Project Report
Modern Application Development - I

Kabilan (22f2000945)

March 15, 2024

1 Introduction

This project report presents a comprehensive overview of my app development project aimed at creating a basic and user-friendly platform for managing a digital library. The primary objective of this project is to develop a scalable library management application that allows users to browse, search, request, read, and return from a collection of digital books conveniently.

In the subsequent sections of this report, we will delve into an overview, the structure, functionalities, and implementation details of my modern application development project. We will also discuss the API endpoints, database schema, and package structure, providing a comprehensive overview of the project's technical aspects.

2 Project Overview

Through the utilization of Flask, the lightweight web framework for Python, coupled with SQLAlchemy for database management, this project demonstrates the capabilities of modern web development tools in creating dynamic and interactive applications.

The application provides functionalities for user registration, user authentication, form validation, password hashing for security and role-based access control to ensure data security and privacy. Users can search for books based on various criteria, such as title, author, or section, and request the books, enhancing the overall user experience. The users also get a page to see their pending book requests, issued and completed books, from where the users can read their issued book(s). The user dashboard displays personal statistics.

Librarians have the authority to add, update, or remove books from the digital library inventory. They can maintain accurate records of available books, including details such as title, author, section, and availability status. They are also responsible for processing user requests for borrowing books. Librarians can view pending book requests, approve or reject them based on availability and user eligibility criteria, and facilitate the issuance and return of books. They can view the dynamically generated visualizations to gain insights into library statistics.

3 Package Structure

The project structure follows a Flask application layout as shown in Figure 1. The `create_db.py` script initializes the database. Within the `flask_app` package, `__init__.py` initializes the Flask application. `api.py` handles API endpoints, `forms.py` contains form definitions, and `models.py` defines database models. `routes.py` contains the route definitions for the application. The `static` directory holds static assets like CSS, images, books uploaded by users, and charts. The `templates` directory contains HTML templates rendered by Flask routes. Finally, `run.py` is the entry point to start the Flask application.

