

**FACULTY OF ENGINEERING DESIGN AND TECHNOLOGY**

**NAME: MUNJWOK JAMES ALALA**

**REG NO: M22B23/007**

**ACCESS NO: A98246**

**COURSE: BACHELOR OF SCIENCE IN COMPUTER SCIENCE (BSCS).**

**COURSE UNIT: SOFTWARE CONSTRUCTION**

**LECTURER: SIMON LUBAMBO**

This is to outline the challenges encountered while implementing a **CRUD API** using Django REST Framework and the solutions applied to resolve them. **Postman** was used to test and validate all the endpoints.

The API supports the following operations:

1. **POST** /tasks/ → Create a task
2. **GET** /tasks/ → Retrieve all tasks
3. **GET** /tasks/{id}/ → Retrieve a task by ID
4. **PUT** /tasks/{id}/ → Update a task
5. **DELETE** /tasks/{id}/ → Delete a task

**Challenges and Solutions**

1. **CORS Errors – React Couldn’t Fetch API Data**

The frontend (localhost:3000) was blocked from accessing the backend (localhost:8000) due to CORS restrictions. The solution was to install django-cors-headers and updated **settings.py:**

INSTALLED\_APPS = ['corsheaders', 'rest\_framework', 'tasks']

MIDDLEWARE = ['corsheaders.middleware.CorsMiddleware'] + MIDDLEWARE

CORS\_ALLOW\_ALL\_ORIGINS = True

1. **"400 Bad Request" on POST Requests**

Sending data in Postman resulted in a 400 Bad Request error. It was solved by ensuring all required fields were included in serializers.py:

class TaskSerializer(serializers.ModelSerializer):

class Meta:

model = Task

fields = '\_\_all\_\_'

The API successfully created tasks after fixing missing fields.

1. **"404 Not Found" on GET by ID Requests**

Fetching a task using /tasks/{id}/ returned 404 Not Found. The solution was to verify if the **Task ID existed** before making the request by checking /tasks/.

1. **"405 Method Not Allowed" on PUT & DELETE Requests**

Updating (PUT) or deleting (DELETE) a task resulted in 405 Method Not Allowed.The TaskDetailView was missing in urls.py. Added:

path('<int:pk>/', TaskDetailView.as\_view(), name='task-detail'),

Updates and deletions worked properly in Postman.

1. **Database Migration Errors**

Running python manage.py migrate failed due to missing migrations. The solution was to deleted old migrations, recreated them, and applied migrations again:

rm -rf tasks/migrations/

python manage.py makemigrations tasks

python manage.py migrate

**Lessons Learned & Improvements**

1. **Django REST Framework** simplifies API development with class-based views.
2. **Postman Testing** is crucial to validate API responses before integrating with the frontend.
3. **Error Logs & Debugging** (Django logs + Postman responses) helped identify issues quickly.
4. **Future Improvements**: Add **JWT authentication**, **pagination**, and **unit tests** for better API security and performance.

