Library Management System Documentation

Backend (Django Rest Framework - DRF)

Database Structure

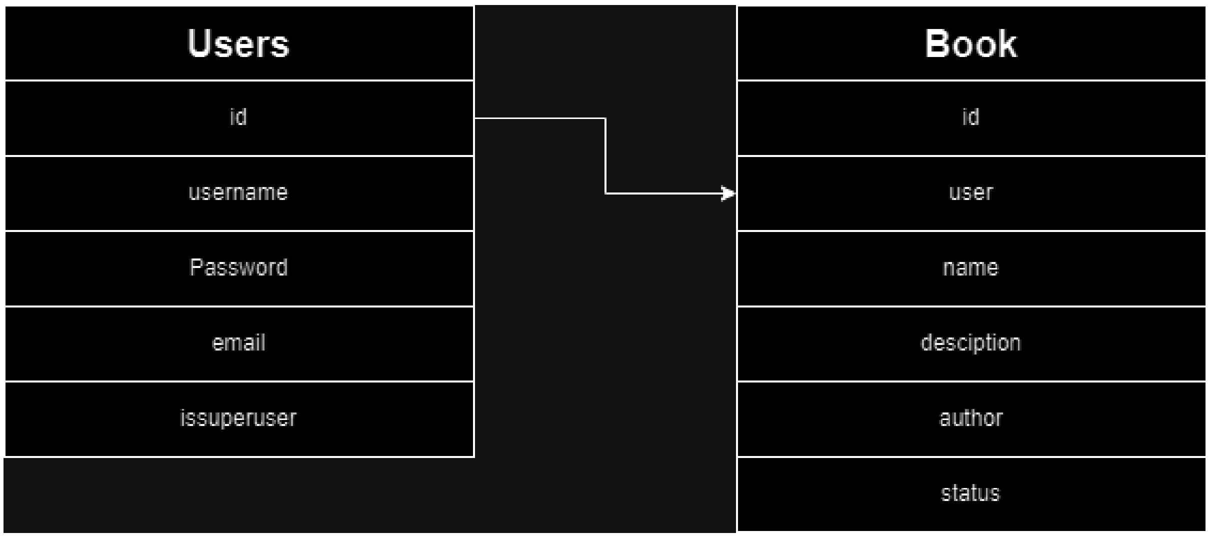
Books Table

- Fields:
  - `id`: Integer (Primary Key)
  - `User`: Integer (Foreign Key)
  - `title`: String
  - `description`: String
  - `author`: String
  - `status`: Boolean (AVAILABLE, BORROWED)

Users Table

- Fields:
  - `id`: Integer (Primary Key)
  - `username`: String
  - `password`: String (hashed)
  - `email`: String
  - `issuperuser`: Boolean (LIBRARIAN, MEMBER)

Database Diagram



<u>API Documentation</u>

User Authentication

- Endpoint: `/api/token/`
- Method: `POST`
- Requirements:
  - `username`: String
  - `password`: String
- Output:
  - `access`: JWT Token

- Endpoint: `/api/token/refresh/`
- Method: `POST`
- Requirements:
  - `Token `: Refresh Token
- Output:
  - `access`: JWT Token

Books Management

- Endpoint: `/book_view`
- Method: `GET`
- Requirements: JWT Token (Librarian, Member)
- Output: List of books

- Endpoint: `/book_add`
- Method: `POST`
- Requirements: JWT Token (Librarian)
- Output: Book Added

- Endpoint: `/book_update/{book_id}/`
- Method: `PUT`
- Requirements: JWT Token (Librarian), Book data
- Output: Updated book details

- Endpoint: `/book_delete/{book_id}/`
- Method: `DELETE`
- Requirements: JWT Token (Librarian)
- Output: Book Deleted

- Endpoint: `/mybooks/`
- Method: `GET`
- Requirements: JWT Token (Member)
- Output: Borrowed Books

Members Management

- Endpoint: `/member/`
- Method: `GET`
- Requirements: JWT Token (Librarian)
- Output: List of members

- Endpoint: `/member/`
- Method: `POST`
- Requirements: JWT Token (Librarian), Member data
- Output: Member Added

- Endpoint: `/member/{member_id}/`
- Method: `PUT`
- Requirements: JWT Token (Librarian), Updated Data
- Output: Member Updated

- Endpoint: `/member/{member_id}/`
- Method: `DELETE`
- Requirements: JWT Token (Librarian)
- Output: Member deletion confirmation

Own Member Delete

- Endpoint: `/ownmemberdelete/`
- Method: `DELETE`
- Requirements: JWT Token (Member)
- Output: Member deletion confirmation

<u>Frontend (HTML, CSS, JS)</u>

Folder Structure

- `/index.html`: Main entry point
- `/librarian.html`: Librarian View
- `/member.html`: Member View
- `/css/style.css`: Stylesheet
- `/css/mystyle.css`: Stylesheet
- `/js/scripts.js`: Main JavaScript file
- `/js/librarian.js`: Librarian file
- `/js/member.js`: Member file

HTML Structure

- Used separate HTML files for Librarian and Member interfaces.

CSS Styling

- Used simple Bootstrap components for styling.
- Customized styles as needed.

JavaScript Functions

- Many JavaScript functions are used for fetching the data from the API.

User Authentication

- `login(username, password)`: Send a POST request to `/api/token/` with user credentials.
- Only the librarian can add the members to the library.
- Librarian credentials are (Username: Admin and Password: Admin).

Books Management

- `getBooks(token)`: Send a GET request to `/book_view` with JWT token for Librarian or Member.

- `createBook(token)`: Send a POST request to `/book_add` with JWT token and book data for Librarian.

- `updateBook(token, bookId, bookData)`: Send a PUT request to `/book_update/{book_id}/` with JWT token and updated book data.

- `deleteBook(token, bookId)`: Send a DELETE request to `/book_delete/{book_id}/` with JWT token.

- `BorrowedBooks(token)`: Send a GET request to `/mybooks/` with JWT token for Member.

Members Management

- `getMembers(token)`: Send a GET request to `/member/` with JWT token for Librarian.

- `addMember(token, data)`: Send a POST request to `/member/` with JWT token and data for Librarian.

- `updateMember(token, data, memberId)`: Send a PUT request to `/member/{member_id}/` with JWT token and updated data for Librarian.

- `deleteMember(token, memberId)`: Send a DELETE request to `/member/{member_id}/` with JWT token.

Self Delete

- - `owndelete(token)`: Send a DELETE request to `/memberowndelete/` with JWT token.

Hosting Instructions

Backend (DRF)

1. Python Anywhere Deployment:
   - Fork this repository: [https://github.com/kiranrokkam09/library]
   - Live At: [https://kiran1432.pythonanywhere.com/]
   - Deploy to Python Anywhere following the steps in the README.

Frontend (HTML, CSS, JS)

1. Vercel:
   - Fork this repository: [https://github.com/kiranrokkam09/library_static]
   - Live At: [https://library-static-1owi.vercel.app/]
   - Host on GitHub Pages following the steps in the README.