Problem Statement:

https://docs.google.com/document/d/1F90KvSqxxPzlRyeX7kD3wULrT0xNJMLdlgGVHq4KMxk/

Installation

Download this repository and change into directory of project. Then use pip to install all required pacakages.

pip install -r requirements.txt

Files and Directory Structure

django_rest_kaushik_jadhav: Root folder of project

|---- cms_api : This folder contains our main api

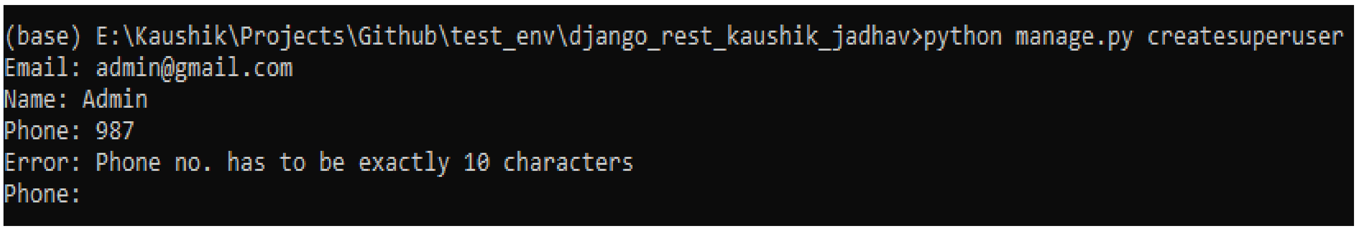
|---- django_rest_kaushik_jadhav: This folder contains essential settings and configuration files of project.

Creating Admin via seeding

To create an Admin user via seeding, run the following command:

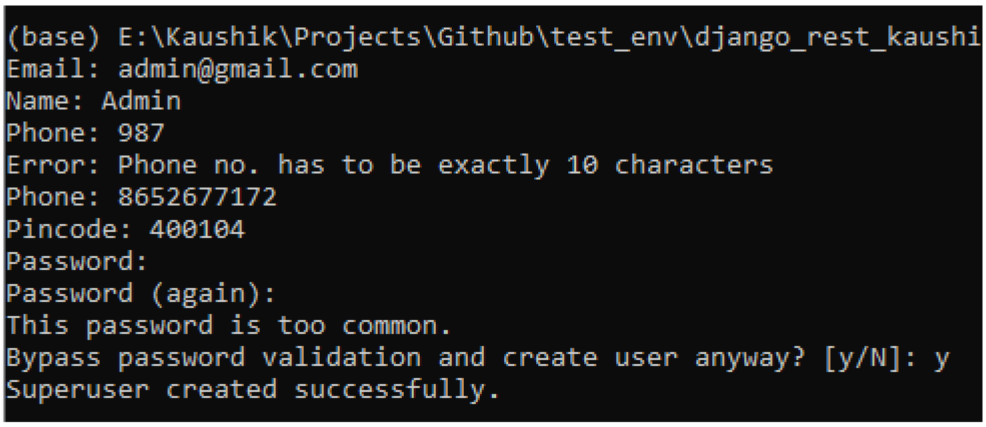
python manage.py createsuperuser

Note that all fields are validated even during admin creation and would throw error if rules mentioned in the problem statement are not followed



```
(base) E:\Kaushik\Projects\Github\test_env\django_rest_kaushik_jadhav>python manage.py createsuperuser
Email: admin@gmail.com
Name: Admin
Phone: 987
Error: Phone no. has to be exactly 10 characters
Phone:
```