PostMan Screenshots

* 1. GET Request

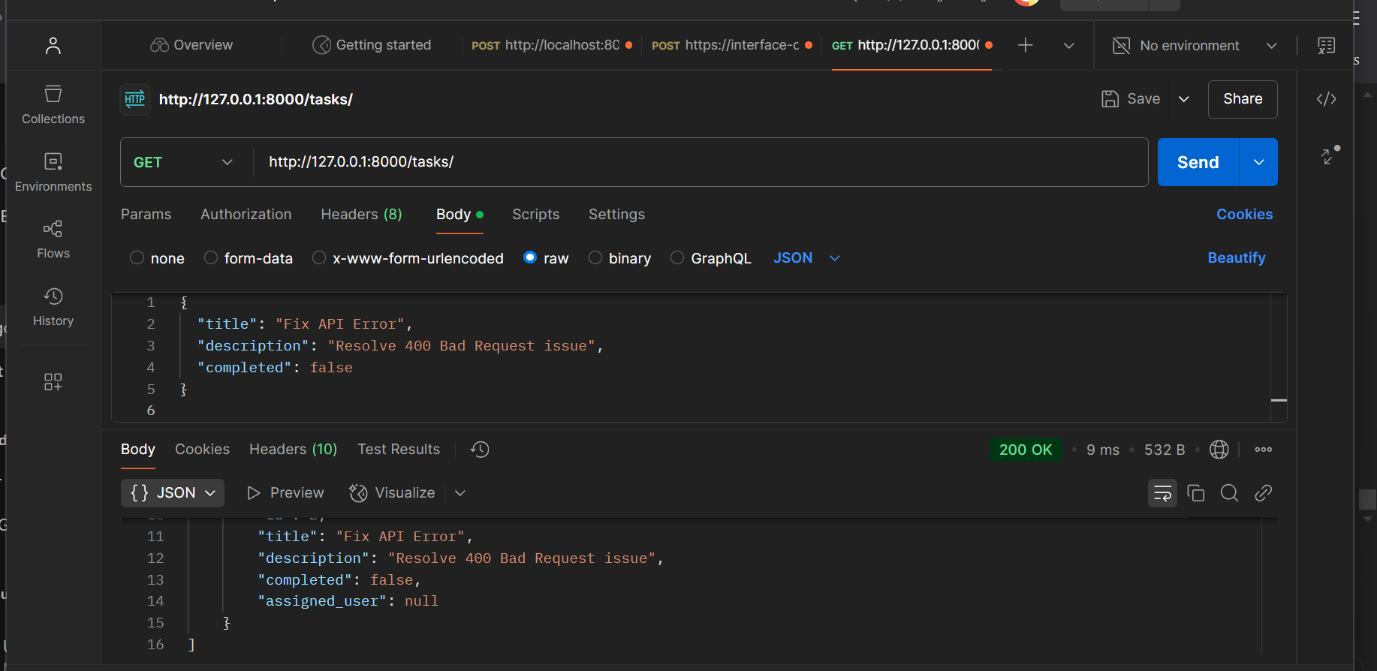


Figure 1:Get Request

* 1. **POST Request**

