

Assignment Submission Portal

Overview

The **Assignment Submission Portal** is a backend system developed using Flask and MongoDB. It allows users to upload assignments and enables admins to manage these assignments by accepting or rejecting them. The system utilizes JWT for secure authentication.

Technologies Used

- **Programming Language:** Python
- **Web Framework:** Flask
- **Database:** MongoDB
- **ORM:** PyMongo
- **Authentication:** Flask-JWT-Extended
- **Password Hashing:** Werkzeug
- **Tool for Testing :** Postman

Installation Instructions

Prerequisites

- Python 3 installed on your system.
 - Download python from official webpage if not in the system
[Download Link](#)
- MongoDB server running locally or on a cloud service.
 - To install MongoDB locally → [Download Link](#)
 - Install Mongosh → [Download Link](#)
- **I preferred in Localhost**
- After Installing , create a connection which is
mongodb://127.0.0.1:27017
- We can use Mongosh which is MongoDB shell

- We can start by going to command prompt and enter “mongosh”

```
C:\Users\PC>mongosh
Current Mongosh Log ID: 6707e2c5f10195676d86b01c
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.2
Using MongoDB:      8.0.0
Using Mongosh:       2.3.2

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

-----
  The server generated these startup warnings when booting
  2024-10-10T10:03:23.320+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
  -----

test>
```

-
- We can also check here whether the user, admins, assignments are adding to the database or not

```
test> show dbs
Assignment 144.00 KiB
admin       40.00 KiB
config      108.00 KiB
local       72.00 KiB
test> use Assignment
switched to db Assignment
Assignment>
```

- Assignment> db.users.find().pretty()


```
[
  {
    _id: ObjectId('67067c41be42aa4d22179ae0'),
    username: 'Kumar',
    password: 'scrypt:32768:8:1$NVbC1myfr9Z0WvDZ$943ba5a80de939a2acc76d160f2338c13041e9a600b8d50d4eb8205d6d4035032ad5194900Fae11f8d767322a781d37c590283f55e5e21454ecf1157195a3d6e',
    role: 'user'
  },
  {
    _id: ObjectId('67067c5bbe42aa4d22179ae1'),
    username: 'Dev',
    password: 'scrypt:32768:8:1$4a4ZkFCEVPdn8QmP$61def7910874d82ed90cbb9fc0bde964c4657041f75422f2f2636cff8e449c520880b2c1bf3d40a14f2903b96071d3fabbd3bdb75e169a8c74b46da0229a3aef',
    role: 'admin'
  }
]
```

- Assignment> db.assignments.find().pretty()


```
[
  {
    _id: ObjectId('67067ca6be42aa4d22179ae2'),
    userId: 'Kumar',
    task: null,
    admin: null,
    submitted_at: ISODate('2024-10-09T18:22:54.125Z'),
    status: 'pending'
  },
  {
    _id: ObjectId('67067cf6be42aa4d22179ae3'),
    userId: 'Kumar',
    task: 'Hello',
    admin: 'Dev',
    submitted_at: ISODate('2024-10-09T18:24:14.576Z'),
    status: 'pending'
  }
]
```
-

Step 1:

To install the required packages using pip, run the following commands in the terminal of VS Code Editor:

```
pip install Flask
```

```
pip install Flask-PyMongo
```

```
pip install Flask-JWT-Extended
```

```
pip install Werkzeug
```

Step 2:

Once you've installed the required dependencies,

Set up MongoDB:

```
app.config["MONGO_URI"] = "mongodb://localhost:27017/Assignment"
```

Step 3:

Run the Flask Application

Now that the dependencies are installed and MongoDB is set up, we can run our Flask application: using **python app.py**

```
PS C:\Users\PC\Desktop\Assignment Submission Portal> python app.py
* Serving Flask app 'app'
* Debug mode: on
```