Follow all validations for successful admin creation



```
(base) E:\Kaushik\Projects\Github\test_env\django_rest_kaushi
Email: admin@gmail.com
Name: Admin
Phone: 987
Error: Phone no. has to be exactly 10 characters
Phone: 8652677172
Pincode: 400104
Password:
Password (again):
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

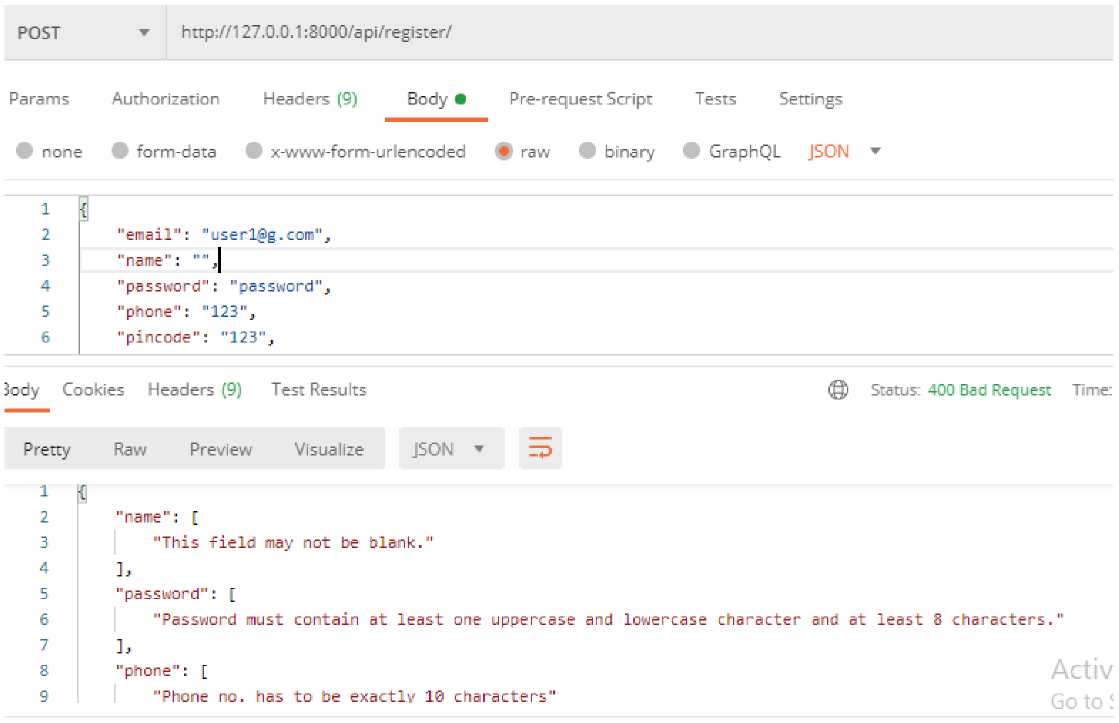
Launching the API

Start the server on localhost by running

python manage.py runserver

Registering New User

Make a POST request to http://domain_name**/api/register/** to register a user. By default, domain_name is **127.0.0.1:8000**. All fields are validated. Invalid fields will throw respective errors as shown below



POST http://127.0.0.1:8000/api/register/

raw JSON

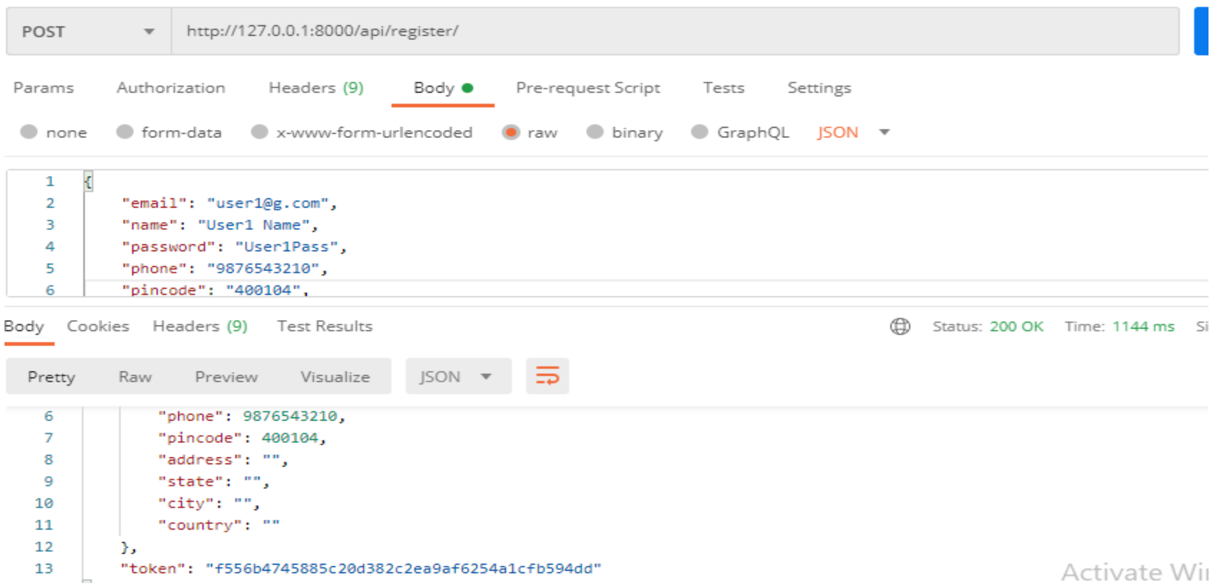
```
{
    "email": "user1@g.com",
    "name": "",
    "password": "password",
    "phone": "123",
    "pincode": "123",
```

Status: 400 Bad Request

JSON

```
{
    "name": [
        "This field may not be blank."
    ],
    "password": [
        "Password must contain at least one uppercase and lowercase character and at least 8 characters."
    ],
    "phone": [
        "Phone no. has to be exactly 10 characters"
```

The response to a valid request will contain the user data and a token.



POST http://127.0.0.1:8000/api/register/

raw JSON