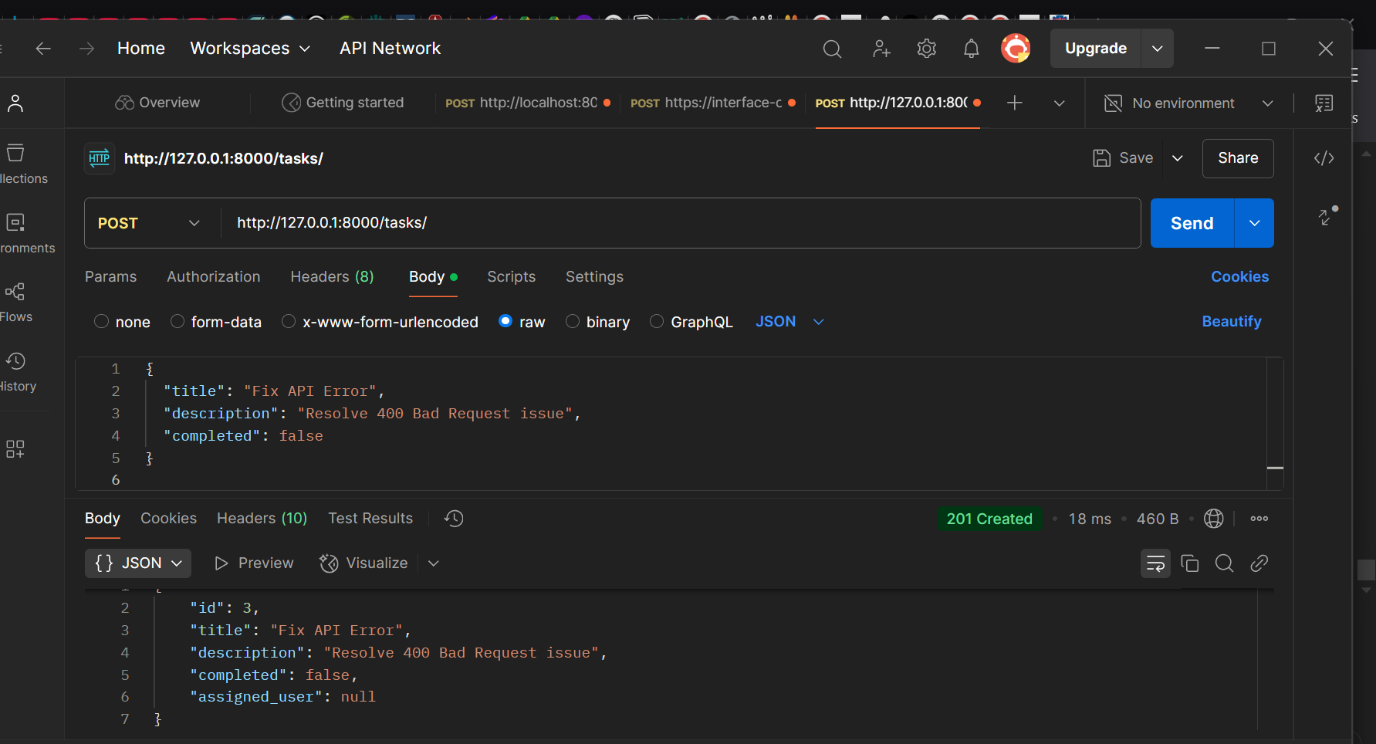
****

Figure 2: Post Request Successful

* 1. GET by ID Request

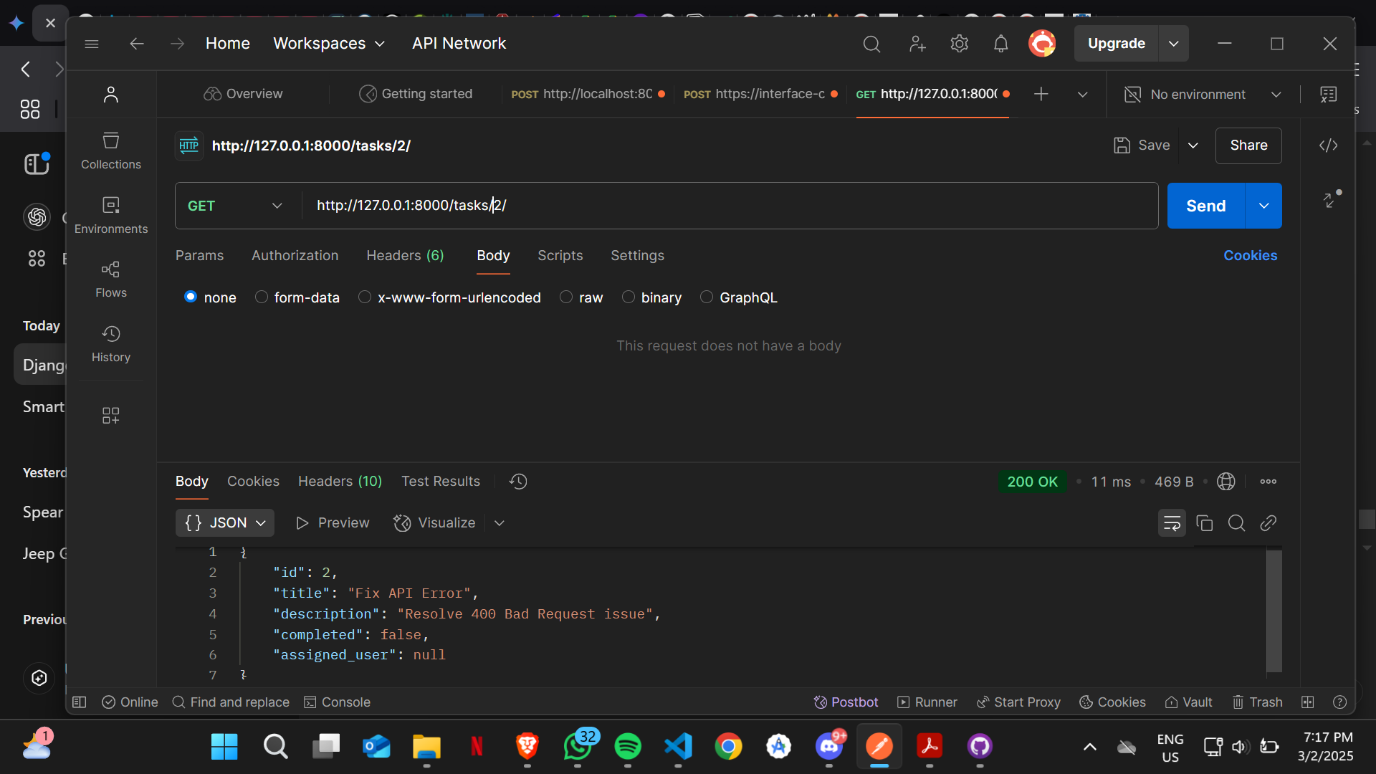


