UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

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:

a) **POST** /tasks/ → Create a task

b) **GET** /tasks/ → Retrieve all tasks

c) **GET** /tasks/{id}/ → Retrieve a task by ID

d) **PUT** /tasks/{id}/ → Update a task

e) **DELETE** /tasks/{id}/ → Delete a task

Challenges and Solutions

i) 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
```

ii) "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.

iii) "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/.

iv) "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.

v) 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

a) **Django REST Framework** simplifies API development with class-based views.

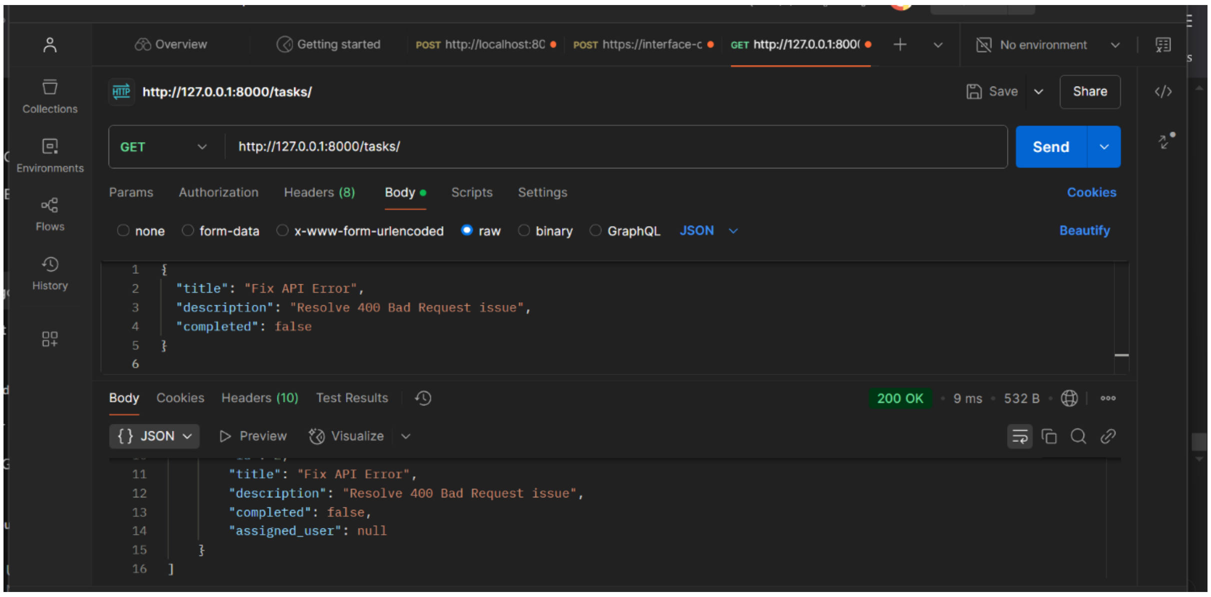
b) **Postman Testing** is crucial to validate API responses before integrating with the frontend.

c) **Error Logs & Debugging** (Django logs + Postman responses) helped identify issues quickly.

d) **Future Improvements**: Add **JWT authentication**, **pagination**, and **unit tests** for better API security and performance.

PostMan Screenshots

1) GET Request



http://127.0.0.1:8000/tasks/

GET http://127.0.0.1:8000/tasks/

Body: raw JSON

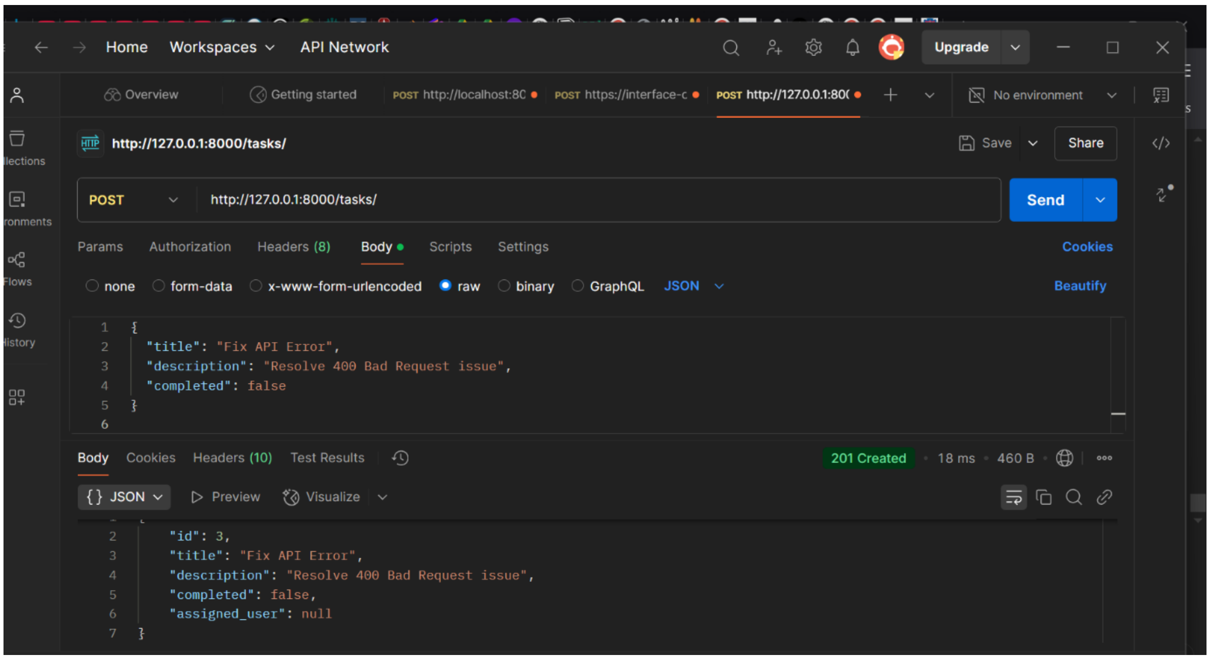
```
{
"title": "Fix API Error",
"description": "Resolve 400 Bad Request issue",
"completed": false
}
```

200 OK · 9 ms · 532 B

