Library Management System

Author: Deba Ghosh | DGclasher

Description

The Library Management System is a web application that facilitates efficient management of books and members in a library. Users can register and log in to the system, allowing librarians and members to access various functionalities. Librarians can add, edit, and delete books, as well as manage members' accounts. Members can borrow and return books, view their borrowed books, and manage their own account settings.

Architecture and Design

System Architecture

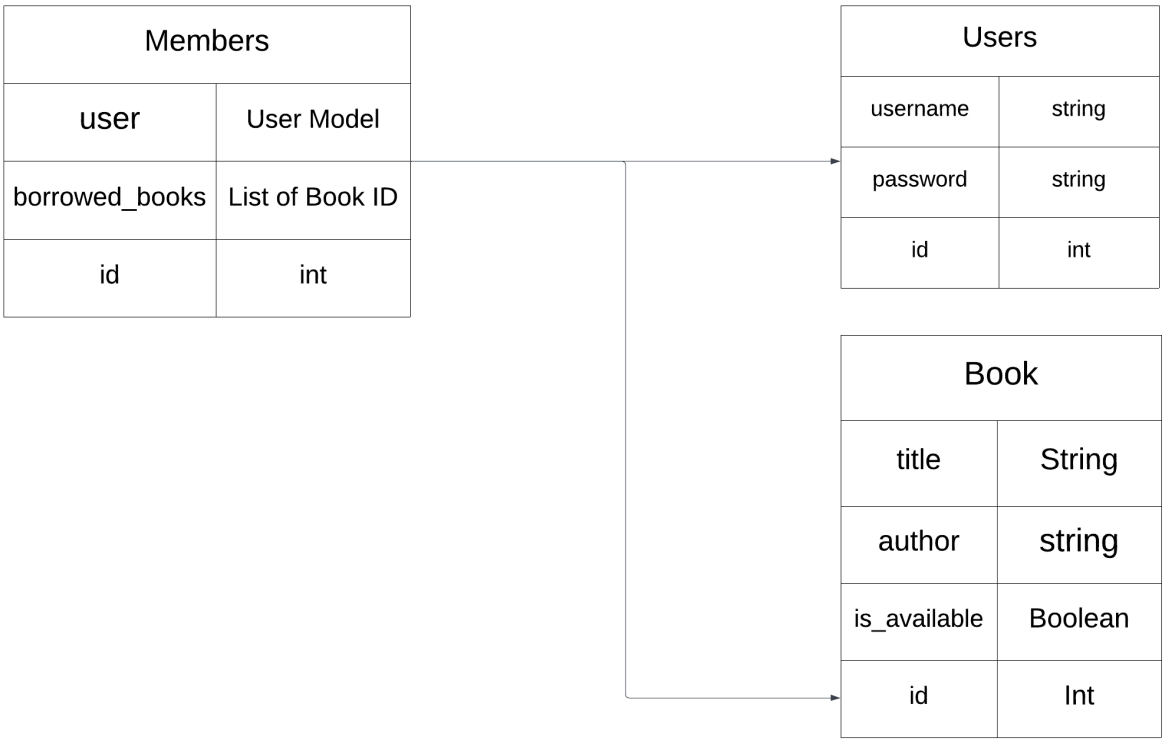
1. **Frontend Client:**
    - The frontend client is responsible for presenting the user interface and interacting with users.
    - It's built using HTML, CSS, and JavaScript, and utilizes the Tailwind CSS framework for styling.
    - JavaScript is used to make asynchronous requests to the backend API endpoints and handle user interactions.
2. **Backend Server (Django):**
    - The backend server is built using the Django framework, a high-level Python web framework.
    - It handles API requests, performs data processing, interacts with the database, and sends responses back to the frontend.
    - Django's built-in features like models, views, serializers, and permissions are used to structure and manage the application.
    - The server is hosted on a local development environment during development and can be deployed to a production server.
3. **Database:**
    - The system uses a MySQL database to store and manage data.
    - Django's ORM (Object-Relational Mapping) is used to define data models and manage database operations.
    - Two main tables are used: `Book` and `Member`. The `Book` table stores information about books, including title, author, and availability. The `Member` table stores user information and the list of borrowed books.
4. **API Endpoints:**
    - The backend exposes a set of API endpoints that the frontend interacts with to perform CRUD (Create, Read, Update, Delete) operations on books and members.

- Endpoints include `/api/books/` for managing books, `/api/members/` for managing members, `/api/books/borrow/` and `/api/books/return/` for borrowing and returning books, and more.

5. **User Authentication and Authorization:**
   - User authentication is managed using Django's built-in authentication system.
   - Different user roles are defined, including librarians and members, with varying levels of access to certain endpoints.
6. **Page Navigation and Interaction:**
   - Users interact with the application through the frontend UI, browsing book lists, borrowing books, and managing their account.
   - Clicking on "Edit" buttons allows users to edit book details, while "Delete" buttons let them remove entries.
   - Borrow and Return buttons beside books enable users to perform these actions with the respective books.
7. **Deployment:**
   - The application can be deployed to a production server for public access.
   - Deployment involves configuring the server environment, database, and application settings.

Database Design

Table Structure



1. **User Table**:
    - This table stores information about users who can log in to the system.
    - It includes standard user fields like username, password, email, and names.
2. **Member Table**:
    - This table is used to associate registered users (members) with their library-related activities.
    - Each member is linked to a user through a foreign key relationship.
    - The borrowed_books field stores the books a member has borrowed through a Many-to-Many relationship.
3. **Book Table**:
    - This table holds information about the books available in the library.
    - Fields include the book's title, author, and a boolean indicator (is_available) to denote whether the book is currently available for borrowing.

User Roles and Permissions

1. **Librarian:**
    - **Description:** Librarians are users with administrative privileges and are responsible for managing the library's resources and members.
    - **Permissions:**
        - View and manage the list of all books.
        - Add new books to the system.
        - Edit book details (title, author, etc.).
        - Delete books from the system.
        - View and manage the list of all members.
        - Edit member details (if necessary).
        - Delete member accounts.
    - **Access:** Librarians can access all sections of the dashboard, including book and member management.
2. **Member:**
    - **Description:** Members are users who can borrow and return books from the library.
    - **Permissions:**
        - View the list of available books.
        - Borrow available books.
        - Return borrowed books.
    - **Access:** Members have access to a limited set of functionalities that pertain to their borrowing and returning of books. They can't manage other members or edit book details.

API Documentation

All API endpoints and methods are documented here.

Deployment

Clone the **shelfsphere-api**

git clone https://github.com/DGclasher/shelfsphere-api

Create python environment

```
python3 -m venv venv
source venv/bin/activate
```

Make a .env file with contents

```
SECRET_KEY=
DB_NAME=
DB_USER=
DB_PASS=
DB_HOST=
DB_PORT=
```

Install dependencies and run required migrations

```
make setup
```

If make not installed

```
python3 manage.py makemigrations
python3 manage.py migrate
```

Start the server

```
python3 manage.py runserver
```

Links

Backend:

- Deployment: https://shelfsphere.pythonanywhere.com
- GitHub: https://github.com/DGclasher/shelfsphere-api

Frontend:

- Deployment: https://shelfsphere.netlify.app
- GitHub: https://github.com/DGclasher/shelfsphere