Figure 3: Get by Id

* 1. DELETE by ID Request

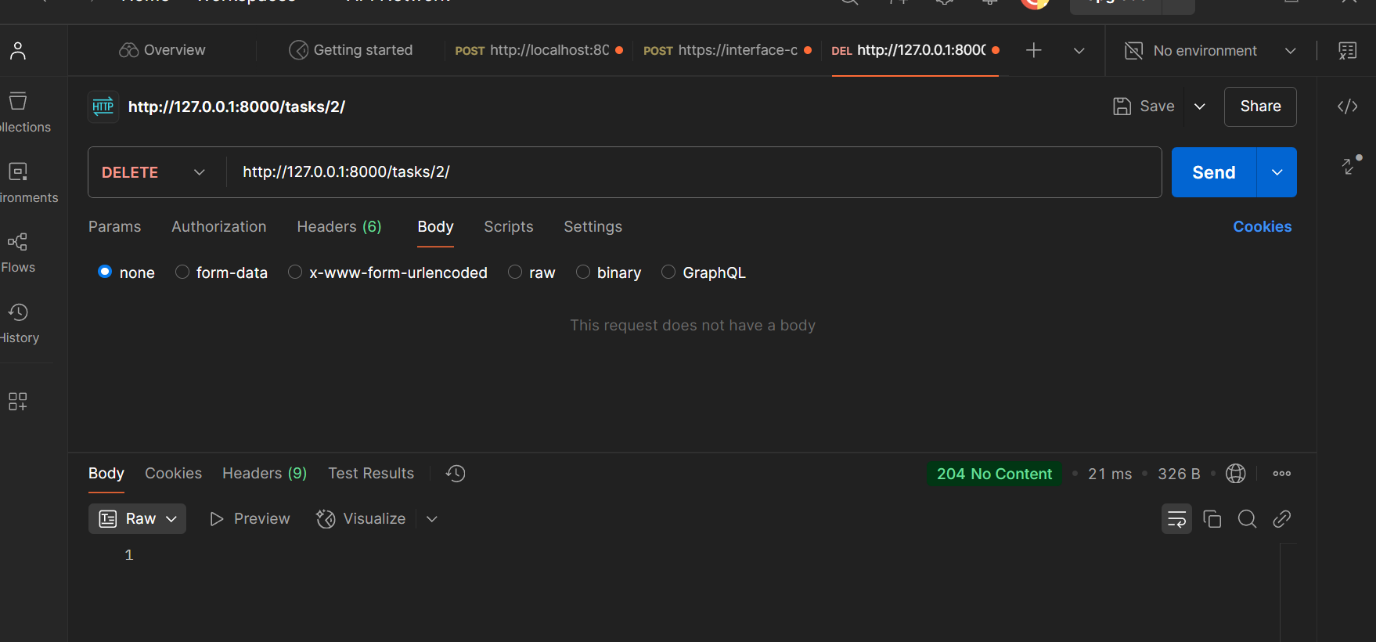


Figure 4: Delete by ID

* 1. **PUT Request**

