JAN 2024 , MADI
Library Management APP
22f2000936

Application Design

1. **User Authentication and Authorization**:
   - Implement user authentication to differentiate between regular users and admins.
   - Allow the primary admin to grant or revoke admin access to other users.

2. **Admin Functionality**:
   - **Admins can manage books and sections:**
     - Add, view, update, and delete books and sections.
     - Add books directly to a section or search for books within a section.
     - Option to delete sections with or without deleting the books in them.
     - Add or remove books from sections.
   - **Admins can manage requests:**
     - Approve or reject requests for book issuance.
     - Revoke access to any book from any user.
     - View the users to whom a book is issued and their profiles.
   - **Admin dashboard:**
     - Display various real-time data plots from the logs table using Plotly Py and Dash.

3. **User Functionality**:
   - **Users can manage their profile:**
     - Update profile information.
     - View the books currently issued to them.
   - **Request functionality:**
     - Users can request books, which are pending admin approval.
     - Limit users to have at most 5 pending requests at a time.
     - Approved or rejected requests are automatically deleted periodically, preserving the last 5 requests.
   - **Book issuance:**
     - Users can have at most 5 free issues at a time.
     - Choose to return books or automatically return them after the return date.
     - For paid permanent buys, users get access to a default PDF viewer for downloading.

4. **Feedback System**:
   - Users can provide feedback for each book once, regardless of whether the book is issued to them.

5. **Search Functionality**:
   - Both users and admins can search for books and sections.
   - Admins can also search for users.

6. **Viewing Books**:
   - Users and admins can view books in a particular section.

7. **Implementation Details**:
   - Use Flask for the backend.
   - Utilize SQLAlchemy for database interactions.
   - Incorporate Plotly Py and Dash for creating interactive data visualizations.
   - Implement a PDF rendering library like PyMuPDF for viewing books page by page.

8.**Free Issue with PDF Viewer Preview**:

- Users can request a free issue for a book.
- Before issuing the book, users can preview the first two pages of the book using a PDF viewer-like structure.
- After previewing, users can choose to proceed with the issue.
- If the request is approved, users can access the full book through the PDF viewer.
- Users can view the book page by page and navigate between pages

JAN 2024 , MADI
Library Management APP
22f2000936

Here's a brief description of each model in my project:

1. **User**: This model stores user login credentials and information about the books each user has. It also includes admin data. Each user can have multiple books associated with them.

2. **Books**: This model contains all the data related to books, such as author, file paths, and the user to whom the book is issued. It serves as a central repository for book-related information.

3. **Section**: This model represents a group of one or more books. If a specified section does not exist when adding a new book, a new section is automatically created. It includes attributes such as section name, description, date created, and the books it contains.

