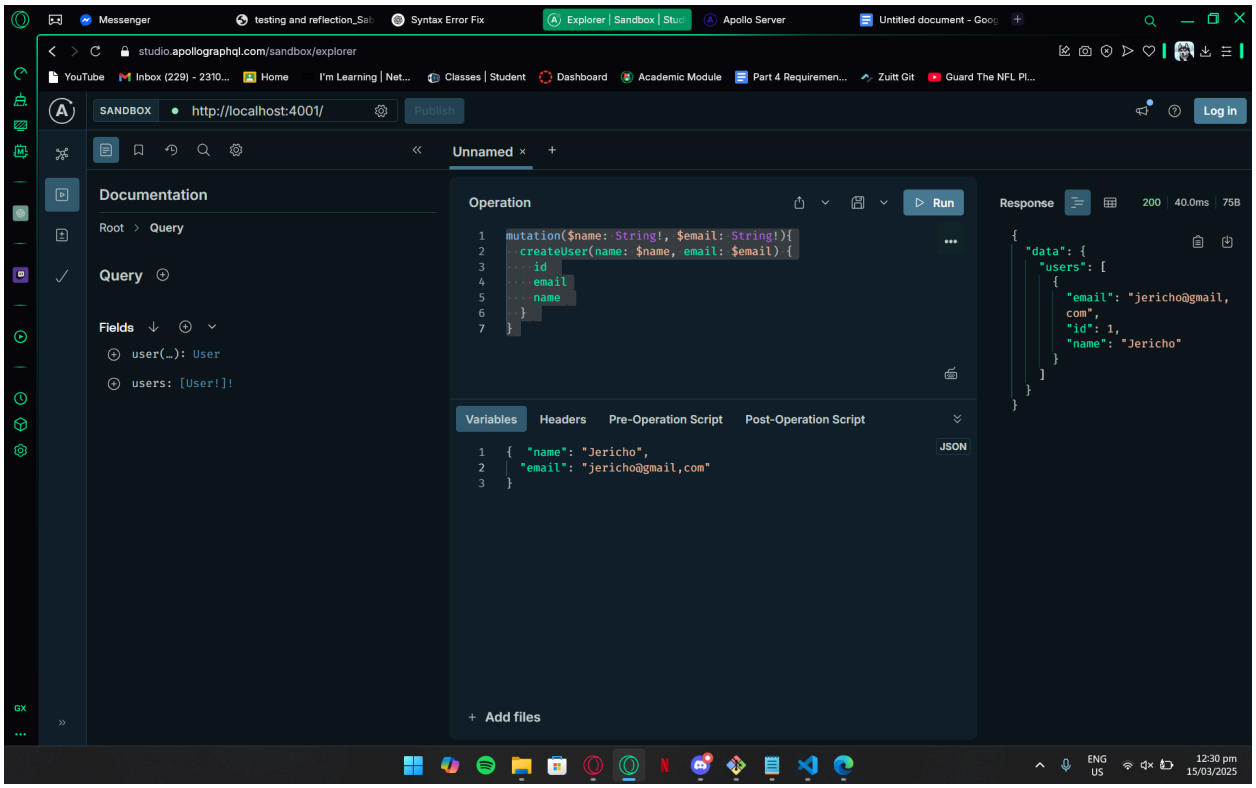
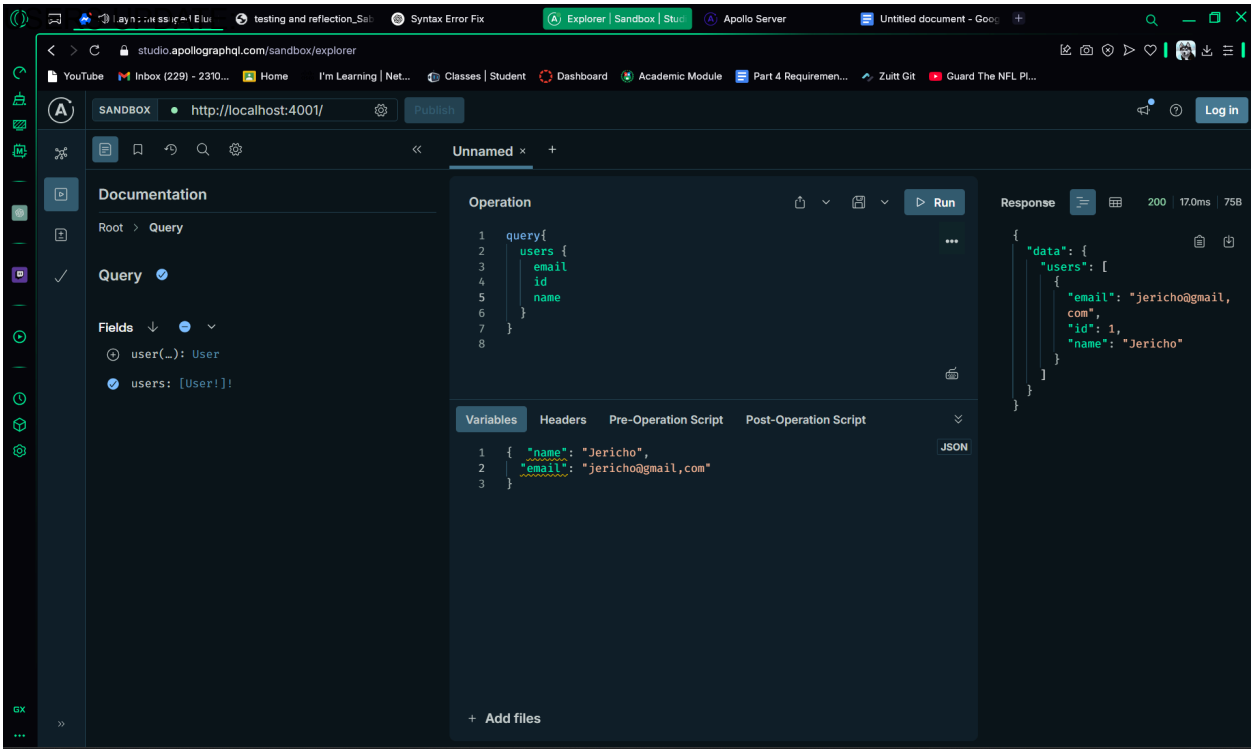