```
        "title": "Fix API Error",
        "description": "Resolve 400 Bad Request issue",
        "completed": false,
        "assigned_user": null
    }
]
```

Figure 1:Get Request

2) **POST Request**



http://127.0.0.1:8000/tasks/

POST http://127.0.0.1:8000/tasks/

Body: raw JSON

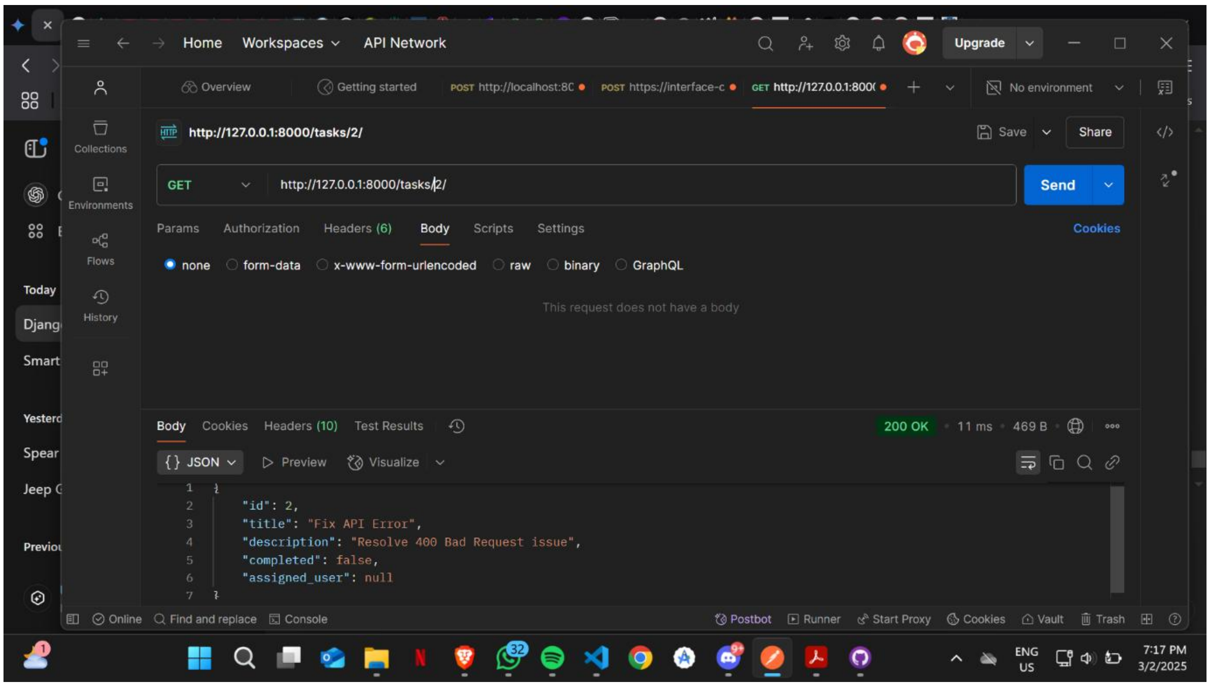
```
{
"title": "Fix API Error",
"description": "Resolve 400 Bad Request issue",
"completed": false
}
```

201 Created · 18 ms · 460 B