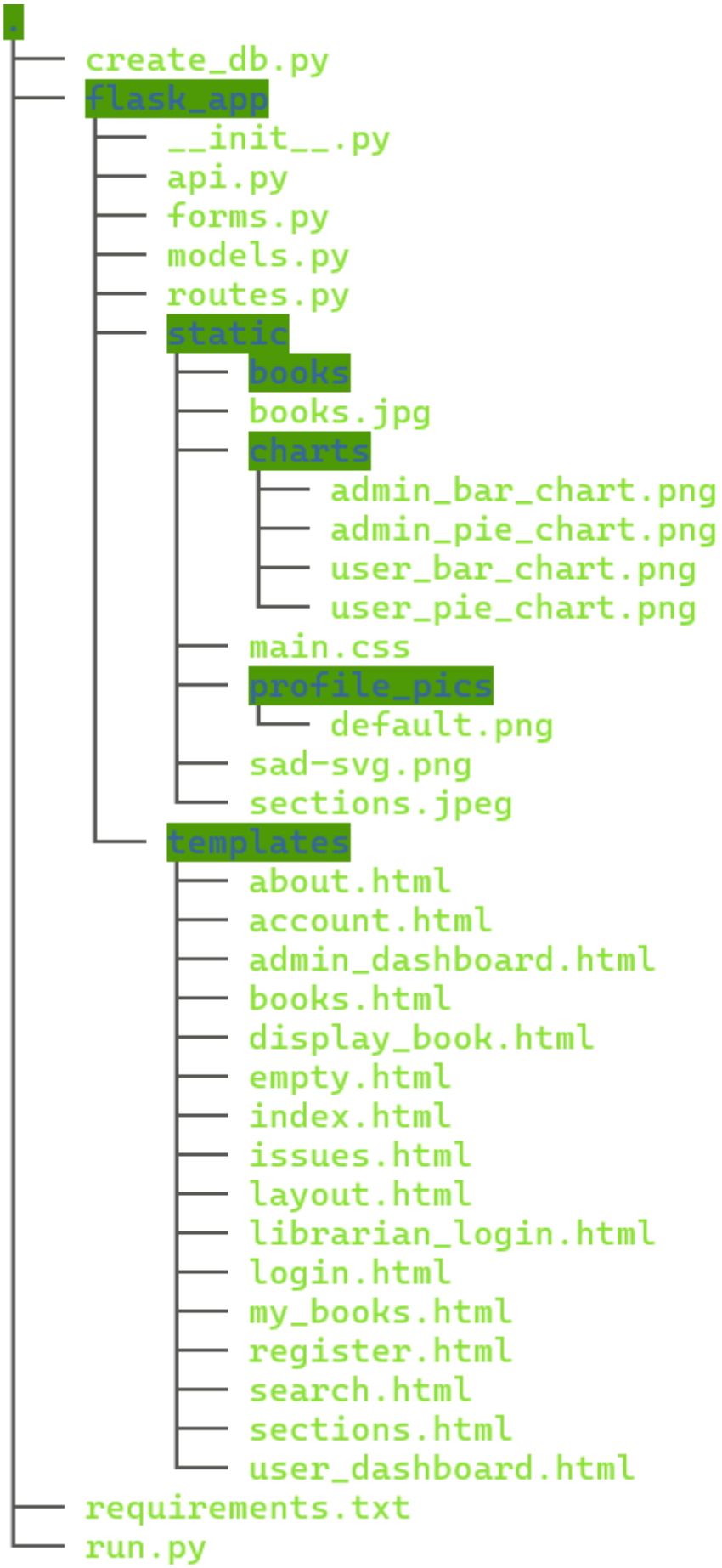

Figure 1: Package Structure

4 Database Schema

The database schema designed for the project is as shown in Figure 2. The schema consists of several tables that represent entities in the application.

4.1 User Table

The **User** table stores information about the users of the application. Each user is uniquely identified by an *id* field. The table includes fields for *username*, *email*, *password*, *role*, and *profile_pic*. The *role* field determines the user's role in the system, with a default value of *user*. The *profile_pic* field stores the path to the user's profile picture.

4.2 Section Table

The **Section** table represents different sections or categories of books available in the application. Each section has a unique identifier (*id*) and includes fields for *name*, *date_created*, and *description*. The *name* field stores the name of the section, *date_created* stores the creation date of the section, and *description* provides a brief description of the section.

4.3 Book Table

The **Book** table contains information about individual books available in the application. Each book is uniquely identified by an *id* field and is associated with a section through a foreign key reference. The table includes fields for *name*, *pdf*, *authors*, and *section_id*. The *name* field stores the title of the book, *pdf* stores the path to the PDF file of the book, *authors* lists the author(s) of the book, and *section_id* references the section to which the book belongs.

4.4 BookRequest Table

The **BookRequest** table tracks requests made by users to borrow books from the library. Each request is uniquely identified by an *id* field and includes fields for *book_id*, *user_id*, *days_requested*, *request_status*, *date_requested*, and *return_date*. The *book_id* and *user_id* fields reference the book and user making the request, respectively. The *days_requested* field stores the number of days the book is requested for, *request_status* indicates the status of the request, *date_requested* stores the date of the request, and *return_date* specifies the expected return date of the book.

4.5 Feedback Table

The **Feedback** table allows users to provide ratings and reviews for books they have read. Each feedback entry is uniquely identified by an *id* field and includes fields for *user_id*, *book_id*, *rating*, *review*, and *date_posted*. The *user_id* and *book_id* fields reference the user providing the feedback and the book being reviewed, respectively. The *rating* field stores the user's rating for the book, *review* contains the user's review text, and *date_posted* records the date when the feedback was posted.

5 API Endpoints

The application comes with the following basic endpoints for interaction.

- **/api/section**: This endpoint allows CRUD operations on sections. It supports GET, POST, PUT, and DELETE methods for managing sections.
- **/api/book**: This endpoint allows CRUD operations on books. It supports GET, PUT, and DELETE methods for managing books.
- **/api/graph/bar_chart**: This endpoint serves the bar chart image for the admin dashboard.
- **/api/graph/pie_chart**: This endpoint serves the pie chart image for the admin dashboard.