```
{
    "email": "user1@g.com",
    "name": "User1 Name",
    "password": "User1Pass",
    "phone": "9876543210",
    "pincode": "400104",
```

Status: 200 OK Time: 1144 ms

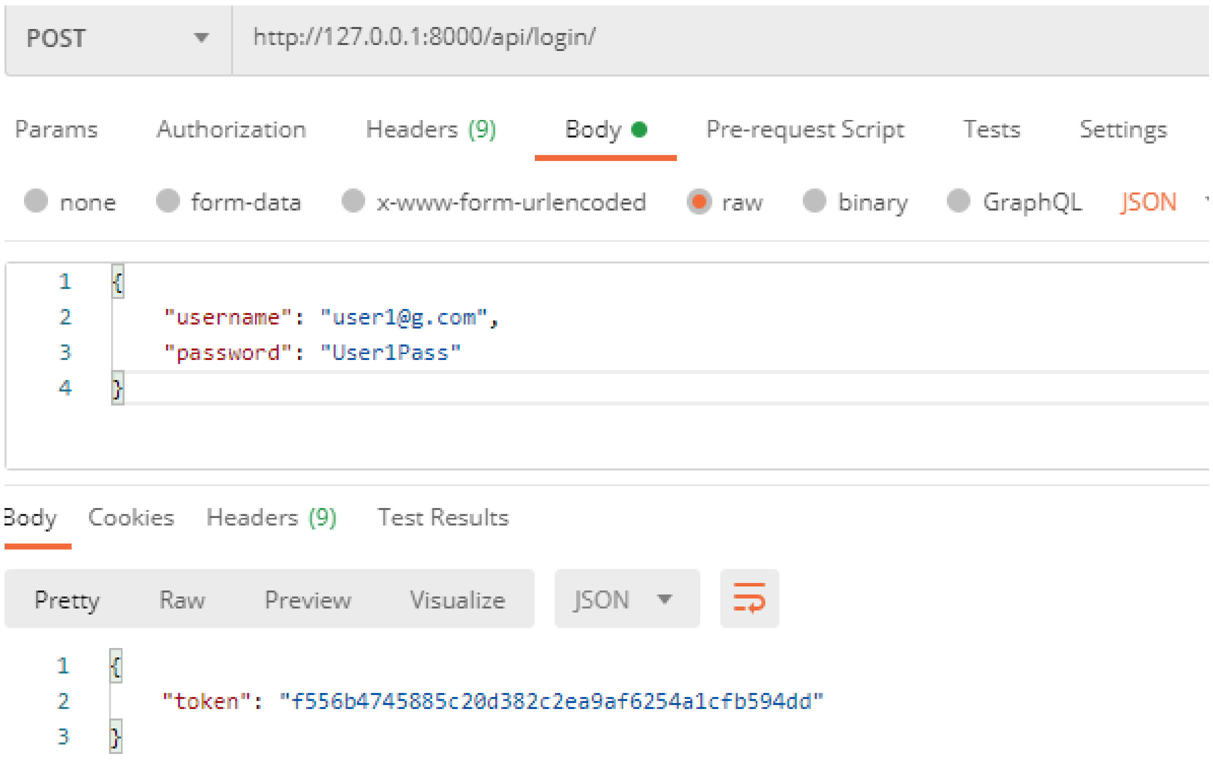
JSON

```
        "phone": 9876543210,
        "pincode": 400104,
        "address": "",
        "state": "",
        "city": "",
        "country": ""
    },
    "token": "f556b4745885c20d382c2ea9af6254a1cfb594dd"
```

Login Existing Users

Existing Users can Log In to get a token. Tokens can be used to access protected routes.

To login, make a POST request to http://domain_name**/api/login**



POST http://127.0.0.1:8000/api/login/

raw JSON

