Library Management System
By Harshit Mishra

API Documentation

1. User Authentication

Register

- **Endpoint:** `/auth/register`
- **Method:** POST
- **Requirements:**
  - `username` (string)
  - `password` (string)
  - `role` (string,e.g., 'LIBRARIAN', 'MEMBER')
- **Output:**
  - `200 OK` with `{'message': 'User registered successfully!'}` on success
  - `400 Bad Request` if any required fields are missing
- **Errors:**
  - `400 Bad Request` if user already exists

Login

- **Endpoint:** `/auth/login`
- **Method:** POST
- **Requirements:**
  - `username` (string)
  - `password` (string)
- **Output:**
  - `200 OK` with `{'access_token': <token>}` on success
  - `401 Unauthorized` if credentials are invalid
- **Errors:**
  - `401 Unauthorized` if user does not exist or password is incorrect

Logout

- **Endpoint:** `/auth/logout`
- **Method:** POST
- **Requirements:** JWT Token
- **Output:**
  - `200 OK` with `{'message': 'Logout successful!'}` on success
- **Errors:**

    - `401 Unauthorized` if token is invalid or missing

2. Book Management

Add Book

- Endpoint: `/books/`
- Method: POST
- Requirements:
    - JWT Token with role 'LIBRARIAN'
    - `title` (string)
    - `author` (string)
- Output:
    - `201 Created` with `{'message': 'Book added successfully!'}` on success
- Errors:
    - `403 Forbidden` if user does not have 'LIBRARIAN' role

Update Book

- Endpoint: `/books/<id>`
- Method: PUT
- Requirements:
    - JWT Token with role 'LIBRARIAN'
    - `title` (string)
    - `author` (string)
- Output:
    - `200 OK` with `{'message': 'Book updated successfully!'}` on success
    - `404 Not Found` if book does not exist
- Errors:
    - `403 Forbidden` if user does not have 'LIBRARIAN' role

Delete Book

- Endpoint: `/books/<id>`
- Method: DELETE
- Requirements:
    - JWT Token with role 'LIBRARIAN'
- Output:
    - `200 OK` with `{'message': 'Book deleted successfully!'}` on success
    - `404 Not Found` if book does not exist
- Errors:
    - `403 Forbidden` if user does not have 'LIBRARIAN' role

View Books

- **Endpoint:** `/books/view`
- **Method: GET**
- **Output:**
  - `200 OK` **with a list of books in JSON format**
- **Errors:**
  - **None**

Borrow Book

- **Endpoint:** `/books/<id>/borrow`
- **Method: POST**
- **Requirements:**
  - **JWT Token with role 'MEMBER'**
- **Output:**
  - `200 OK` **with** `{'message': 'Book borrowed successfully!'}` **on success**
  - `400 Bad Request` **if book is already borrowed**
  - `403 Forbidden` **if book is borrowed by someone else**
- **Errors:**
  - `403 Forbidden` **if user does not have 'MEMBER' role**

Return Book

- **Endpoint:** `/books/<id>/return`
- **Method: POST**
- **Requirements:**
  - **JWT Token with role 'MEMBER'**
- **Output:**
  - `200 OK` **with** `{'message': 'Book returned successfully!'}` **on success**
  - `400 Bad Request` **if book is already available**
  - `403 Forbidden` **if user did not borrow the book**
- **Errors:**
  - `403 Forbidden` **if user does not have 'MEMBER' role**

3. User Management

Add Member

- **Endpoint:** `/users/add`
- **Method: POST**
- **Requirements:**
  - **JWT Token with role 'LIBRARIAN'**
  - `username` **(string)**

    - `password` (string)
- Output:
    - `201 Created` with `{'message': 'Member added successfully!'}` on success
- Errors:
    - `403 Forbidden` if user does not have 'LIBRARIAN' role

Update Member

- Endpoint: `/users/<id>`
- Method: PUT
- Requirements:
    - JWT Token with role 'LIBRARIAN'
    - `username` (string)
    - `password` (string, optional)
- Output:
    - `200 OK` with `{'message': 'Member updated successfully!'}` on success
    - `404 Not Found` if member does not exist
- Errors:
    - `403 Forbidden` if user does not have 'LIBRARIAN' role

Delete Member

- Endpoint: `/users/<id>`
- Method: DELETE
- Requirements:
    - JWT Token with role 'LIBRARIAN'
- Output:
    - `200 OK` with `{'message': 'Member deleted successfully!'}` on success
    - `404 Not Found` if member does not exist
- Errors:
    - `403 Forbidden` if user does not have 'LIBRARIAN' role

View Members

- Endpoint: `/users/`
- Method: GET
- Requirements: JWT Token with role 'LIBRARIAN'
- Output:
    - `200 OK` with a list of members in JSON format
- Errors:
    - `403 Forbidden` if user does not have 'LIBRARIAN' role

Delete Own Account

- **Endpoint:** `/users/me`
- **Method: DELETE**
- **Requirements: JWT Token with role 'MEMBER'**
- **Output:**
  - `200 OK` **with** `{'message': 'Account deleted successfully!'}` **on success**
- **Errors:**
  - `403 Forbidden` **if user does not have 'MEMBER' role**

DataBase Schema

User
Id (p.k.) int
username (varchar)
Password (varchar)
role (varchar)