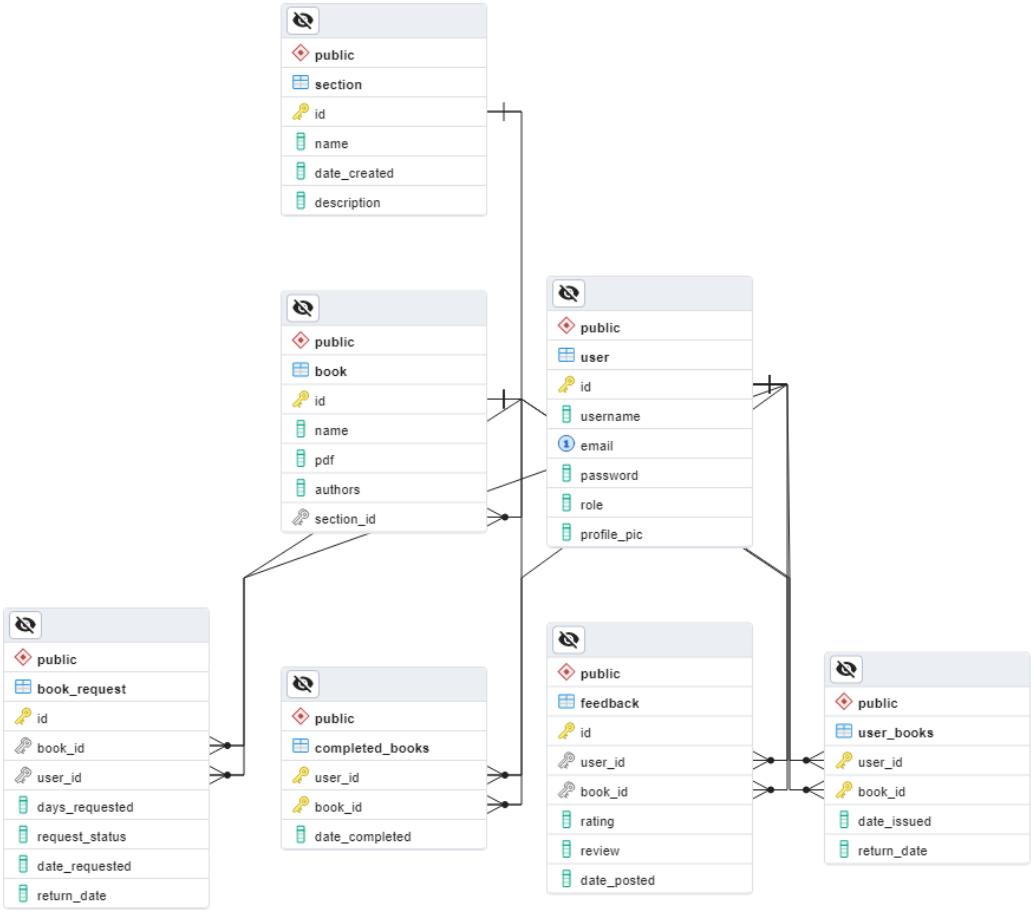

Figure 2: Database Schema Diagram

6 Technologies Used

The following technologies were used in the development of this project:

- **HTML:** The standard markup language for creating web pages.
- **CSS:** Cascading Style Sheets for styling web pages.
- **Bootstrap:** A popular CSS framework for developing responsive websites.
- **Flask:** A micro web framework for Python.
- **SQLAlchemy:** An SQL toolkit and Object-Relational Mapping (ORM) library for Python.
- **Flask-Login:** A Flask extension for managing user sessions.
- **Flask-WTF:** A Flask extension for handling web forms.
- **Flask-Bcrypt:** A password hashing library.
- **Flask-RESTful:** A Flask extension for creating REST APIs.
- **Matplotlib:** A plotting library for creating visualizations.
- **Seaborn:** A statistical data visualization library based on Matplotlib.

7 Demonstration Video

Click the following link to watch the video : Video