```
{
    "username": "user1@g.com",
    "password": "User1Pass"
}
```

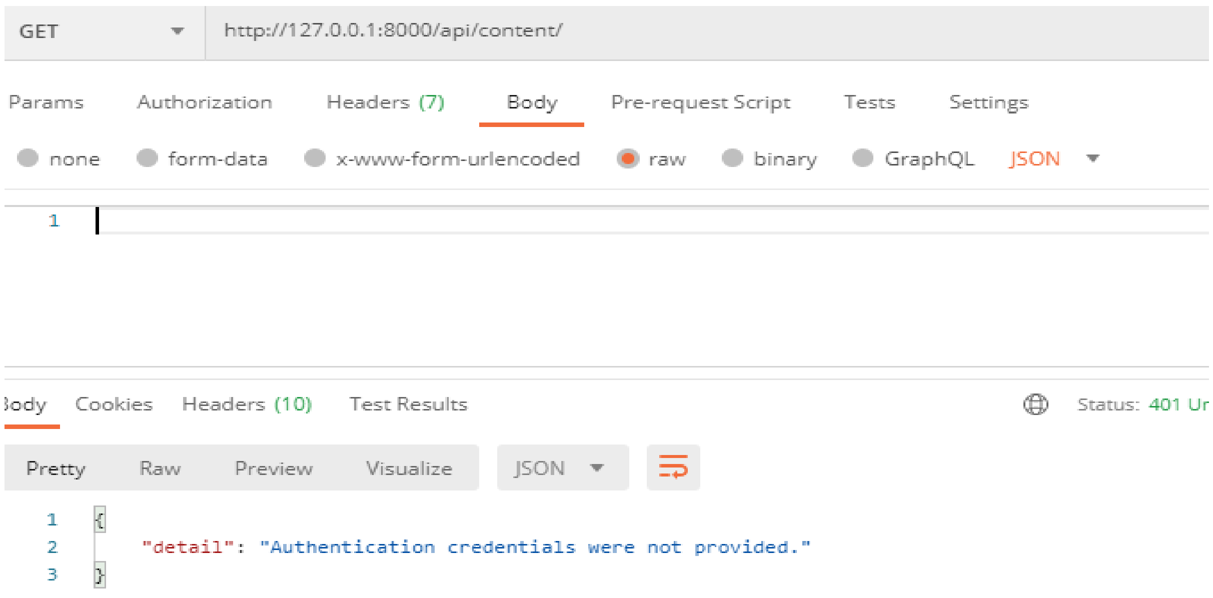
JSON

```
{
    "token": "f556b4745885c20d382c2ea9af6254a1cfb594dd"
}
```

Viewing & Posting Content

Users can View and Create contents on the route http://domain_name**/api/content/**

This is a protected route and any user who is not logged in cannot access it.



GET http://127.0.0.1:8000/api/content/

raw JSON

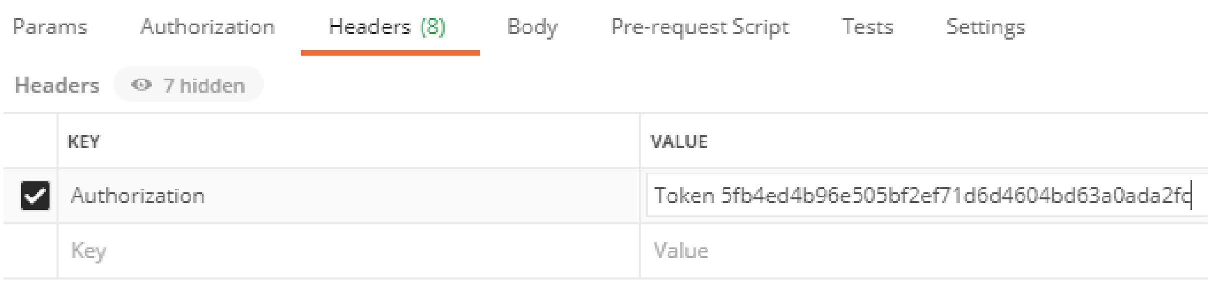
Status: 401 Ur

JSON

