

# API Coding Challenge - Task Management

## API testing using POSTMAN:

Output:

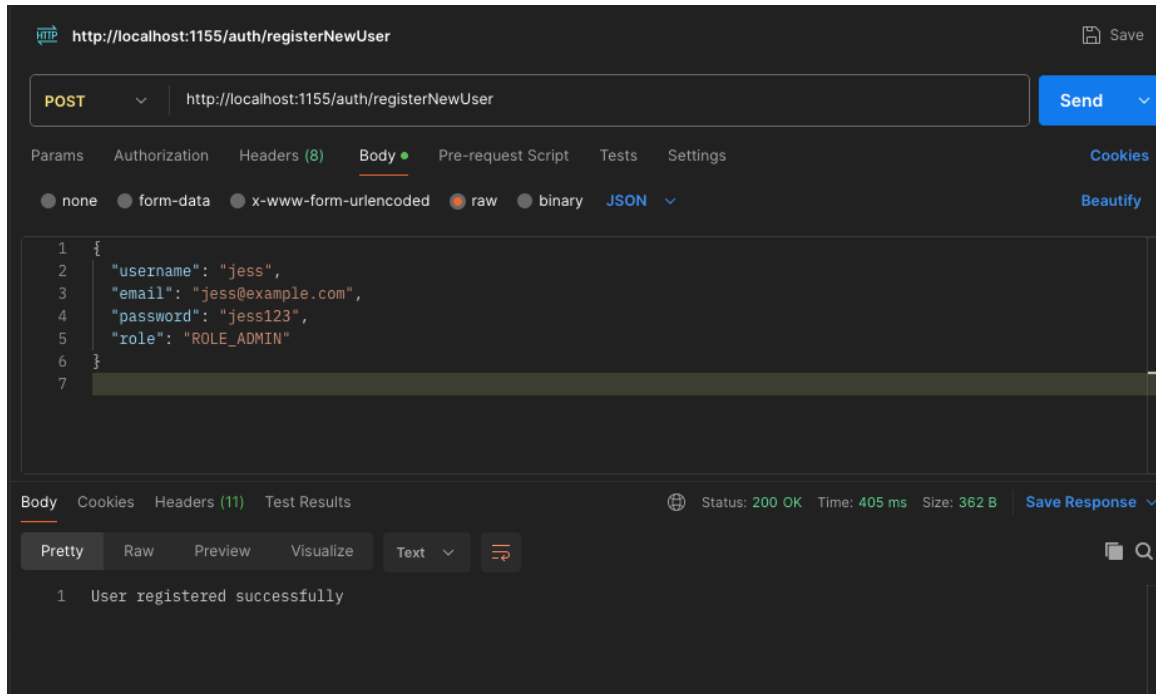
### USER

1. Register User:

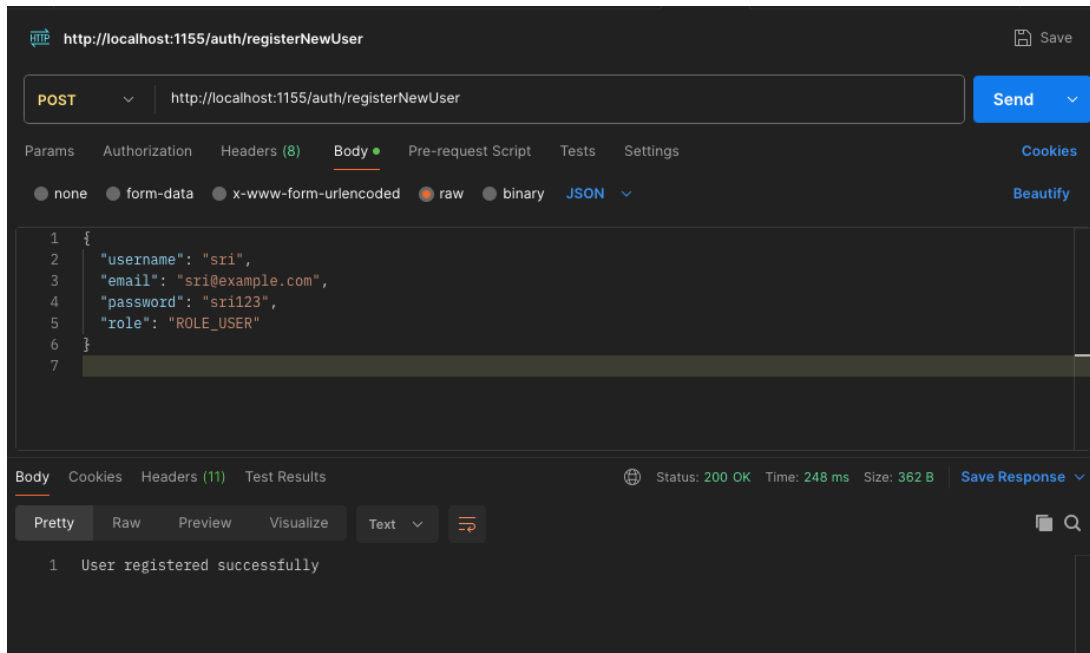
POST

<http://localhost:1155/auth/registerNewUser>

Admin role:



User role:

