[redacted]

Description

This is a project about Online Library Management. There will be one admin and many users for the web dev app. There is many Sections and many books in those sections for the users to read or purchase . The admin can perform CRUD Operations on Section and Books .

Technologies Used

- **Flask** as it helps me build web applications in Python, making it easier to handle user requests, manage routing, and create web pages.
- **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.
- **Bootstrap** for styling tables , forms and buttons .
- **Chart.js** to create interactive charts and graphs for my dashboard.
- **Werkzeug Security** for password hashing in my application.

DB Schema Design

- **User Table:**

  Contains user information such as *username, password, name, and admin* status. Each user has a *unique ID* and can be associated with book requests.

- **Section Table:**

  Represents different sections in the library. Each section has a *unique ID, name, creation date,* and *description.*

- **Book Table**:

  Stores information about books available in the library. Includes details like *book name*, *content, authors, date added, price*, and the section it belongs to. Each book is associated with a specific section.

- **BookRequest Table:**

  Tracks requests made by users to borrow books. Includes *user name*, *book name*, *request date*, *return date*, and *request status*. Users can request books, and their requests are stored here.

- **BookIssue Table:**

  Records book issues to users, indicating which books are issued, when they were issued, *return dates, approval status, and feedback*. Links *user names, book names, authors, issue dates*, and related book requests. Tracks the status of book issues and user feedback

Architecture and Features

app.py file contains the setup for my Flask application, including creating the Flask app object, setting up the database connection, and importing necessary files like routes.py

routes.py file is responsible for defining the routes (URL endpoints) of my application that users can access. It contains the logic for handling HTTP requests and returning appropriate responses, such as rendering HTML templates, processing form data, and interacting with the database through models.

models.py file defines the database models for the application using Flask-SQLAlchemy. It contains Python classes that represent tables in your database, including fields and relationships between tables.

Basic Routes for Users and Admin:

- Register, login, logout and profile routes to manage their accounts and authentication.

CRUD Operations for Sections and Books:

- Routes for admins to create, read, update, and delete sections and books in the library.

Admin-Specific Routes:

- Routes for admins to view all books, sections, book requests, issue history, and user feedback.
- Dashboard route to access analytics and insights.
- Routes to accept/deny book requests and revoke book access for users.

User-Specific Routes:

- User actions include requesting books, buying books, viewing book issue history, and managing currently issued books.
- Users can also return books and provide feedback about specific books to enhance the library experience**.**

Search Functionality:

- Both admin and regular users can search based on:
  - ➢ Section name
  - ➢ Book name
  - ➢ Book authors

Bar Charts:

- Displays the number of records in each model (User, Section, Book, BookRequest, BookIssue).
- Shows the top 5 most issued books based on the number of times they have been issued)

Pie Chart:

- Pie chart: Illustrates the distribution of books in each section by showing the number of books in each particular section.

Video

https://youtu.be/CwYJEzy__9k