```
{
    "detail": "Authentication credentials were not provided."
}
```

As mentioned, users are of two roles: Admin and Author. Authors are regular users who can View, Create, Update and Delete only their own content. Whereas, Admins can View, Create, Update and Delete content of multiple Authors.

Before making any request, user needs to be logged in. To do this, the token of the user needs to be specified in the HTTP headers in the following format:

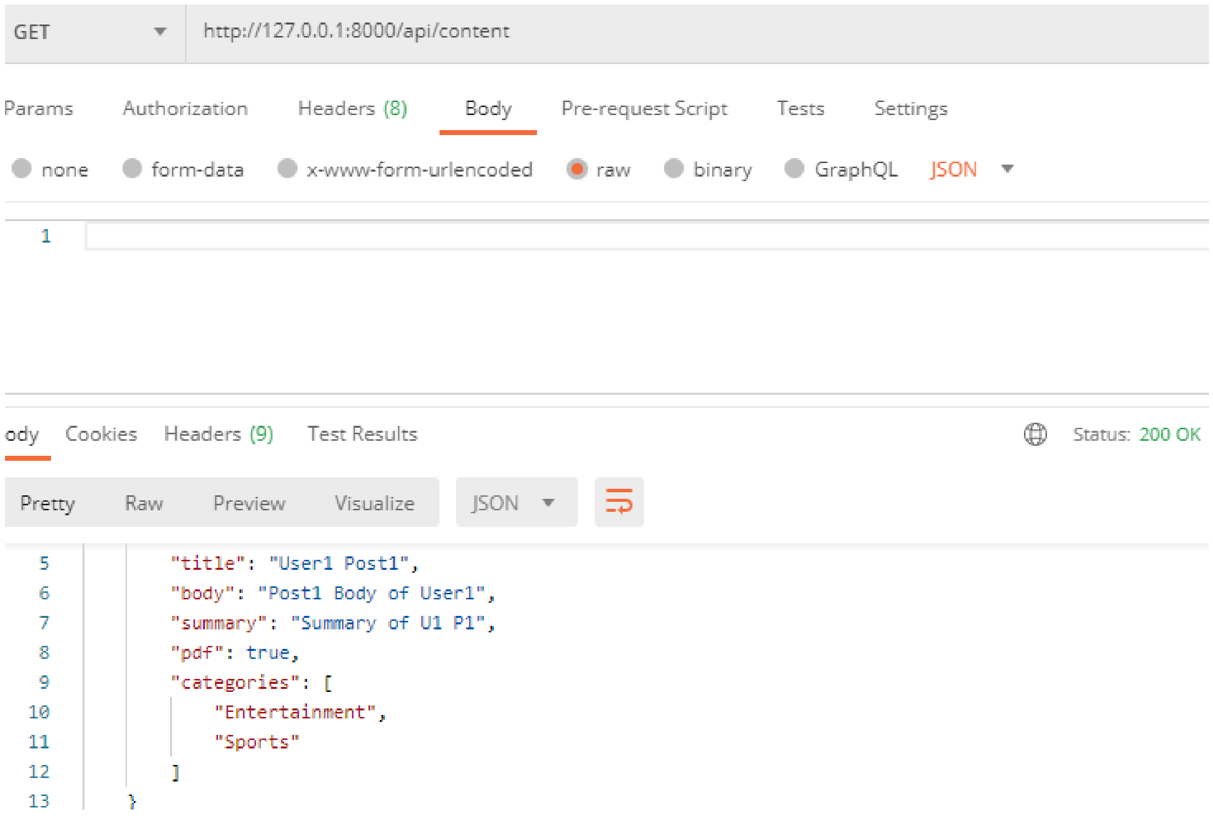


	KEY	VALUE
✓	Authorization	Token 5fb4ed4b96e505bf2ef71d6d4604bd63a0ada2fc
	Key	Value

Viewing Content

View content by making a GET request to http://domain_name**/api/content/**

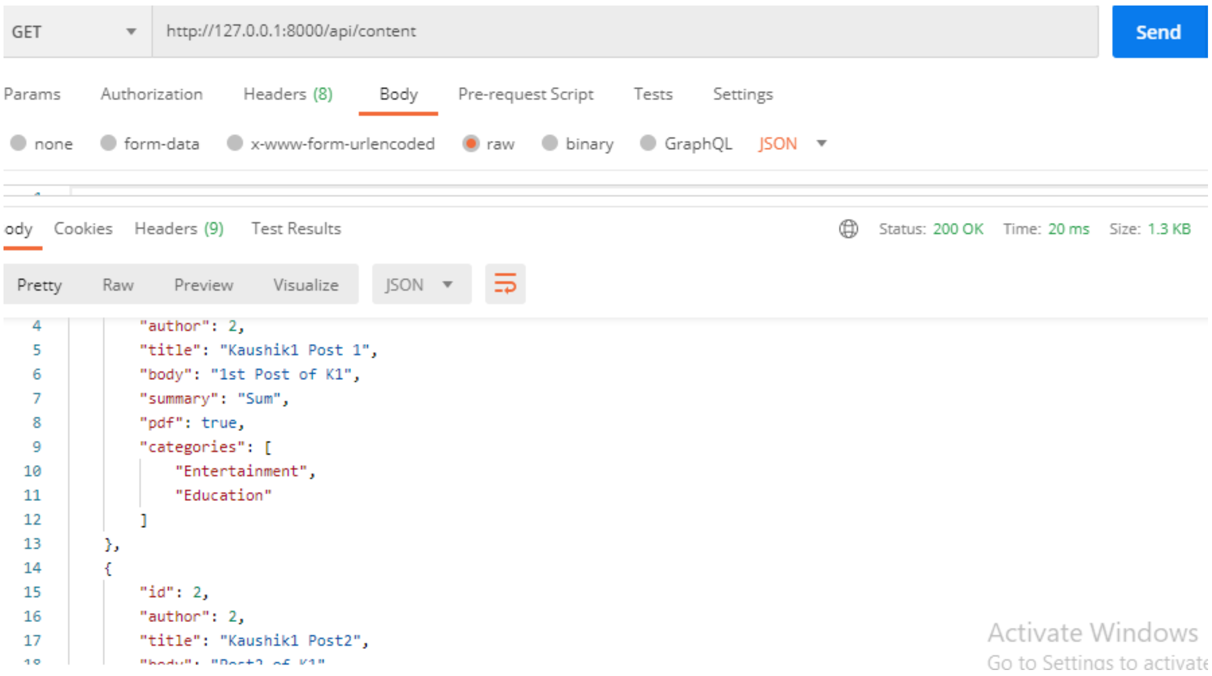
Regular users can view only their own content