```
    "id": 3,
    "title": "Fix API Error",
    "description": "Resolve 400 Bad Request issue",
    "completed": false,
    "assigned_user": null
}
```

Figure 2: Post Request Successful

3) GET by ID Request



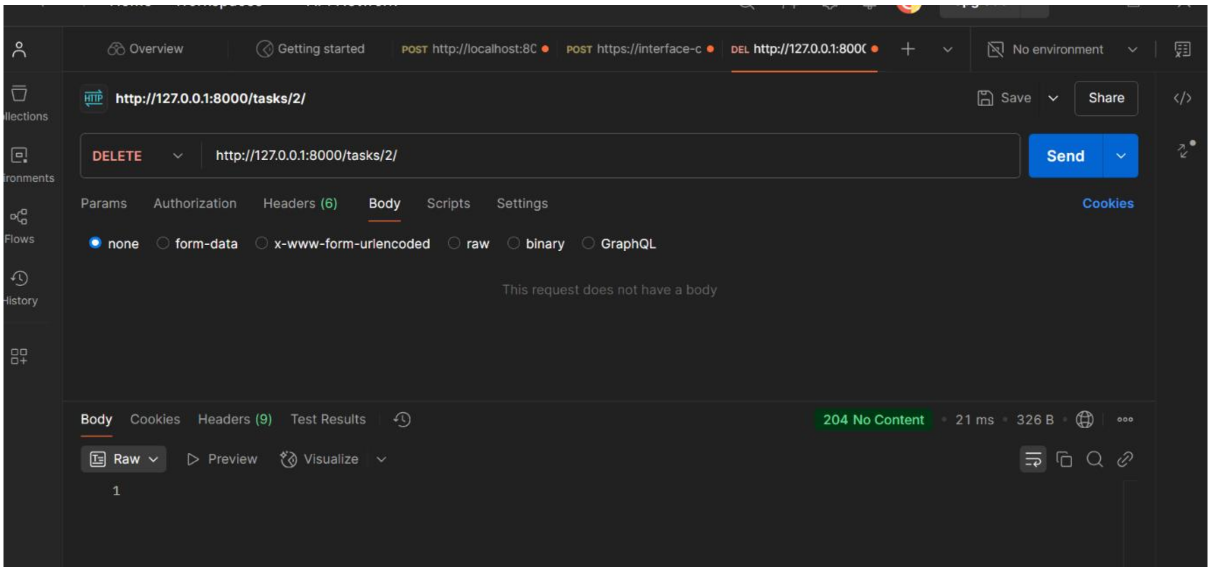
GET http://127.0.0.1:8000/tasks/2/

200 OK · 11 ms · 469 B