Book
Id (p.k.) (integer)
title (varchar)
author(varchar)
status(varchar)
Borrower_id (f.k.) (integer)

Code

class User(db.Model):

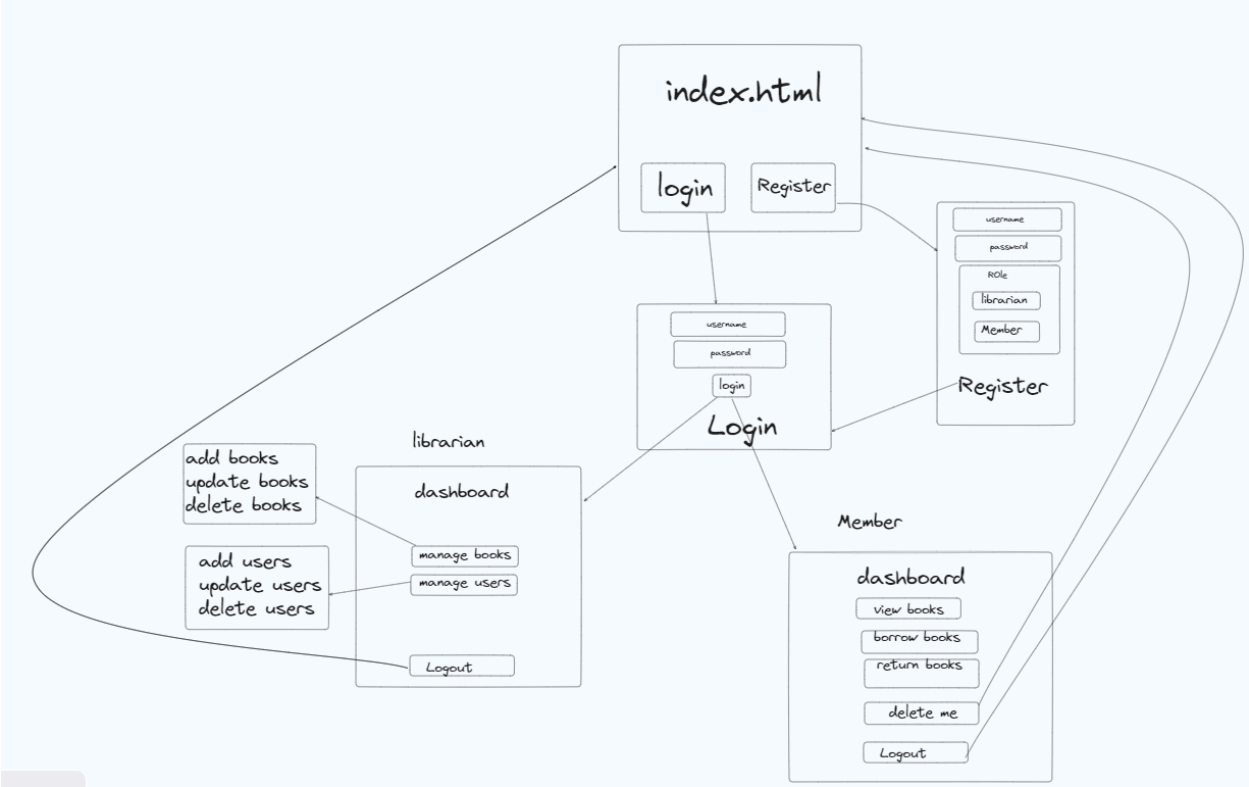
```
id = db.Column(db.Integer, primary_key=True)
username = db.Column(db.String(80), unique=True, nullable=False)
password = db.Column(db.String(120), nullable=False)
role = db.Column(db.String(10), nullable=False)
```

class Book(db.Model):

```
id = db.Column(db.Integer, primary_key=True)
title = db.Column(db.String(120), nullable=False)
author = db.Column(db.String(120), nullable=False)
status = db.Column(db.String(20), default='AVAILABLE')
borrower_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=True)
borrower = db.relationship('User', backref='borrowed_books')
```

Frontend Flow

Link:- https://excalidraw.com/#json=VvN93-2e-TZdLe748Y5Kl,mFPXTpDnPiDXaV_mAky4kg



Hosting Solution

Render

Backend is Deployed on Render Using github, it consists of CI/CD Pipeline to the GitHub Repository which streamline the deployment process which makes deployment smooth.

1. Created a Github Repository
2. Committed and omitted code according to the requirements by the deployment server.
3. Created a Procfile defining the status and location of the of app file in flask
4. Then clone on the render and then deployed it to the render

GitHubPages

The Frontend is Hosted here as it was asked not to use any framework and deploying any html,css and js is pretty good on git pages.

1. Created a Github Repository
2. Committed code to the git pages
3. Go to the git pages section in the setting of the git repo
4. Choose branch and root folder from where the page to be deployed
5. Now your frontend is hosted on the github pages

Links For all of the work I have done:

Backend

GitHub: https://github.com/auscode/library_management
Live Hosted: https://lib-mgmt.onrender.com/

Frontend

GitHub: https://github.com/auscode/Frontend
Live Hosted: https://auscode.github.io/Frontend/

Complete structure

Link:-
https://excalidraw.com/#json=VvN93-2e-TZdLe748Y5Kl,mFPXTpDnPiDXaV_mAky4kg

My GitHub:

Link: https://github.com/auscode