Assignment Submission Portal Documentation

Table of Contents

Project Overview

The Assignment Submission Portal is a web-based application designed to facilitate the submission and review of assignments. Users can register, log in, and upload assignments, while admins can review and manage those assignments.

Technologies Used

- **Flask**: A micro web framework for Python.
- **MongoDB**: NoSQL database for storing user and assignment data.
- **Python 3.13.0**: The programming language used for development.
- **Flask-PyMongo**: An extension that simplifies working with MongoDB in Flask.

Getting Started

Prerequisites

- Python 3.13.0 or higher
- MongoDB installed and running
- Flask and other dependencies installed (see `requirements.txt`)

Installation

- **Clone the repository:**

  ```
  git clone
  https://github.com/yoginath-it/Assignment-Submission-Portal.
  git
  cd Assignment
  ```

- **Set up a virtual environment:**

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

- **Install the dependencies:**

  ```
  pip install -r requirements.txt
  ```

- **Start the Flask application:**

  ```
      python app.py
  ```

API Endpoints

User Endpoints

- **Register User**
    - **URL**: `/user/register`
    - **Method**: `POST`
    - **Request Body**: json

      ```
      {

        "username": "string",
        "password": "string"
      }
      ```

    - **Response**:
        - `201 Created`: User registered successfully.
        - `400 Bad Request`: Invalid input.
- **Login User**
    - **URL**: `/user/login`
    - **Method**: `POST`
    - **Request Body**: json

      ```
      {

        "username": "string",
        "password": "string"
      }
      ```

    - **Response**:
        - `200 OK`: Login successful.
        - `401 Unauthorized`: Invalid credentials.

- **Upload Assignment**
    - **URL**: `/user/upload`
    - **Method**: `POST`
    - **Request Body**: json

      ```
      {
        "userId": "string",
        "task": "string",
        "admin": "string"
      }
      ```

    - **Response**:
        - `201 Created`: Assignment uploaded successfully.
        - `400 Bad Request`: Invalid input.
- **Fetch Admins**
    - **URL:** `/user/admins`
    - **Method:** `GET`
    - **Request Body:** json
    - **Response**:
        - `201 Created`: Returns a list of all admins.
        - `400 Bad Request`: No admins found.

Admin Endpoints

- **Register Admin**
    - **URL**: `/admin/register`
    - **Method**: `POST`
    - **Request Body**: json

      ```
      {
        "username": "string",
        "password": "string"
      }
      ```

    - **Response**:
        - `201 Created`: Admin registered successfully.
        - `400 Bad Request`: Invalid input.

- **Login Admin**
    - **URL**: `/admin/login`
    - **Method**: `POST`
    - **Request Body**: json

      ```
      {

        "username": "string",
        "password": "string"
      }
      ```

    - **Response**:
        - `200 OK`: Login successful.
        - `401 Unauthorized`: Invalid credentials.
- **View Assignments**
    - **URL**: `/admin/assignments`
    - **Method**: `GET`
    - **Response**:
        - `200 OK`: Returns a list of assignments tagged to the admin.
        - `404 Not Found`: No assignments found.
- **Accept Assignment**
    - **URL**: `/admin/assignments/<assignment_id>/accept`
    - **Method**: `POST`
    - **Response**:
        - `200 OK`: Assignment accepted successfully.
        - `404 Not Found`: Assignment not found.
- **Reject Assignment**
    - **URL**: `/admin/assignments/<assignment_id>/reject`
    - **Method**: `POST`
    - **Response**:
        - `200 OK`: Assignment rejected successfully.
        - `404 Not Found`: Assignment not found.

Sample CURL Requests

User Endpoints:

1. **Register User**

```
curl -X POST http://127.0.0.1:5000/user/register \
-H "Content-Type: application/json" \
-d '{
  "username": "user1",
  "password": "pass123"
}'
```

2. **Login User**

```
curl -X POST http://127.0.0.1:5000/user/login \
-H "Content-Type: application/json" \
-d '{
  "username": "user1",
  "password": "pass123"
}’
```

3. **Upload Assignment**

```
curl -X POST http://127.0.0.1:5000/user/upload \
-H "Content-Type: application/json" \
-d '{
  "userId": "user1",
  "task": "Hello World",
  "admin": "admin1"
}'
```

4. **Fetch All Admins**

```
curl -X GET http://127.0.0.1:5000/user/admins \
-H "Content-Type: application/json"
```

Admin Endpoints:

1. **Register Admin**

```
curl -X POST http://127.0.0.1:5000/admin/register \
-H "Content-Type: application/json" \
-d '{
  "username": "admin1",
  "password": "pass123"
}’
```

2. **Login Admin**

```
curl -X POST http://127.0.0.1:5000/admin/login \
-H "Content-Type: application/json" \
-d '{
  "username": "admin1",
  "password": "pass123"
}'
```

3. **View Assignments**

```
curl -X GET http://127.0.0.1:5000/admin/assignments \
-H "Content-Type: application/json"
```

4. **Accept Assignment**

```
curl -X POST
http://127.0.0.1:5000/admin/assignments/<assignment_id>/accept
```

5. **Reject Assignment**

```
curl -X POST
http://127.0.0.1:5000/admin/assignments/<assignment_id>/reject
```

Database Structure

Users Collection

- **username**: String
- **password**: String (hashed)
- **role**: String (either "user" or "admin")

Assignments Collection

- **userId**: String (ID of the user)
- **task**: String (assignment task)
- **admin**: String (admin assigned to the task)
- **status**: String (e.g., "pending", "accepted", "rejected")

Error Handling

- **400 Bad Request**: Indicates a problem with the request data.
- **401 Unauthorized**: Invalid credentials provided during login.
- **404 Not Found**: The resource requested was not found.
- **500 Internal Server Error**: An unexpected error occurred on the server.

Conclusion

This documentation provides a detailed overview of the Assignment Submission Portal project. It includes information on the project's functionality, setup instructions, API endpoints, and error handling. Feel free to extend or modify the documentation as needed.