MAD2 Project Documentation

Author: Karteek Tadimalla
Roll Number: 22f1001543
Email ID: 22f1001543@ds.study.iitm.ac.in

I am currently pursuing a B.E. in Artificial Intelligence & Machine Learning, in my 4th year, at University College of Engineering, Osmania University, Telangana. My career focus is on Data Science, where I am keen to apply my skills and knowledge in this field.

Description:

The **Library Management System** allows **Users** to register and access the portal. Users can request access to eBooks for downloading and search for books by titles and sections. The application enables users to view detailed information about selected eBooks and rate them.

Users receive **email notifications** if they do not log in daily by the end of the day (EOD). This functionality serves as a reminder to log in daily.

Librarians have a separate module to log in and **administer portal** tasks, such as **CRUD** operations on sections and books. They can view and browse all available eBooks and sections, approve or revoke access requests, and generate reports. The application includes a specialized dashboard that allows librarians to view pending requests, granted requests, and returned books. Librarians can grant or reject any eBook access requests made by users. Librarian can grant or reject any request for Ebooks requested by User. Librarian can generate reports also

Technologies Used:

- **Flask** – For API
- **JWT Security** – Involves the use of tokens for authentication and authorization.
- **SQLite and Flak_SQLAlchemy** – For Database and ORM.
- **Pytz** – For timezone-aware datetime handling.
- **Vue.js 3** – A progressive JavaScript framework for building the frontend user interface.
- **Vuex Store** – A state management pattern and library for Vue.js applications.
- **Vue Router** – The official router for Vue.js, enabling navigation between different pages.
- **HTML** – For structuring the web pages.
- **CSS** – For styling the web pages.
- **Jinja2** – A templating engine for Python, used in conjunction with Flask for rendering dynamic HTML content

DB Schema Design

User

- **id**: Integer, Primary Key, Auto-increment.
- **username**: String(80), Unique, Non-nullable.
- **email**: String(120), Unique, Non-nullable.
- **password**: String(128), Non-nullable.
- **role**: Enum('user', 'librarian'), Non-nullable, Default='user'.
- **login_date**: DateTime, Default=datetime.now(pytz.utc).

Section

- **sid**: Integer, Primary Key, Auto-increment.
- **title**: String(255), Non-nullable.
- **date**: String, Non-nullable.
- **description**: Text.
- **books**: Relationship with EBook, dynamic loading.

EBook

- **ebid**: Integer, Primary Key, Auto-increment.
- **title**: String(255), Non-nullable.
- **author**: String(120), Non-nullable.
- **description**: Text.
- **date**: String, Non-nullable.
- **quantity**: Integer.
- **price**: Integer, Non-nullable.
- **file**: String(255).
- **sectionid**: Foreign Key to Section.sid.

Transaction

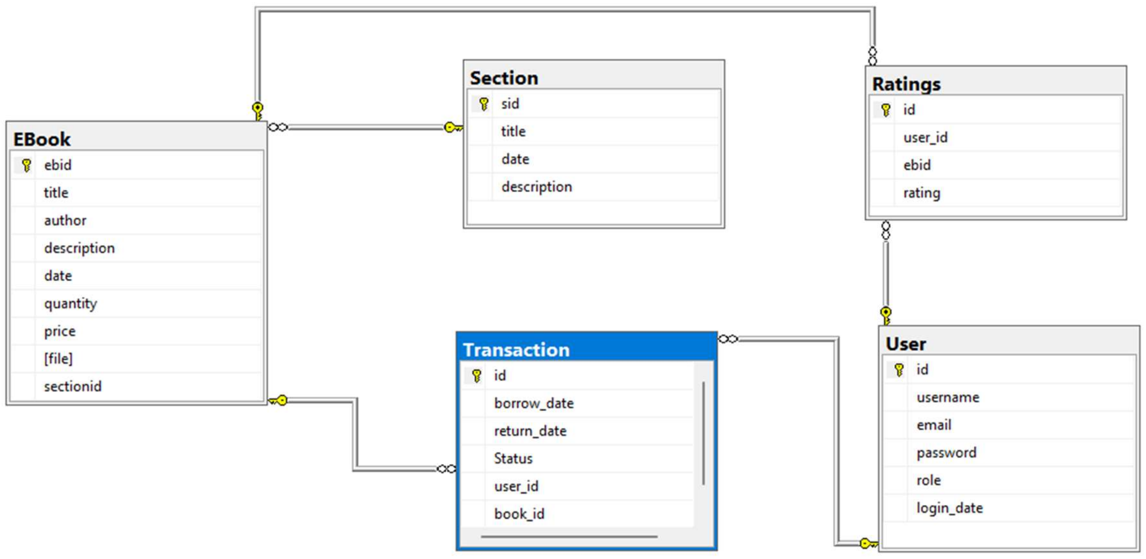
- **id**: Integer, Primary Key, Auto-increment.
- **borrow_date**: Date, Non-nullable.
- **return_date**: Date, Nullable.
- **Status**: String(255).
- **user_id**: Foreign Key to User.id, Non-nullable.
- **book_id**: Foreign Key to EBook.ebid, Non-nullable.

Ratings

- **id**: Integer, Primary Key, Auto-increment.
- **user_id**: Foreign Key to User.id, Non-nullable.

- **ebid**: Foreign Key to EBook.ebid, Non-nullable.
- **rating**: Integer, Non-nullable.

ER Diagram



API Endpoints:

User Endpoints:

- /api/register: POST, OPTIONS
- /api/users/<int:user_id>: DELETE, OPTIONS
- /api/update_user/<int:user_id>: PUT, OPTIONS
- /api/add_section: POST, OPTIONS, GET
- /api/edit-section: POST, OPTIONS
- /api/delete-section: DELETE, OPTIONS, GET
- /api/add_book: POST, OPTIONS, GET
- /api/edit_book: POST, OPTIONS
- /api/delete_book: DELETE, OPTIONS, GET
- /api/pdf/<path:filename>: GET
- /api/getrequests: GET, OPTIONS
- /api/rejectrequest/<int:transaction_id>: DELETE, OPTIONS
- /api/grantrequest/<int:transaction_id>: POST, OPTIONS

Librarian Endpoints:

- /api/liblogin: POST, OPTIONS, GET
- /api/register: POST, OPTIONS
- /api/add_section: POST, OPTIONS, GET
- /api/edit-section: POST, OPTIONS
- /api/delete-section: DELETE, OPTIONS, GET
- /api/add_book: POST, OPTIONS, GET
- /api/edit_book: POST, OPTIONS
- /api/delete_book: DELETE, OPTIONS, GET
- /api/pdf/path:filename: GET
- /api/getrequests: OPTIONS, GET
- /api/rejectrequest/int:transaction_id: OPTIONS, DELETE
- /api/grantrequest/int:transaction_id: OPTIONS, POST

Architecture and Features

The project is structured within the **MAD2** folder, encapsulating both the frontend and backend components.

- **Frontend**: Organized in a dedicated folder that utilizes **Vue.js 3** for building a dynamic and responsive user interface.
- **Backend**: Managed within another folder, where **app.py** serves as the central file for defining application routes and handling requests.
  - Integrates **Celery** for asynchronous task management.
  - Supported by **Redis** as the message broker, ensuring efficient task processing and background job handling.
- **Templates**: Includes a **monthly_activity.html** file, used for rendering monthly activity views.

This modular organization ensures a clear separation of concerns, promoting maintainability and scalability within the application.

Video

22f1001543_MAD2_Video