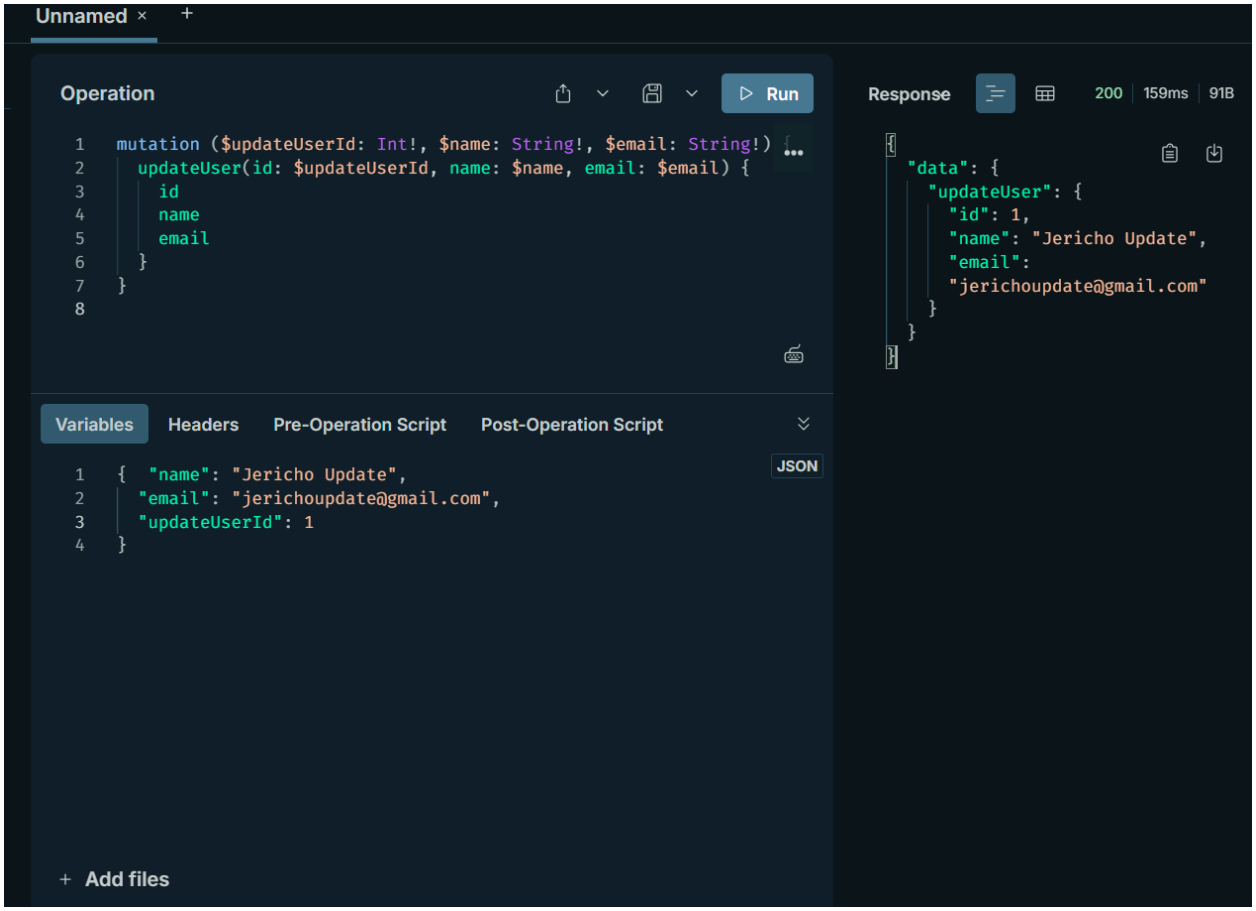
USERS CREATE:

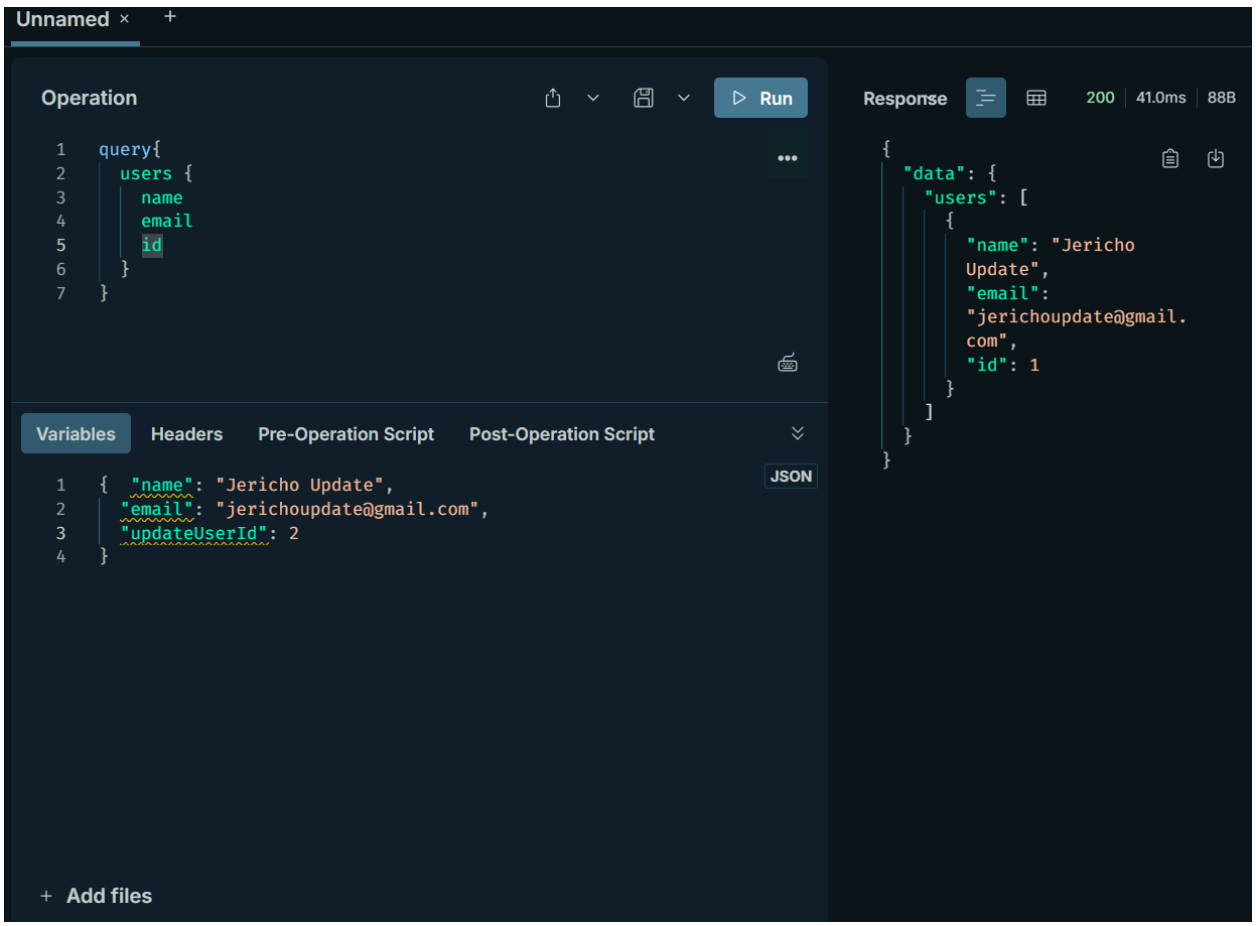


USERS READ:

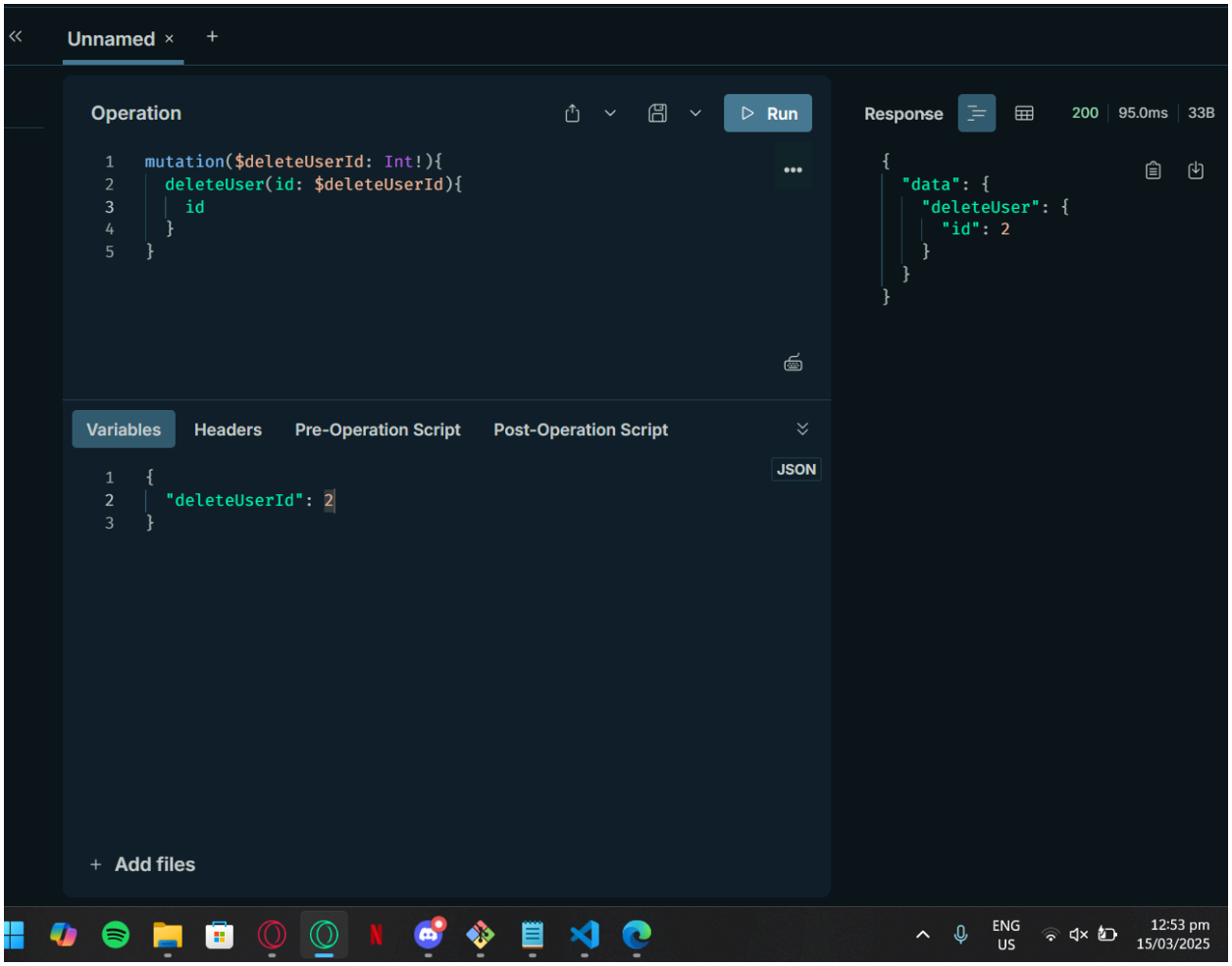


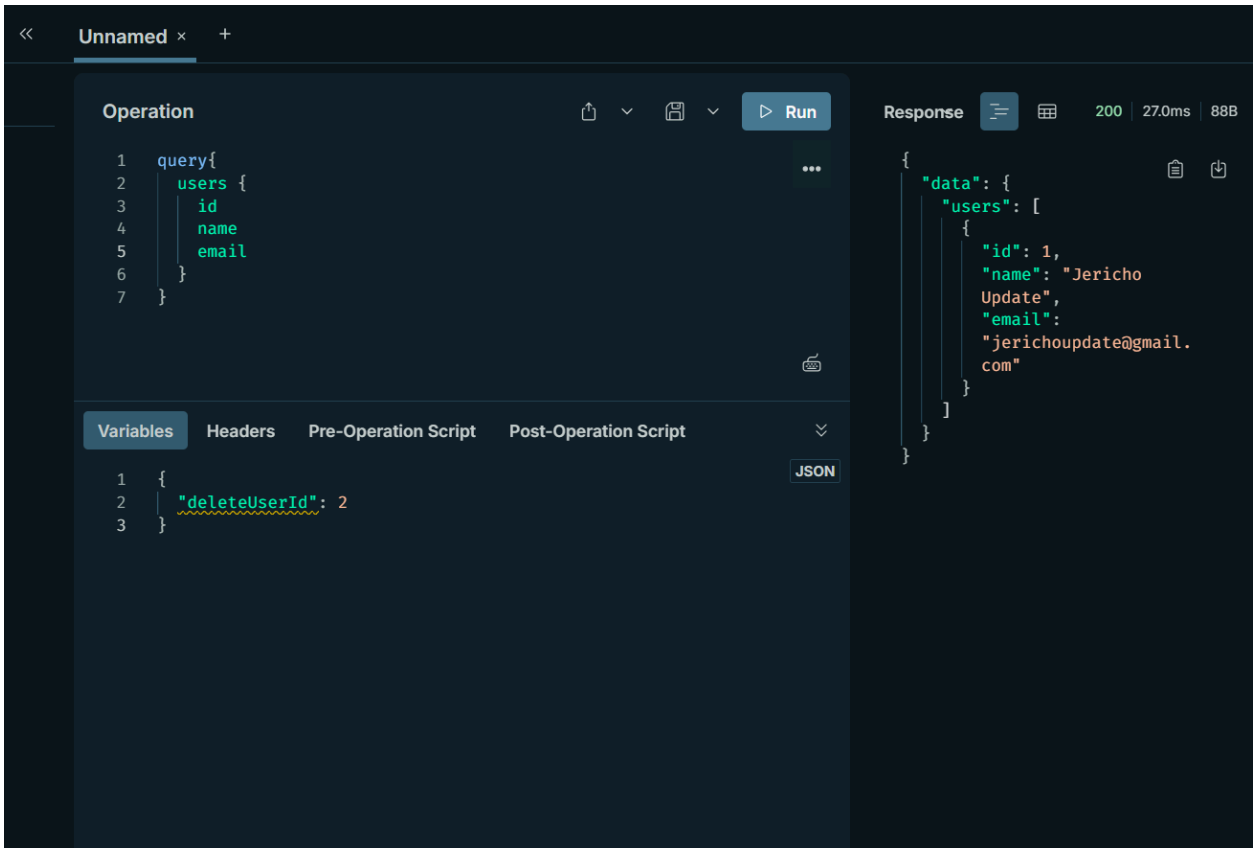
USERS UPDATE:



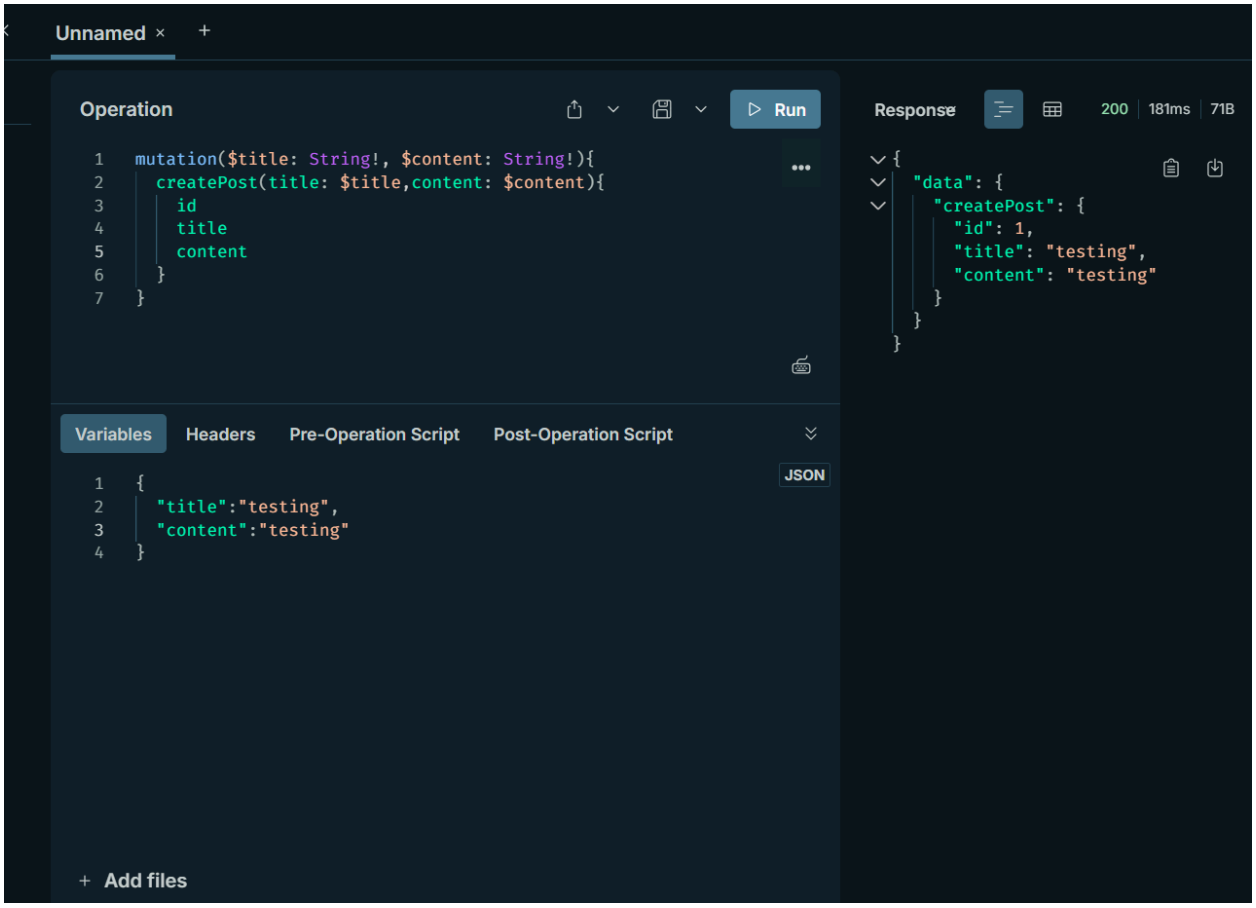


USERS DELETE:

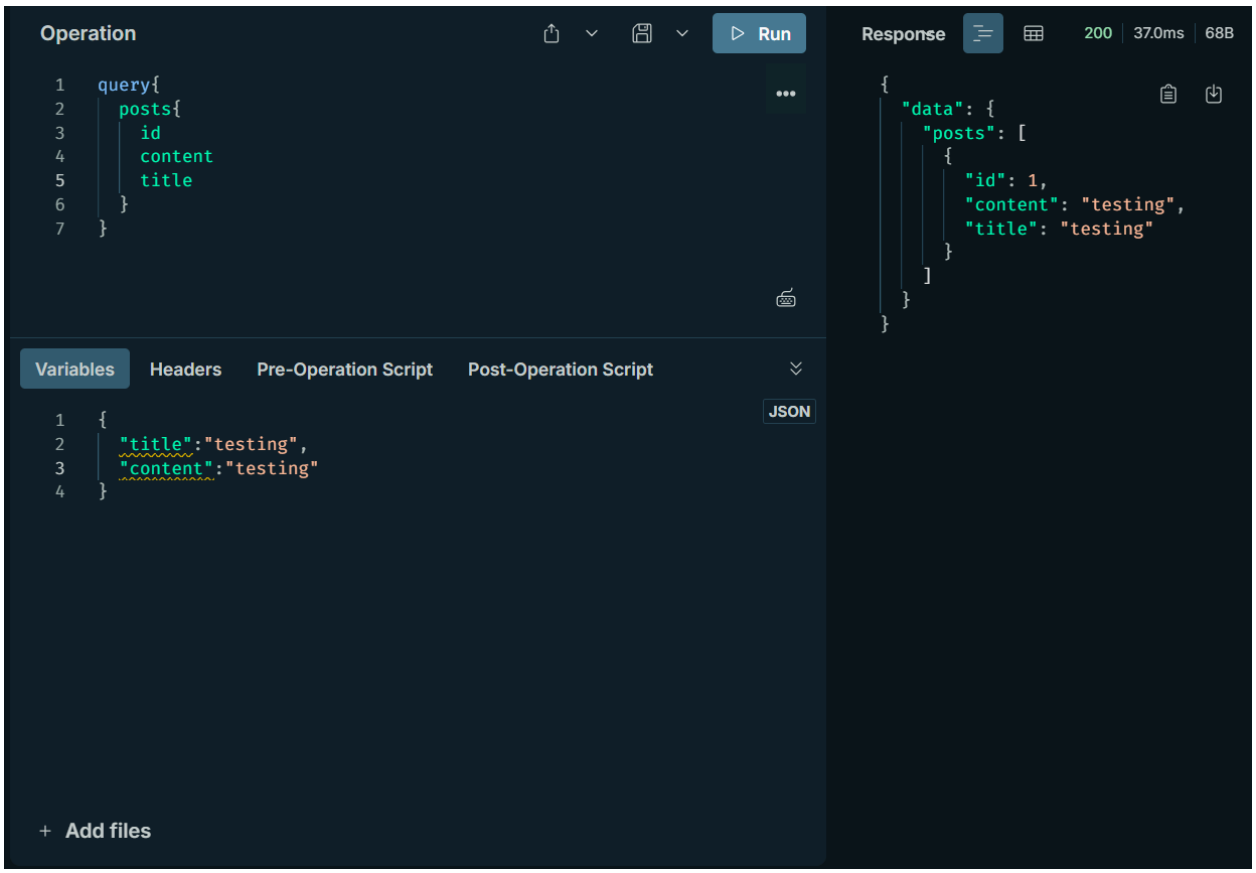




POST CREATE:



POST READ:



POST UPDATE:

Operation

Run

```
mutation($updatePostId: Int!, $title: String!, $content: String!){
  updatePost(id: $updatePostId, title: $title, content: $content){
    id
    title
    content
  }
}
```

