Project Report for Modern Application Development I

Student Details:

Name- Avnish Jajodia

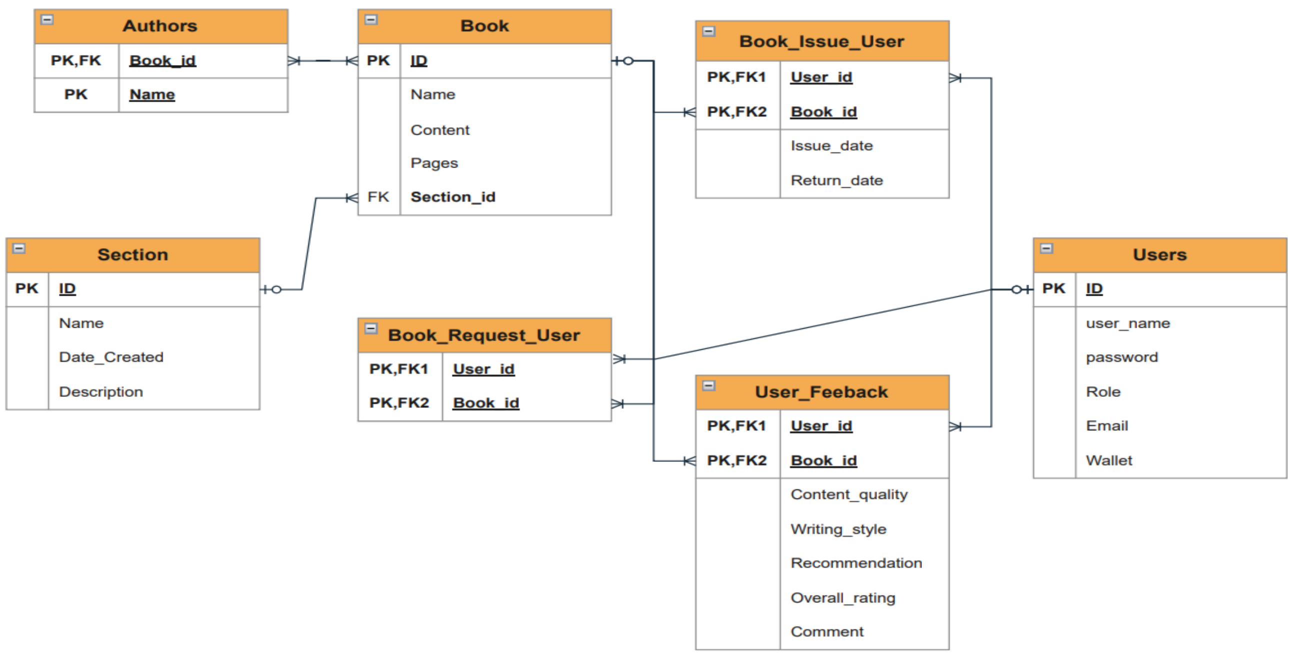
Roll Number – 22F3001777

The topic for my project was to develop a Library Management System. So, for this problem I firstly conducted a comprehensive analysis of the problem statement. Following that, I drew a rough ER Diagram on paper to conceptualize the system's structure. Subsequently, I outlined the necessary tables and devised the business logic for my application. After that utilizing DB Browser, I defined the database schema. Beginning with the backend, I crafted the model.py file using Flask-SQLAlchemy library. Then I went about defining my controllers for the application and created separate files for user and librarian controllers to ensure clarity of my code and easy debugging. After all this was done, I went for my Frontend part wherein I used Bootstrap for my styling and visual appearance. Jinja templates were utilized for integration with the Flask framework. I have also defined some APIs for my application in the api.py file using Flask-restful library.

Model:

The data model for my project comprises of seven tables, each serving distinct purposes within the Library Management System. These tables are designed using the principles of Relational Database Management System, and is implemented in SQLite with the help of DB browser, using Flask-SQLAlchemy.

The Entity-Relationship diagram for my model is given below (drawn with the help of draw.io). The ER diagram clearly elucidates the relationship between the entities, including primary keys and foreign keys.



API endpoints:

In addition to core functionalities of my application, I have also developed some Application Programming Interfaces (APIs) specifically designed to perform CRUD operations on books as well as sections. The endpoints for APIs have been described below:

Book API Endpoints:

- **GET** /api/book/<int:book_id> :

  Outcome:

  200: Returns details of the book if found.

  404: If the book with the given ID is not found.

  500: Internal server error if encountered during processing.

- **DELETE** /api/book/<int:book_id> :

  Outcome:

  200: If the book is successfully deleted.

  404: If the book with the given ID is not found.

  500: Internal server error if encountered during processing.

- **PUT** /api/book/<int:book_id> :

  Outcome:

  200: If the book is successfully updated.

  404: If the book with the given ID is not found.

  500: Internal server error if encountered during processing.

  400: error is raised if the provided Section ID does not exist.

- **POST** /api/book:

  Outcome:

  200: If the book is successfully added.

  400: If required fields (Name, Content) are missing or if the provided section ID does not exist.

  500: Internal server error if encountered during processing.

Section API Endpoints:

- **GET** /api/section/<int:section_id> :

  Outcome:

  200: Returns details of the section if found.

  404: If the section with the given ID is not found.

  500: Internal server error if encountered during processing.

- **DELETE** /api/section/<int:section_id> :

  Outcome:

  200: If the section is successfully deleted along with related books.

  404: If the section with the given ID is not found.

  500: Internal server error if encountered during processing.

- **PUT** /api/section/<int:section_id> :

  Outcome:

  200: If the section is successfully updated.

  404: If the section with the given ID is not found.

  500: Internal server error if encountered during processing.

- **POST** /api/section:

  Outcome:

  200: If the section is successfully added.

  400: If required fields (Name, Date_Created) are missing.

  500: Internal server error if encountered during processing.