setup for my Flask application which creates Flask app object, setting up the database connection, initiating Flask AP Scheduler.
- DB Model to describe database schema using Flask-SQLAlchemy, where tables are defined using python classes with the table fields as class variables. I've also defined how the tables are connected using relationships.
- endpoints or routes with suitable functions that are invoked when users make requests. These functions handle HTTP requests by managing form data, interacting with the database using models, and rendering HTML templates.

General routes/APIs:

- Login (user/admin), register, change password, logout are basic routes for user and admin.

Librarian Specific routes/APIs:

- Admin specific routes provides ability to perform CRUD operations on Sections, books.
- Request routes fetches all the requests from database and renders them and grant/reject routes help librarian to accept/reject user book requests.
- Issuance routes to view issued books to users and also help librarian to revoke access to books issued to user.
- A scheduler is set in place with a task to auto revoke issued books from book_issue table if the date of return is passed, it is scheduled to run at a set period. It also pushes a notification to let the user know.
- Stats routes gives admin a few measurements like user count, no. of books in the library and etc. to keep track of library.
- Admin can search for sections based on section name, search books based on book name and author inside a particular section. In Book requests and Book Issuance admin can put a filter based on book name or user name.
- Once a book is added to a particular section, admin can still move that book to any section of his desire using move route.
- Change pass route enables admin to change their current password.
- Admin tools route provides admin the ability to alter the maximum number of books a user can request.

User Specific routes/APIs:

- Dashboard route renders dashboard template which gives user a few book recommendations based on rating and recent additions in library.
- Books route fetches all the books along with average book rating in the library and renders them. It also lets user swich between desired section and allows user to search the books based on book name and author. User can request any book for a certain number of days or can purchase a book altogether.
- There are validations in place to not let user exceed request limit specified by admin. The code also pushes an error for the same.
- Issued books route provides information of what books user have access to.
- Requested books route fetches the book details of a particular user and displays the information, also a cancel route is provided to cancel a request.
- Purchased books route gathers user purchased book details and a download API generates a PDF using reportlib package and lets user download as an attachment.
- I've introduced notifications for user, for example a welcome notification is pushed when user registers to greet the user. Each time a book is requested/issued/changes password a notification is pushed. These notifications are accessible to user under notifications tab. In addition to it, user can choose to clear any notification or clear all notifications.
- User can read the book If they are issued access to or purchased any book.
- User can provide feedback as rating on a scale of 1-5 for a specific book. User can also edit rating as well. A function is invoked in backend to update average book rating and count of ratings(on unique user basis) in bookRating table.

Additional functionalities:

I've provided validations across various sections such as,

- Push an error if the user name is not unique on sign up.
- Passwords of min-length 8.
- First name and last name of the user must be in alphabet set.
- All the fields which should accept numeric value are validated with regex patterns along with appropriate message like field to accept no. of days for book request must be a numeric value.
- In login, validations like user doesn't exist, incorrect password is set.
- Any mandatory fields are handled.
- Also have backend validations before storing/selecting from database.

Demonstration video link:

https://drive.google.com/file/d/19mf5uxMmmKG33LFUb2RnOpV721r2ts_Z/view?usp=sharing