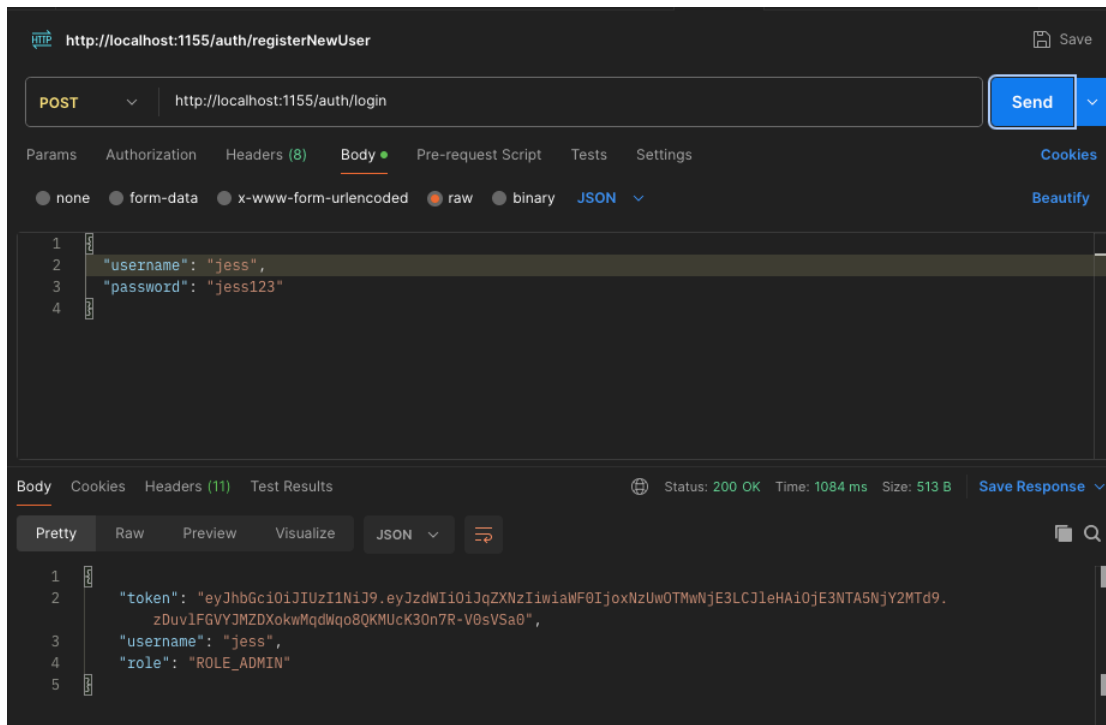


## 2. Login to Get JWT Token

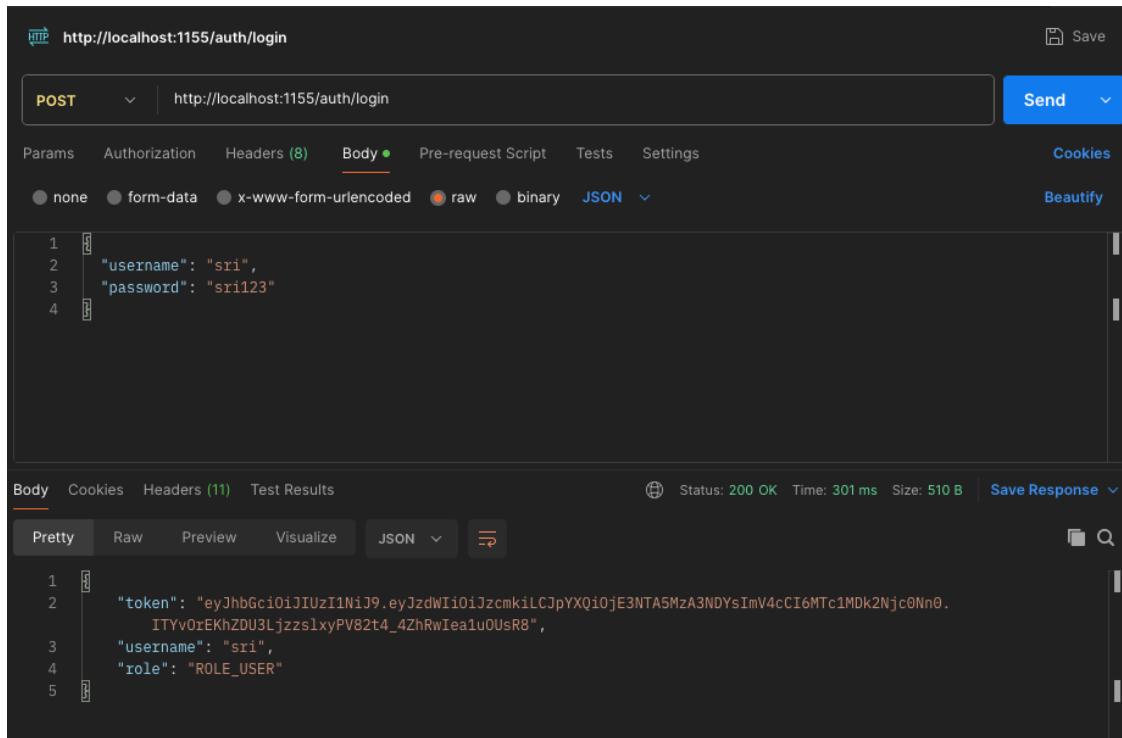