```
http://127.0.0.1:8000/api/content
```

```
        "title": "User1 Post1",
        "body": "Post1 Body of User1",
        "summary": "Summary of U1 P1",
        "pdf": true,
        "categories": [
            "Entertainment",
            "Sports"
        ]
    }
```

Admins can view the contents of all users



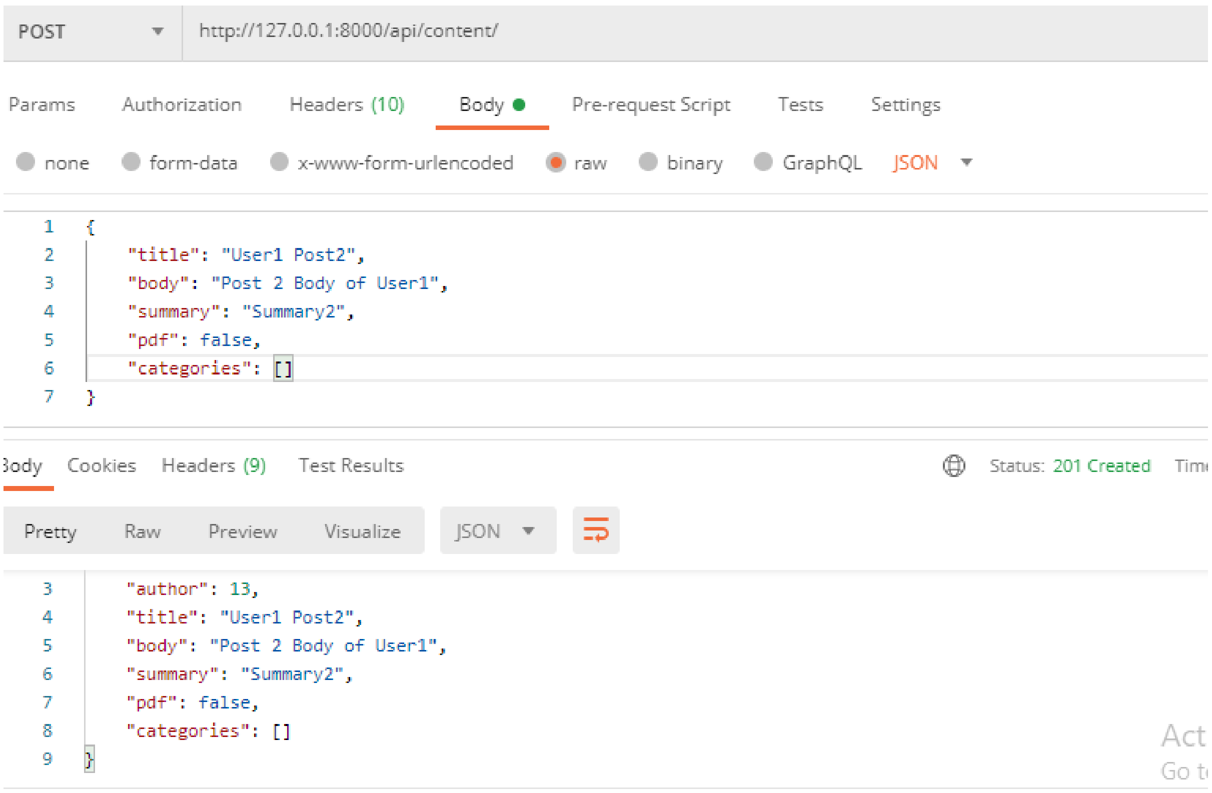
http://127.0.0.1:8000/api/content

Status: 200 OK Time: 20 ms Size: 1.3 KB

```
        "author": 2,
        "title": "Kaushik1 Post 1",
        "body": "1st Post of K1",
        "summary": "Sum",
        "pdf": true,
        "categories": [
            "Entertainment",
            "Education"
        ]
    },
    {
        "id": 2,
        "author": 2,
        "title": "Kaushik1 Post2",
```

Adding New Content

New posts can be added by making a POST request to http://domain_name**/api/content/**



http://127.0.0.1:8000/api/content/