Step 4: Register Users and Admins

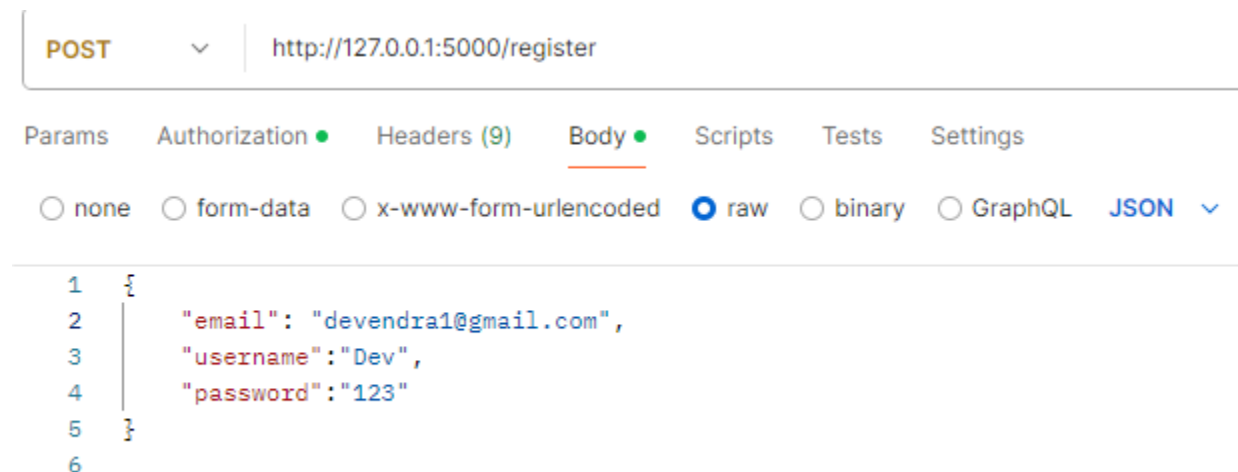
Use tool **Postman** to interact with your API. You can now create user and admin accounts.

We can download Postman from [Download Link](#)

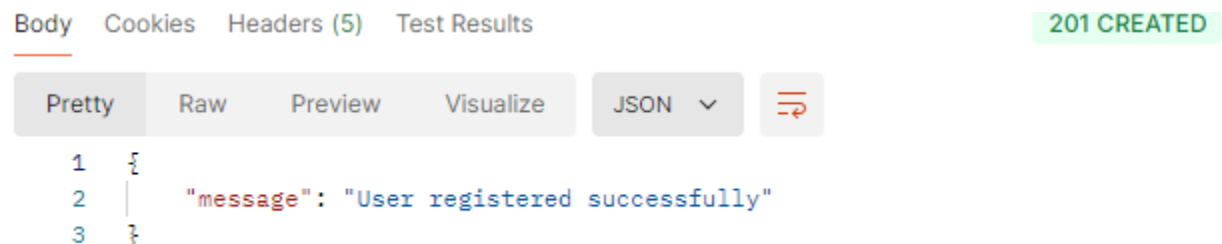
Endpoints Overview

User Endpoints

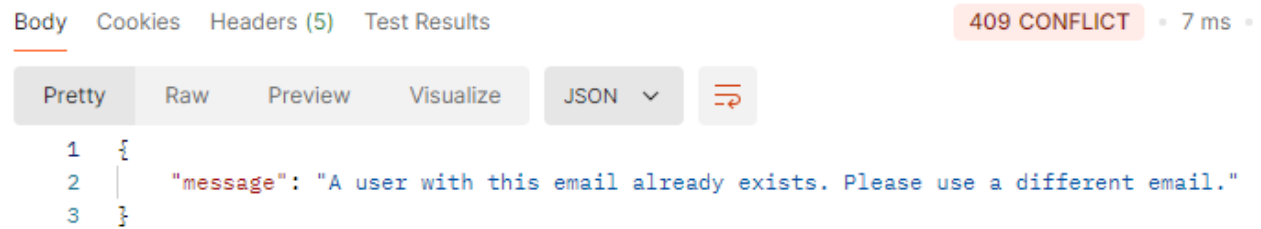
POST /register - Register a new user.