POST

<http://localhost:1155/auth/login>

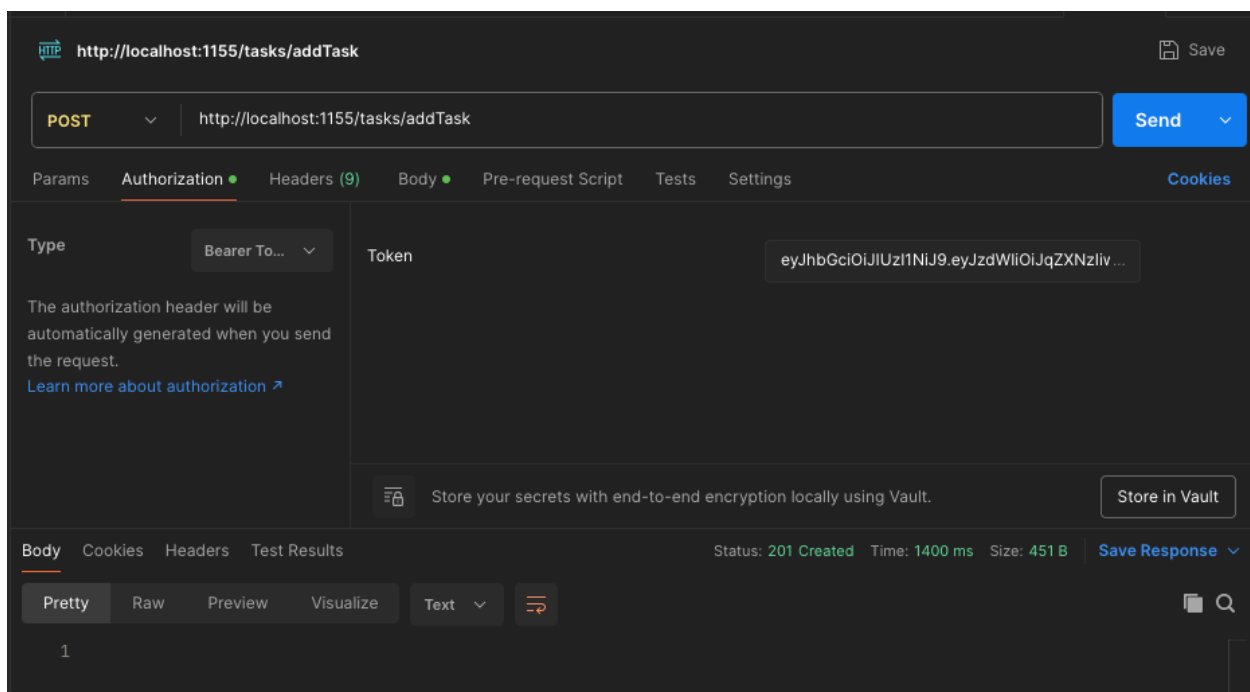
Admin:



User:



3. Add Authorization Header – Token:  
( for testing user and admin roles access)



## TASKS

4. Add a new task (Admin only access):

POST

<http://localhost:1155/tasks/addTask>

Admin:

A screenshot of the Postman application showing a successful POST request. The URL bar displays `http://localhost:1155/tasks/addTask`. The request method is set to **POST**. The request body is a JSON object: 

```
{  "title": "Task3",  "description": "testing",  "dueDate": "2025-06-27",  "priority": "HIGH",  "status": "PENDING"}
```

. The response status is **201 Created**, with a time of **1656 ms** and a size of **451 B**. The response body is a JSON object: 

```
{  "taskId": 7,  "title": "Task3",  "description": "testing",  "dueDate": "2025-06-27",  "priority": "HIGH",  "status": "PENDING"}
```

User:

A screenshot of the Postman application showing a failed POST request. The URL bar displays `http://localhost:1155/tasks/addTask`. The request method is set to **POST**. The request body is a JSON object: 

```
{  "title": "Task4",  "description": "ui implementation",  "dueDate": "2025-07-05",  "priority": "HIGH",  "status": "PENDING"}
```

. The response status is **500 Internal Server Error**, with a time of **47 ms** and a size of **349 B**. The response body is a JSON object: 

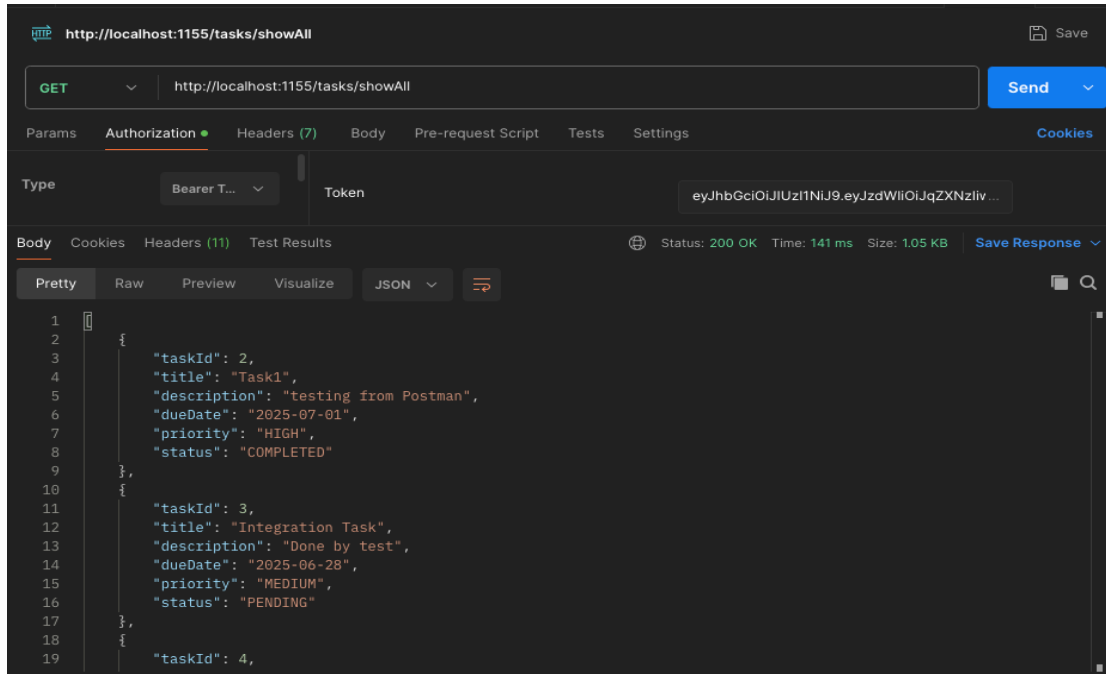
```
{  "error": "Access Denied"}
```

## 5. Retrieve all Tasks (Admin or User access):

POST

<http://localhost:1155/tasks/showAll>

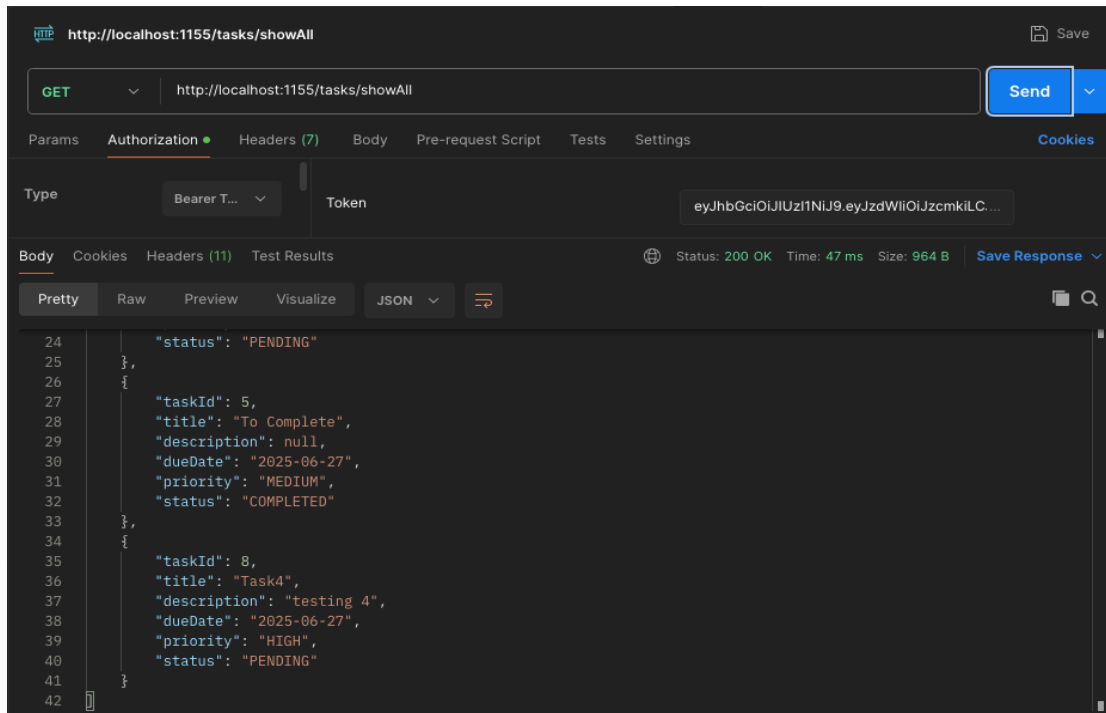
Admin:



A screenshot of the Postman application showing a GET request to `http://localhost:1155/tasks/showAll`. The request is successful with a status of 200 OK. The response body is a JSON array of task objects. The interface shows the 'Authorization' tab with a Bearer token and the 'Body' tab with a JSON response.