```
{
    "title": "User1 Post2",
    "body": "Post 2 Body of User1",
    "summary": "Summary2",
    "pdf": false,
    "categories": []
}
```

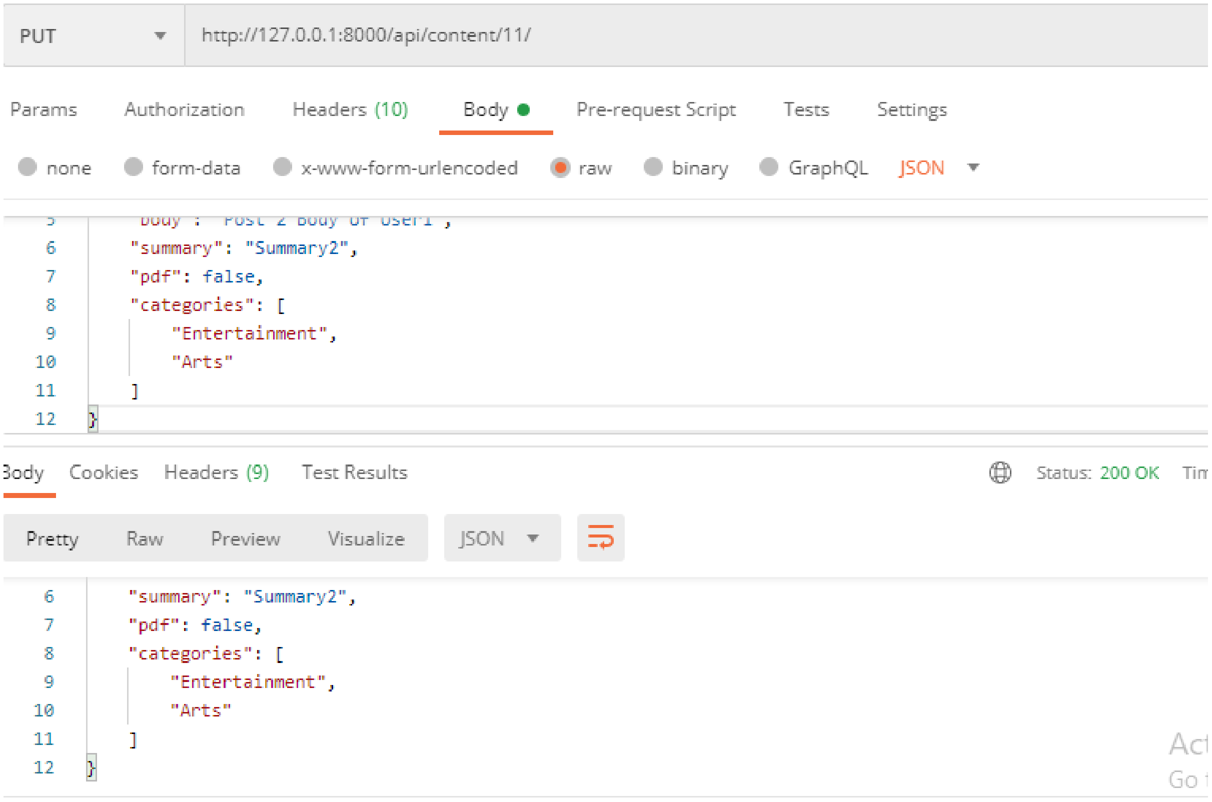
Status: 201 Created

```
    "author": 13,
    "title": "User1 Post2",
    "body": "Post 2 Body of User1",
    "summary": "Summary2",
    "pdf": false,
    "categories": []
}
```

Updating Content

Contents can be updated by making a PUT request to http://domain_name**/api/content/<content_id>/**

Authors can update only contents created by them. Admins can update contents created by all Authors.



PUT http://127.0.0.1:8000/api/content/11/

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
    "body": "Post 2 body of user1",
    "summary": "Summary2",
    "pdf": false,
    "categories": [
        "Entertainment",
        "Arts"
    ]
}
```

Body Cookies Headers (9) Test Results Status: 200 OK

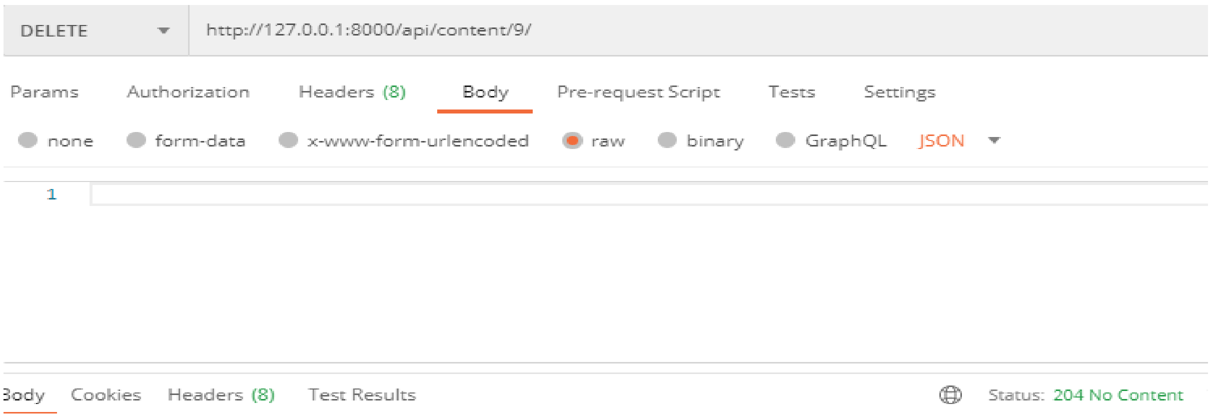
Pretty Raw Preview Visualize JSON

```
    "summary": "Summary2",
    "pdf": false,
    "categories": [
        "Entertainment",
        "Arts"
    ]
}
```