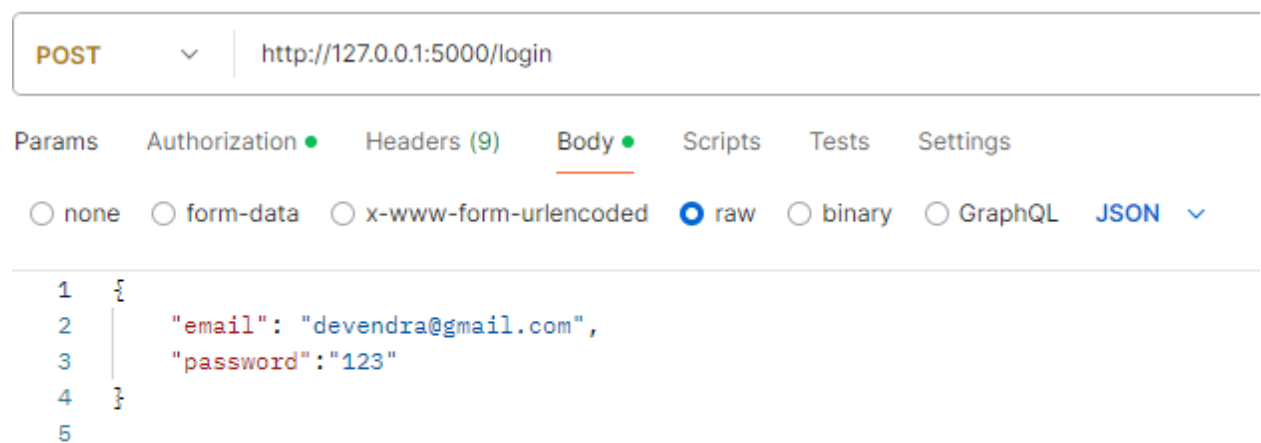
Response:



On Duplicate User:

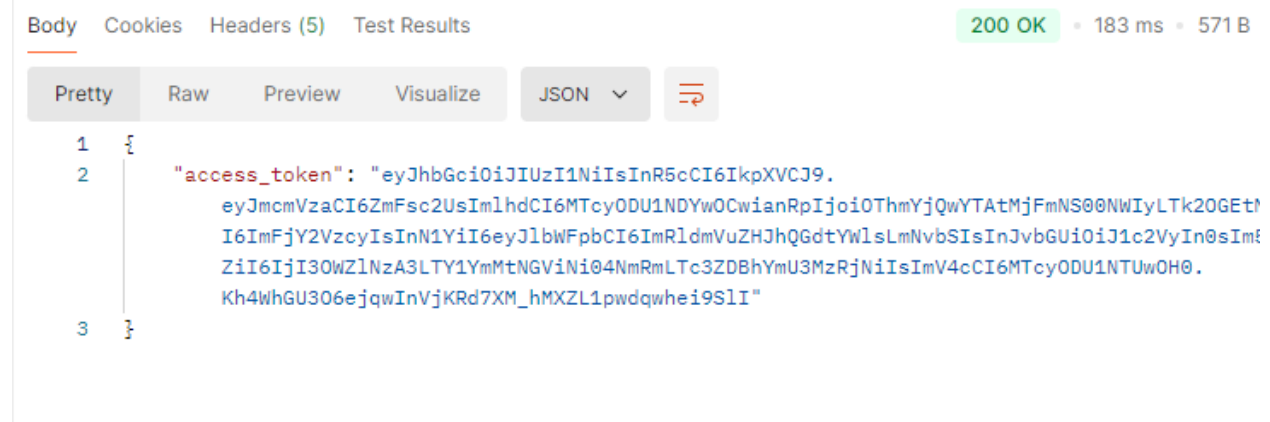


POST /login - User login.

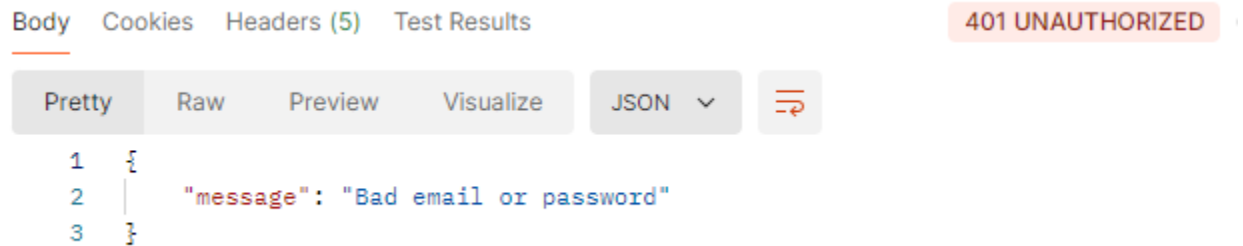


Response:

On Success: 200 status code with JWT access token.



