Author

Name: Shobha Garg
Roll number: 21f3002851
Email: 21f3002851@ds.study.iitm.ac.in
About: I am a bachelor fueled by youthful enthusiasm and I am committed to expanding my knowledge and skills in these domains.

Description

The application is a library management system platform, which allows searching and issuing of books from one or more sections. It is a single admin platform where the users can sign up as a user or an admin. The users can issue the book and view the content. The admin can create, update, delete and view the sections/ books to be displayed to the users. The admin can also restrict / unrestrict a user's book access on the app. The users get the notification regarding any pending returns for new books using GSpace, the admin gets a monthly report about the users in the given month, and it can download data about the books in a CSV format.

Technologies used

- **Python** (Used for creating the backend)
    - Flask (Used to create the web application)
    - Flask-SQLAlchemy (Used for interacting with the SQLite database)
    - Celery (Used for scheduling tasks)
    - Flask-Security (Used for authentication and RBAC)
    - …and related dependencies for the above packages to function
- **Redis** (Used for caching and storing async result)
- **SQLite** (Used for storing data)
- **VueJS** (Used for the front end)
- **CSS3** (Used for custom styling)
- **Bootstrap** (Used for basic styling)

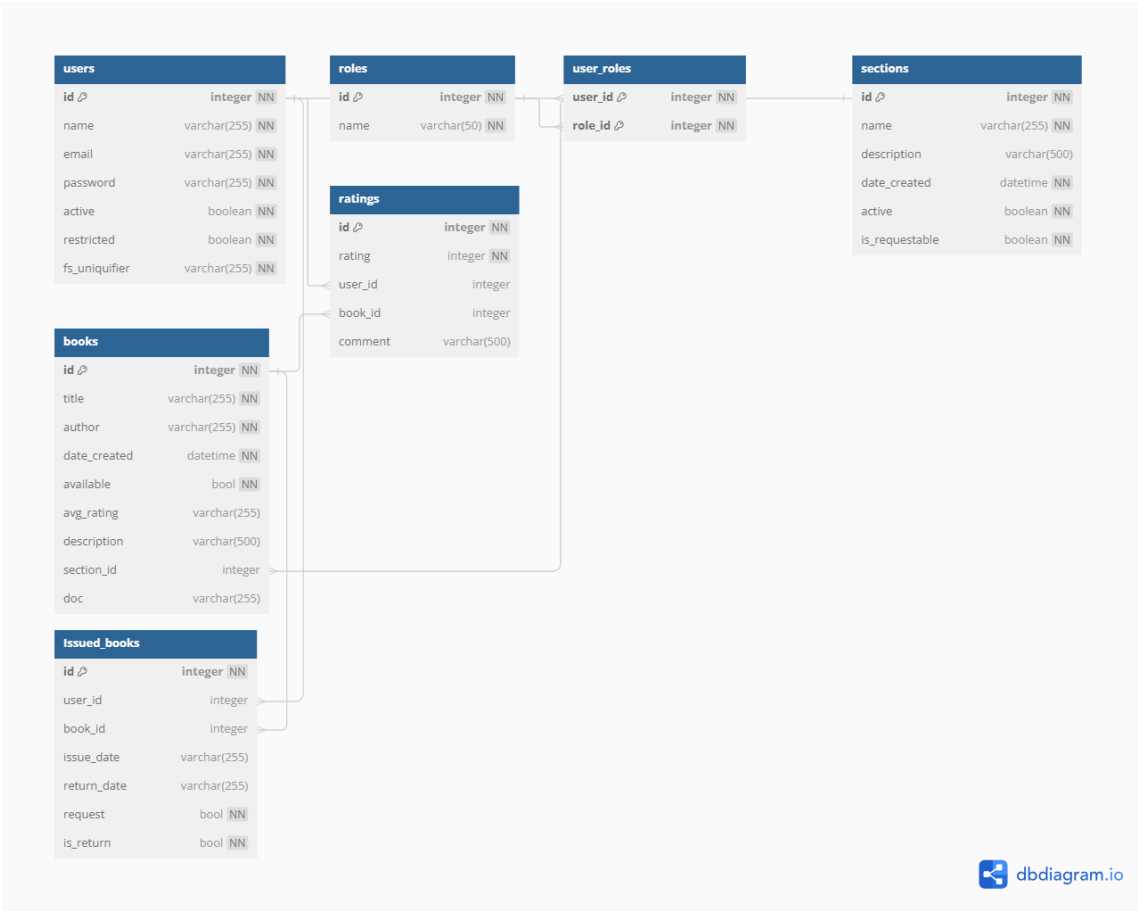
API Design

The API has been created for the following features:

- **Auth**: To perform authentication related functions.
- **Admin (Librarian)**: To perform CRUD on sections and books, approve requests, and manage users by restricting /unrestricting their access to the app and download data about their books.
- **User**: To perform searches across books and sections, issue and return the books, update its profile and browse the marketplace.

The API is secure, it allows restricted access using RBAC to perform these operations.

Database Schema Design



The database was designed to scale, taking a vague inspiration from various library apps. There are 2 main entities: books and sections. These are related to each other via one-to-many relationships, as shown in the diagram above. To accommodate user retention and other business essential metrics, the user watchlist items and books are also stored in the database using necessary models and relationships.

Architecture and Features

The project follows an iteration of the MVC model and follows the fundamental idea of separation of concerns. The models for each of the tables are defined in the `models` folder, the controllers which contain the functional methods with respect to the models are stored in the `controller` folder, the files which host the API endpoints and the usage of controllers with authentication validations are stored in the `views` folder. Additionally, there are some utility methods / classes for authentication, database connection, configuration and exceptions, which are stored in the `utils` folder. All these folders are collectively part of the `app` folder. The SQLite database is stored in the folder called `db`. The `static` and `templates` folder contain the static assets, like CSS files needed for styling the appearance, and the design templates respectively. Further details have been explained in the README.md file in the root folder.

Features implemented in the application are:

- Proper login and signup page for users, and admin using RBAC
- Validation and authenticated access using token-based authentication provided by Flask Security package
- CRUD on books and sections.
- Issuing books across various sections and managing watchlist items.
- Searching for books using various parameters like name and section.
- Displaying all the books and their details to the user.