Deleting Content

Contents can be deleted by making a DELETE request to http://domain_name**/api/content/<content_id>/**

Authors can delete only contents created by them. Admins can delete contents created by all Authors.



DELETE http://127.0.0.1:8000/api/content/9/

Params Authorization Headers (8) Body Pre-request Script Tests Settings

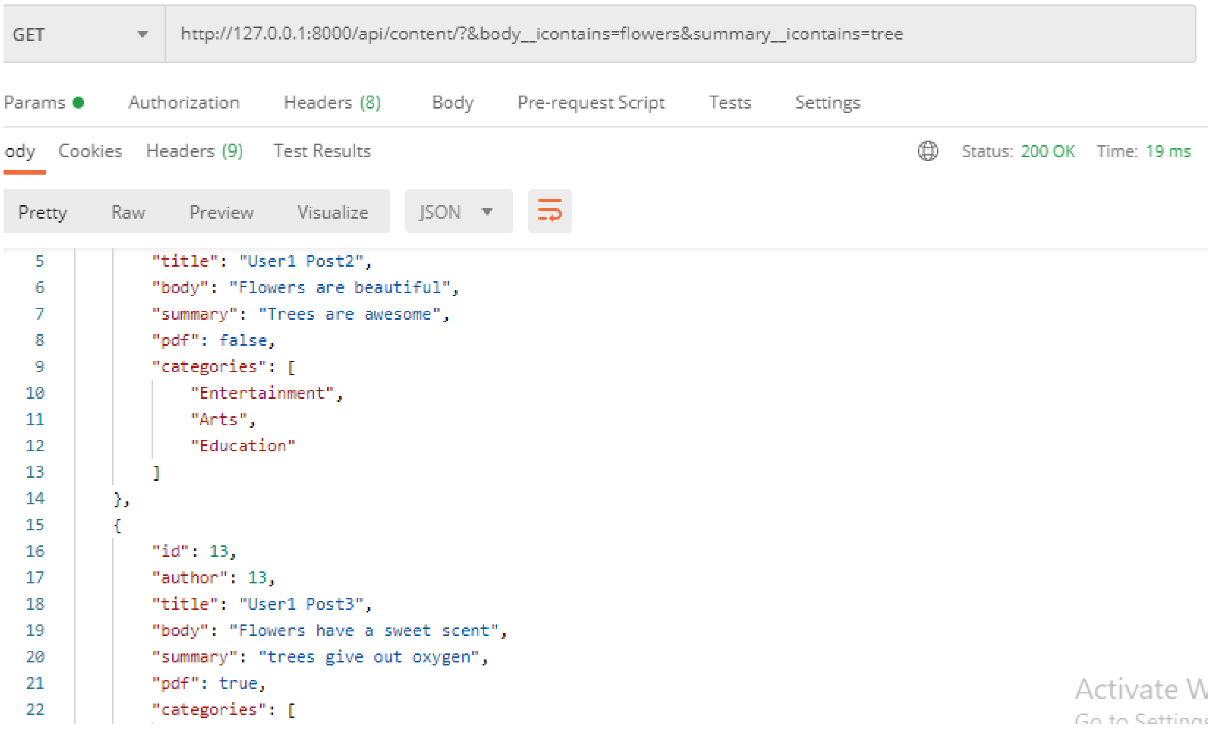
none form-data x-www-form-urlencoded raw binary GraphQL JSON

Body Cookies Headers (8) Test Results Status: 204 No Content

Searching and Filtering

Users can search content by matching terms in title, body, summary and categories by making a request to **/api/content/?** followed by your query in the form of **&<field_name>__icontains=<term>**

Example:
http://127.0.0.1:8000/api/content/?&**body**__icontains=**flowers**&**summary**__icontains=**tree**



```
        "title": "User1 Post2",
        "body": "Flowers are beautiful",
        "summary": "Trees are awesome",
        "pdf": false,
        "categories": [
            "Entertainment",
            "Arts",
            "Education"
        ]
    },
    {
        "id": 13,
        "author": 13,
        "title": "User1 Post3",
        "body": "Flowers have a sweet scent",
        "summary": "trees give out oxygen",
        "pdf": true,
        "categories": [
```