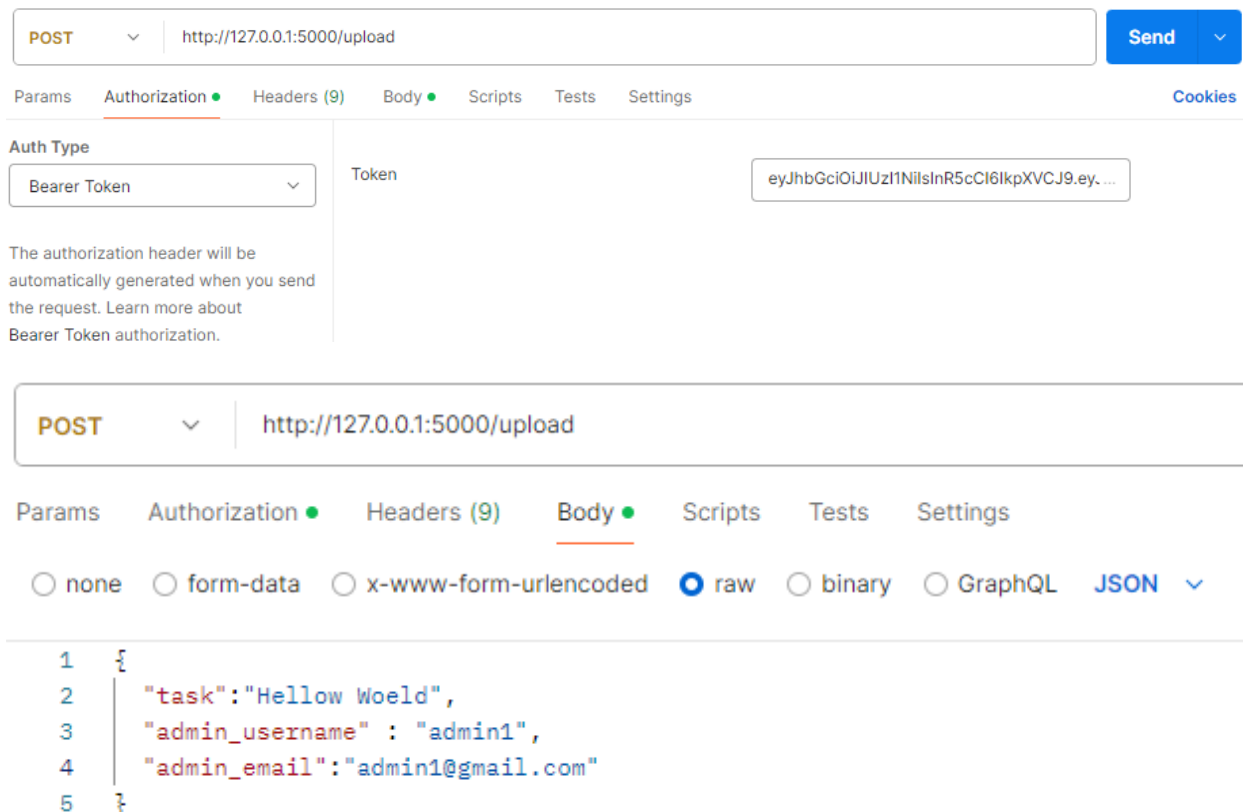
On Failure: 401 status code with an error message.



POST /upload - Upload an assignment.