```
1 {
2   {
3     "taskId": 2,
4     "title": "Task1",
5     "description": "testing from Postman",
6     "dueDate": "2025-07-01",
7     "priority": "HIGH",
8     "status": "COMPLETED"
9   },
10  {
11    "taskId": 3,
12    "title": "Integration Task",
13    "description": "Done by test",
14    "dueDate": "2025-06-28",
15    "priority": "MEDIUM",
16    "status": "PENDING"
17  },
18  {
19    "taskId": 4,
```

User:



A screenshot of the Postman application showing a GET request to `http://localhost:1155/tasks/showAll`. The request is successful with a status of 200 OK. The response body is a JSON array of task objects. The interface shows the 'Authorization' tab with a Bearer token and the 'Body' tab with a JSON response.

```
24   "status": "PENDING"
25   },
26   {
27     "taskId": 5,
28     "title": "To Complete",
29     "description": null,
30     "dueDate": "2025-06-27",
31     "priority": "MEDIUM",
32     "status": "COMPLETED"
33   },
34   {
35     "taskId": 8,
36     "title": "Task4",
37     "description": "testing 4",
38     "dueDate": "2025-06-27",
39     "priority": "HIGH",
40     "status": "PENDING"
41   }
42 }
```

## 6. Retrieve a single task by its ID (Admin or User access):

GET

<http://localhost:1155/tasks/getById/7>

Admin:

The screenshot shows the Postman Admin interface. The URL bar contains `http://localhost:1155/tasks/getById/7` and the method is set to `GET`. The `Send` button is highlighted. Below the URL bar, the `Body` tab is selected, showing a JSON response in the `Pretty` view:

```
1 {
2   "taskId": 7,
3   "title": "Task3",
4   "description": "testing",
5   "dueDate": "2025-06-27",
6   "priority": "HIGH",
7   "status": "PENDING"
8 }
```

The status bar at the bottom indicates `Status: 200 OK`, `Time: 358 ms`, and `Size: 446 B`. The `Save Response` button is visible.