Variables | Headers | Pre-Operation Script | Post-Operation Script

JSON

```
{
  "title":"testing success",
  "content":"testing success",
  "updatePostId": 1
}
```

+ Add files

Response 200 49.0ms 87

```
{
  "data": {
    "updatePost": {
      "id": 1,
      "title": "testing
      success",
      "content": "testing
      success"
    }
  }
}
```

Operation

Run

```
query{
  posts{
    content
    id
    title
  }
}
```

Variables | Headers | Pre-Operation Script | Post-Operation Script

JSON

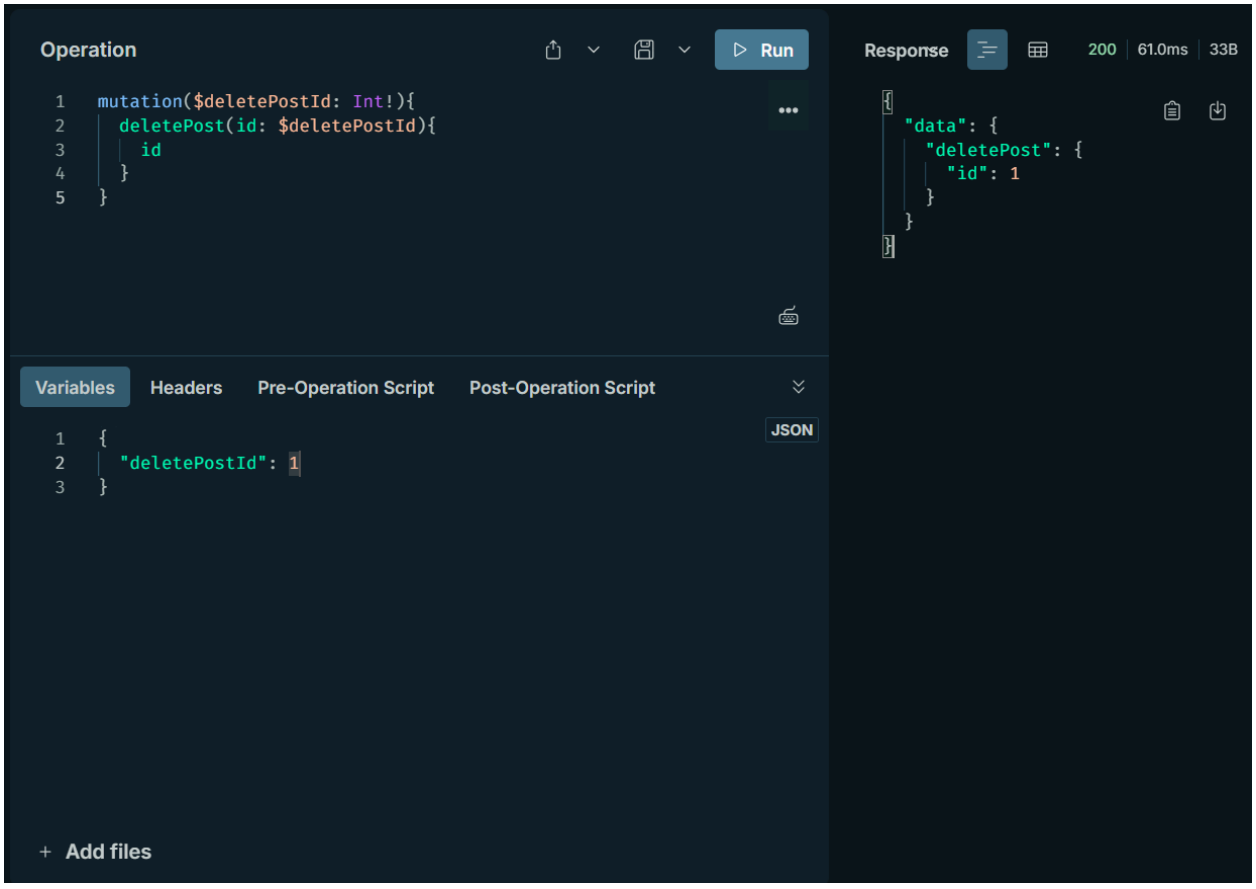
```
{
  "title":"testing success",
  "content":"testing success",
  "updatePostId": 1
}
```