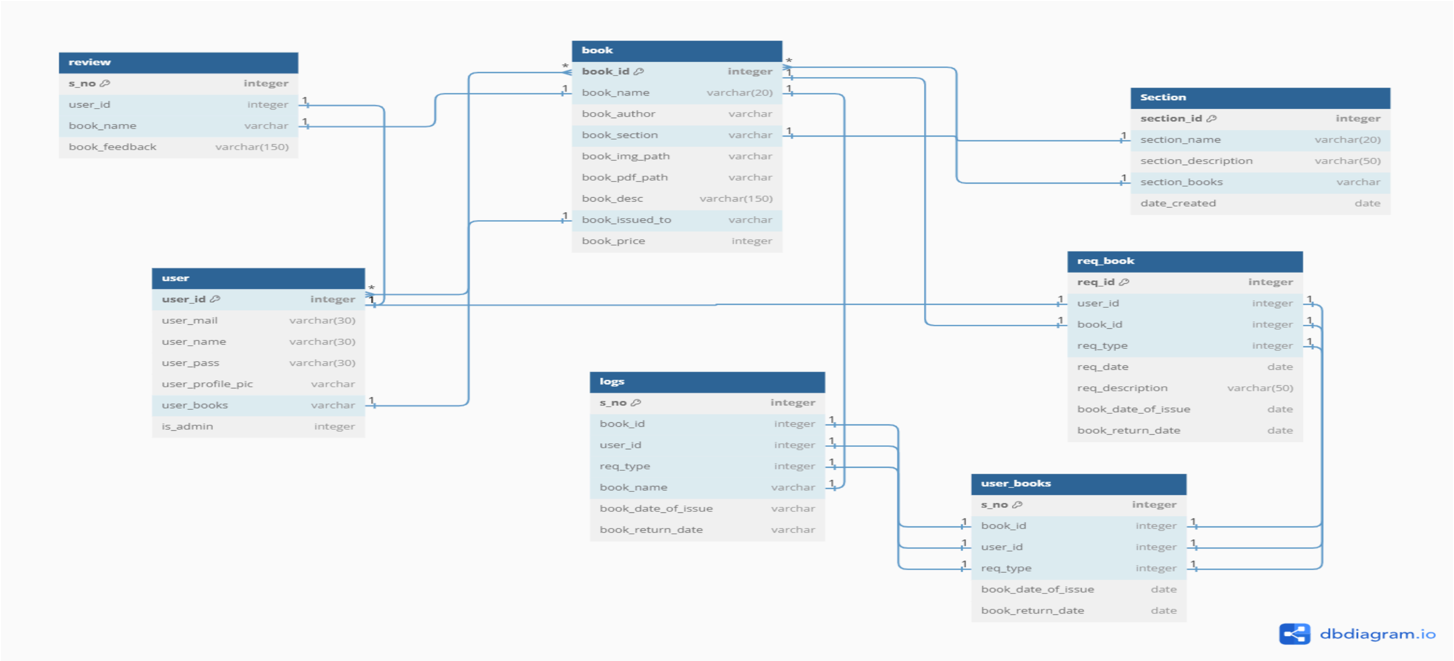
4. **Req_Book**: This model manages requests for books that are pending approval or rejection by an admin. It includes attributes such as request ID, user ID, book ID, request type (e.g., issue or permanent download), date of issue, and return date. It is automatically cleared up periodically, keeping only the last 5 requests for each user.

5. **User_Book**: After a request is approved, the data moves to this model, which represents books issued to users. It includes attributes such as request ID, user ID, book ID, request type, date of issue, and return date. Entries are automatically cleared when the return date arrives.

6. **Logs**: This model acts as an offline register, storing data similar to User_Book and including the book name. Entries are added when requests are approved but are never deleted, making it useful for graphing and analysis purposes.

7. **Review**: This model allows users to provide feedback about books. Each user can leave feedback about a book only once. It includes attributes such as user ID, book name, and book feedback.

Database Structure



review	
s_no	integer
user_id	integer
book_name	varchar
book_feedback	varchar(150)

book	
book_id	integer
book_name	varchar(20)
book_author	varchar
book_section	varchar
book_img_path	varchar
book_pdf_path	varchar
book_desc	varchar(150)
book_issued_to	varchar
book_price	integer

Section	
section_id	integer
section_name	varchar(20)
section_description	varchar(50)
section_books	varchar
date_created	date

user	
user_id	integer
user_mail	varchar(30)
user_name	varchar(30)
user_pass	varchar(30)
user_profile_pic	varchar
user_books	varchar
is_admin	integer

req_book	
req_id	integer
user_id	integer
book_id	integer
req_type	integer
req_date	date
req_description	varchar(50)
book_date_of_issue	date
book_return_date	date

logs	
s_no	integer
book_id	integer
user_id	integer
req_type	integer
book_name	varchar
book_date_of_issue	varchar
book_return_date	varchar

user_books	
s_no	integer
book_id	integer
user_id	integer
req_type	integer
book_date_of_issue	date
book_return_date	date

Demo Video Link: https://drive.google.com/file/d/1LEJbc8j1ceI0BIQ1VQ1C5NwI9Sw6qJic/view?usp=sharing