User:

The screenshot shows the Postman User interface. The URL bar contains `http://localhost:1155/tasks/getById/8` and the method is set to `GET`. The `Send` button is highlighted. Below the URL bar, the `Authorization` tab is selected, showing `Bearer To...` as the type and a token in the `Token` field:

```
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmkiLC...
```

The status bar at the bottom indicates `Status: 200 OK`, `Time: 50 ms`, and `Size: 448 B`. The `Save Response` button is visible.

## 7. Update an existing task (Admin only access):

PUT

<http://localhost:1155/tasks/updateTask/7>

Admin:

The screenshot shows a REST client interface with the URL `http://localhost:1155/tasks/updateTask/1` (though the body refers to task 7). The method is `PUT`. The request body is a JSON object:

```
{
  "title": "Task3",
  "description": "update testing",
  "dueDate": "2025-07-01",
  "priority": "MEDIUM",
  "status": "IN_PROGRESS"
}
```

The response status is `200 OK` with a time of `222 ms` and size of `459 B`. The response body is a JSON object:

```
{
  "taskId": 7,
  "title": "Task3",
  "description": "update testing",
  "dueDate": "2025-07-01",
  "priority": "MEDIUM",
  "status": "IN_PROGRESS"
}
```

User:

The screenshot shows a REST client interface with the URL `http://localhost:1155/tasks/updateTask/8`. The method is `PUT`. The request body is a JSON object:

```
{
  "title": "Task4",
  "description": "update ui",
  "dueDate": "2025-07-08",
  "priority": "LOW",
  "status": "IN_PROGRESS"
}
```

The response status is `500 Internal Server Error` with a time of `74 ms` and size of `349 B`. The response body is a JSON object:

```
{
  "error": "Access Denied"
}
```

## 8. Update an existing Task Status (User or Admin access):

PATCH

<http://localhost:1155/tasks/updateTask/7/status?status=COMPLETED>

Admin:

The screenshot shows a Postman interface for a PATCH request to `http://localhost:1155/tasks/updateTask/7/status?status=COMPLETED`. The request is configured with the following settings:

- Method: PATCH
- URL: `http://localhost:1155/tasks/updateTask/7/status?status=COMPLETED`
- Body: raw (JSON)

The response is displayed in the "Body" tab, showing a JSON object with the following details:

```
1 {
2   "taskId": 7,
3   "title": "Task3",
4   "description": "update testing",
5   "dueDate": "2025-07-01",
6   "priority": "MEDIUM",
7   "status": "COMPLETED"
8 }
```

Response status: 200 OK, Time: 187 ms, Size: 457 B.

User:

The screenshot shows a Postman interface for a PATCH request to `http://localhost:1155/tasks/updateTask/8/status?status=COMPLETED`. The request is configured with the following settings:

- Method: PATCH
- URL: `http://localhost:1155/tasks/updateTask/8/status?status=COMPLETED`
- Authorization: Bearer Token
- Token: `eyJhbGciOiJIUzI1NiJ9.eyJzdWwiOiJzcmkiLC...`

The response is displayed in the "Body" tab, showing a JSON object with the following details:

```
1 {
2   "taskId": 8,
3   "title": "Task4",
4   "description": "testing 4",
5   "dueDate": "2025-06-27",
6   "priority": "HIGH",
7   "status": "COMPLETED"
8 }
```

Response status: 200 OK, Time: 95 ms, Size: 450 B.



## 9. Delete a task by its ID (Admin only access):

### DELETE

<http://localhost:1155/tasks/deleteTask/7>

Admin:

The screenshot shows a Postman interface for a DELETE request to `http://localhost:1155/tasks/deleteTask/1`. The request is configured with the method `DELETE` and the URL `http://localhost:1155/tasks/deleteTask/7`. The Authorization tab is selected, showing a Bearer token: `eyJhbGciOiJIUzI1NiJ9.eyJzdWl0IjJqZXNzliv...`. The response status is `200 OK` with a time of `284 ms` and a size of `346 B`. The response body is `1 Task deleted`.

User:

The screenshot shows a Postman interface for a DELETE request to `http://localhost:1155/tasks/deleteTask/8`. The request is configured with the method `DELETE` and the URL `http://localhost:1155/tasks/deleteTask/8`. The Authorization tab is selected, showing a Bearer token: `eyJhbGciOiJIUzI1NiJ9.eyJzdWl0IjJcmkiLC...`. The response status is `500 Internal Server Error` with a time of `66 ms` and a size of `349 B`. The response body is `{ "error": "Access Denied" }`.