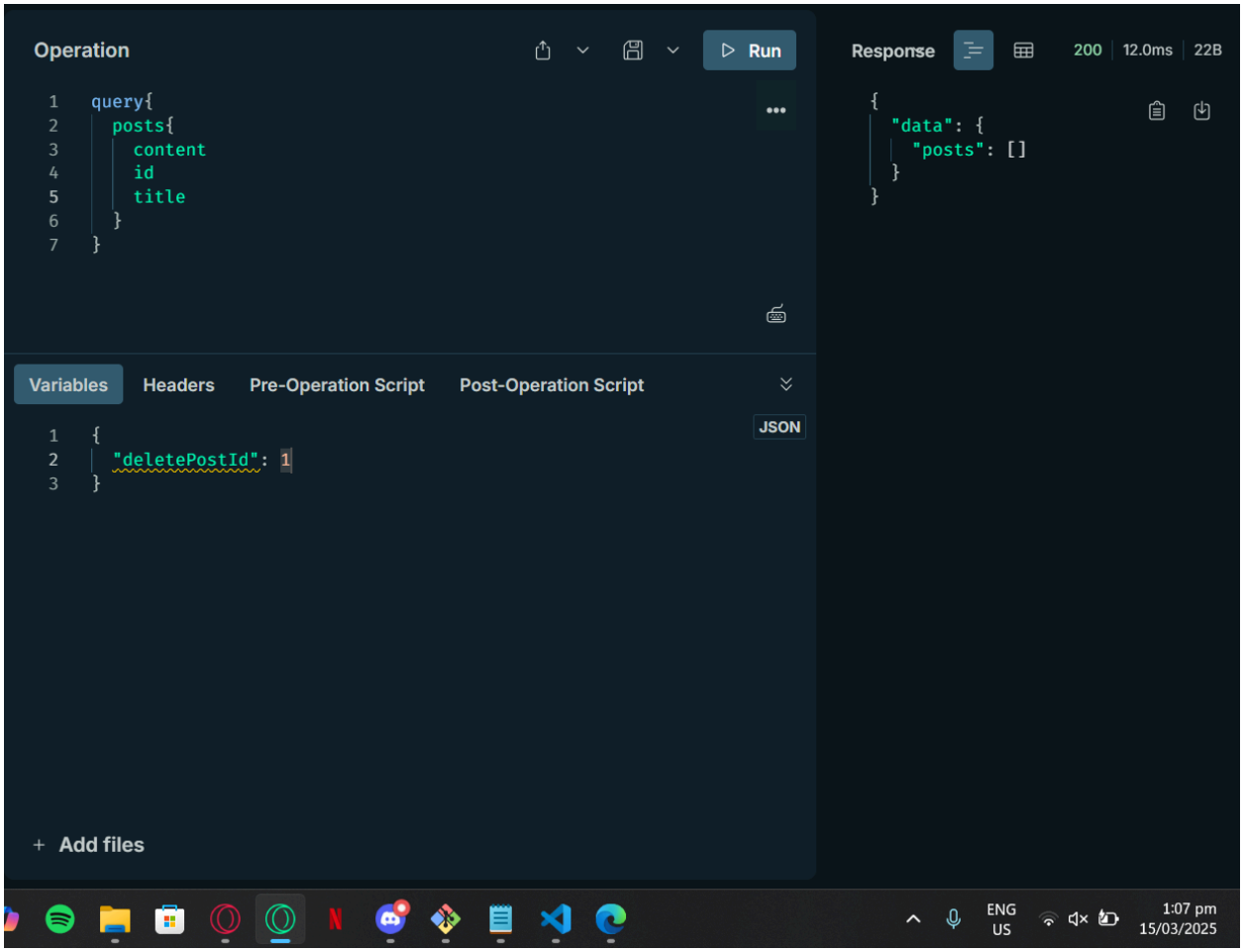
+ Add files

Response 200 29.0ms 84

```
{
  "data": {
    "posts": [
      {
        "content": "testing
        success",
        "id": 1,
        "title": "testing
        success"
      }
    ]
  }
}
```

POST DELETE:


Operation
Run
mutation($deletePostId: Int!){
deletePost(id: $deletePostId){
id
}
}
Response
200
61.0ms
33B
{
"data": {
"deletePost": {
"id": 1
}
}
}
Variables
Headers
Pre-Operation Script
Post-Operation Script
JSON
{
"deletePostId": 1
}
Add files


Operation
Run
query{
posts{
content
id
title
}
}
Response
200
12.0ms
22B
{
"data": {
"posts": []
}
}
Variables
Headers
Pre-Operation Script
Post-Operation Script
JSON
{
"deletePostId": 1
}
Add files
ENG
US
1:07 pm
15/03/2025

- What do database migrations do and why are they useful?

As I gathered, database migrations are essentially known as a “snapshot” in that it reflects your database state at that point in time. Whenever there is anything needed to change or add to the database, each migration builds over the previous. Migrations are helpful as it keeps a record of previous versions of each database. This is particularly useful if you ever need to revert to a previous version of the database.

- How does GraphQL differ from REST for CRUD operations?

Unlike RESTful APIs where multiple endpoints are created for each resource, GraphQL exposes a single endpoint, which helps clients to fetch their respective data as per the requirement. REST, on the other hand, is frequently tied to many endpoints which need to be fixed data, causing discrepancies. GraphQL’s type system also enhances client-server communication by helping both sides reach an agreement regarding the structure of the data exchanged.