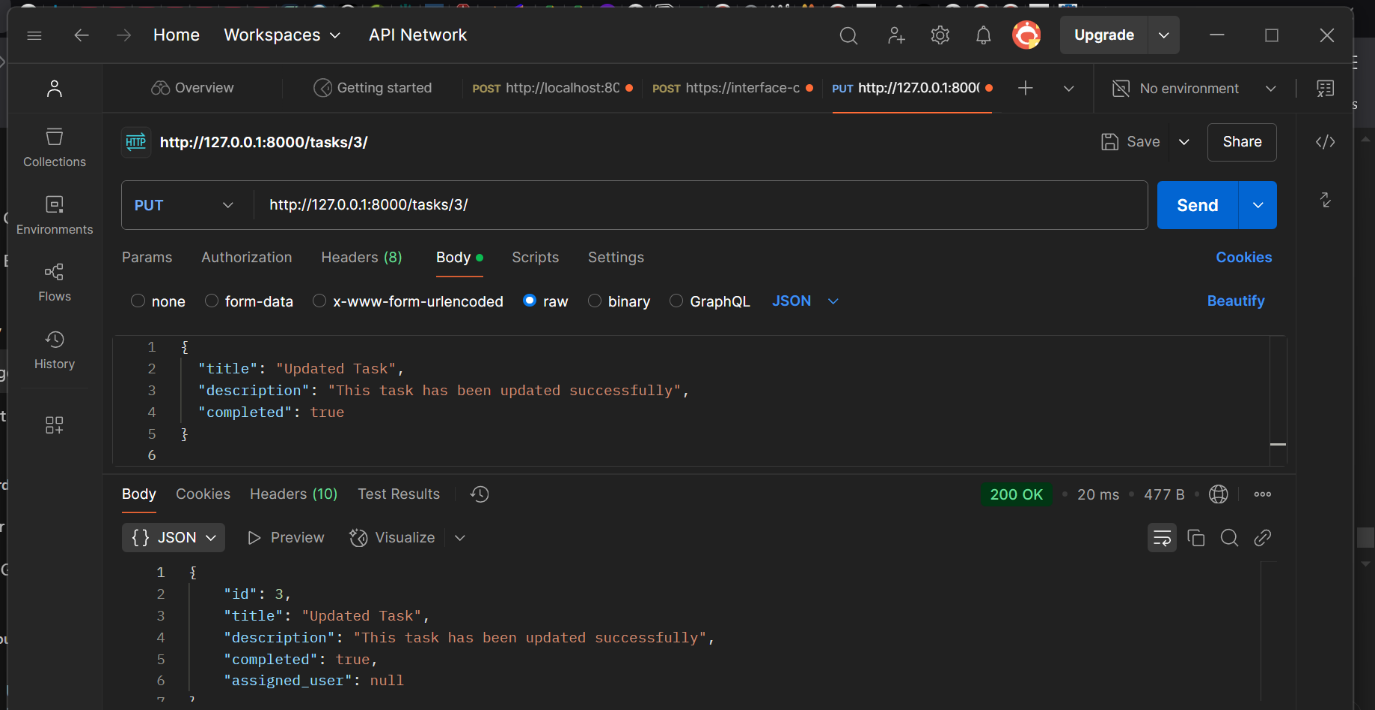
****

Figure 5: Put Request

**GitHub Repository Link**: [GIthub](https://github.com/munji74/Software-Design)