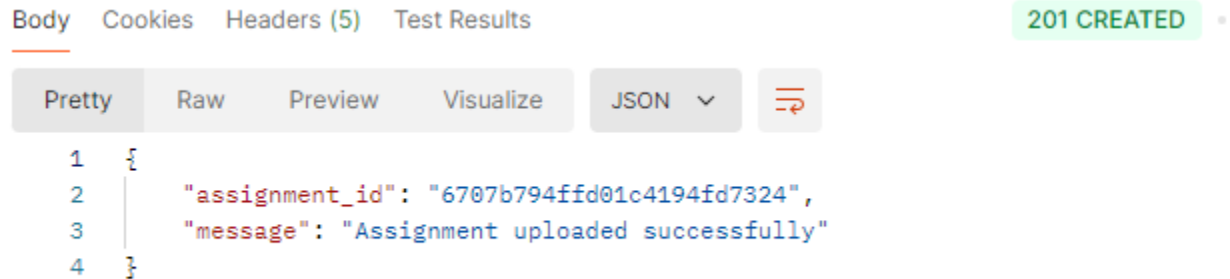
Request Headers:

Authorization: Bearer <JWT Access Token>



Response:

On Success: 201 status code with success message.



The screenshot shows a REST client interface with tabs for Body, Cookies, Headers (5), and Test Results. The Body tab is selected. The response status is 201 CREATED. The response body is displayed in JSON format, showing an assignment ID and a success message.

```
1 {
2   "assignment_id": "6707b794ffd01c4194fd7324",
3   "message": "Assignment uploaded successfully"
4 }
```

GET /admins - Fetch all registered admins.

Request Headers:

Authorization: Bearer <JWT Access Token>




The screenshot shows a REST client interface with tabs for Body, Cookies, Headers (5), and Test Results. The Body tab is selected. The response status is 200 OK. The response body is displayed in JSON format, showing a list of admins with their usernames.



```
1 {
2   "admins": [
3     {
4       "username": "Dev"
5     },
6     {
7       "username": "Nani"
8     }
9   ]
}
```


Admin Endpoints

POST - /admin/register - Register a new admin.

Request Body:

POST  http://127.0.0.1:5000/admin/register

Params Authorization  Headers (9) **Body ** Scripts Tests Settings


☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON **

```
1  {
2    "email": "admin1@gmail.com",
3    "username": "admin1",
4    "password": "123"
5  }
```

Response:

On Success: 201 status code with a success message.



Body Cookies Headers (5) Test Results **201 CREATED**

Pretty Raw Preview Visualize JSON  

```
1  {
2    "message": "Admin registered successfully"
3  }
```

On Duplicate Admin: 400 status code with error message.

Body Cookies Headers (5) Test Results **409 CONFLICT** = 78 ms = 21

Pretty Raw Preview Visualize JSON  

```
1  {
2    "message": "An admin with this email already exists. Please use a different email."
3  }
```


GET /admin/assignments - View assignments tagged to the admin.

Request Headers:

Authorization: Bearer <JWT Access Token>

Response:

```
Body Cookies Headers (5) Test Results 200 OK
Pretty Raw Preview Visualize JSON ↕
9      },
10     {
11       "id": "6707ce463833b7771416b75c",
12       "status": "pending",
13       "submitted_at": "Thu, 10 Oct 2024 18:23:26 GMT",
14       "task": "Hellow Woeld",
15       "userId": "devendra@gmail.com"
16     }
17   ]
18 }
```

POST /assignments/<assignment_id>/accept - Accept an assignment.

Request Headers:

Authorization: Bearer <JWT Access Token>

Response:

```
Body Cookies Headers (5) Test Results 200 OK
Pretty Raw Preview Visualize JSON ↕
1  {
2  |   "message": "Assignment accepted successfully"
3  }
```

POST /assignments/<assignment_id>/reject - Reject an assignment.

Request Headers:

Authorization: Bearer <JWT Access Token>



Final Notes

This backend system for the Assignment Submission Portal includes all essential features, such as user and admin management, JWT authentication, assignment uploads, and status tracking.

Key Features Recap:

User Registration & Login: Secure registration and authentication using hashed passwords and JWT tokens.

Admin Registration & Login: Separate admin roles with secure authentication.

Assignment Submission: Users can upload assignments and assign them to specific admins.

Assignment Management for Admins: Admins can view, accept, or reject assignments.

Assignment Status Tracking: Users can check the status of their assignments (pending, accepted, or rejected).

Additional Notes:

Ensure that the MongoDB server is running during the development or production phases of the project.

Result: Backend system for the Assignment Submission Portal is Ready!

Thank You

Name: Devendra Kumar Pudi