```
{
    "id": 2,
    "title": "Fix API Error",
    "description": "Resolve 400 Bad Request issue",
    "completed": false,
    "assigned_user": null
}
```

Figure 3: Get by Id

4) DELETE by ID Request

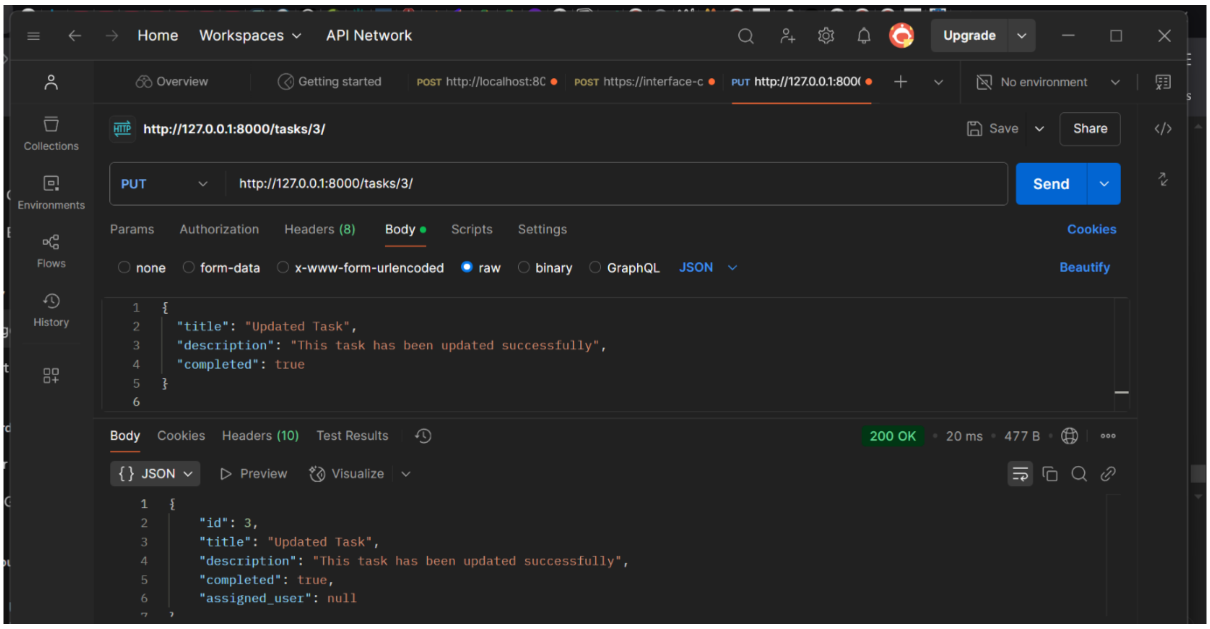


DELETE http://127.0.0.1:8000/tasks/2/

204 No Content · 21 ms · 326 B

Figure 4: Delete by ID

5) PUT Request



http://127.0.0.1:8000/tasks/3/

PUT http://127.0.0.1:8000/tasks/3/

```
{
  "title": "Updated Task",
  "description": "This task has been updated successfully",
  "completed": true
}
```

```
{
    "id": 3,
    "title": "Updated Task",
    "description": "This task has been updated successfully",
    "completed": true,
    "assigned_user": null
}
```

Figure 5: Put Request

GitHub Repository Link: GIthub