

ASSIGNMENT

Tools Used

- Visual Studio Code
- Postman

Programming Language

- Python

Frameworks

- Python Django
- Django REST framework

Super-User Login

Username : admin

Password : Admin@123

Steps

Step 1 : Create Virtual Environment

- `Python -m venv environment_name`

Step 2 : Activate Virtual Environment

- `Source environment_name/scripts/activate`

Step 3 : Install Django and Django REST Framework

- `pip install django`
- `pip install djangorestframework`

Step 4 : Create a new Django project

- `django-admin startproject ecommerce`

Step 5 : Create a new App as API

- `python manage.py startapp customers`

Step 6 : After Developing Make migrations

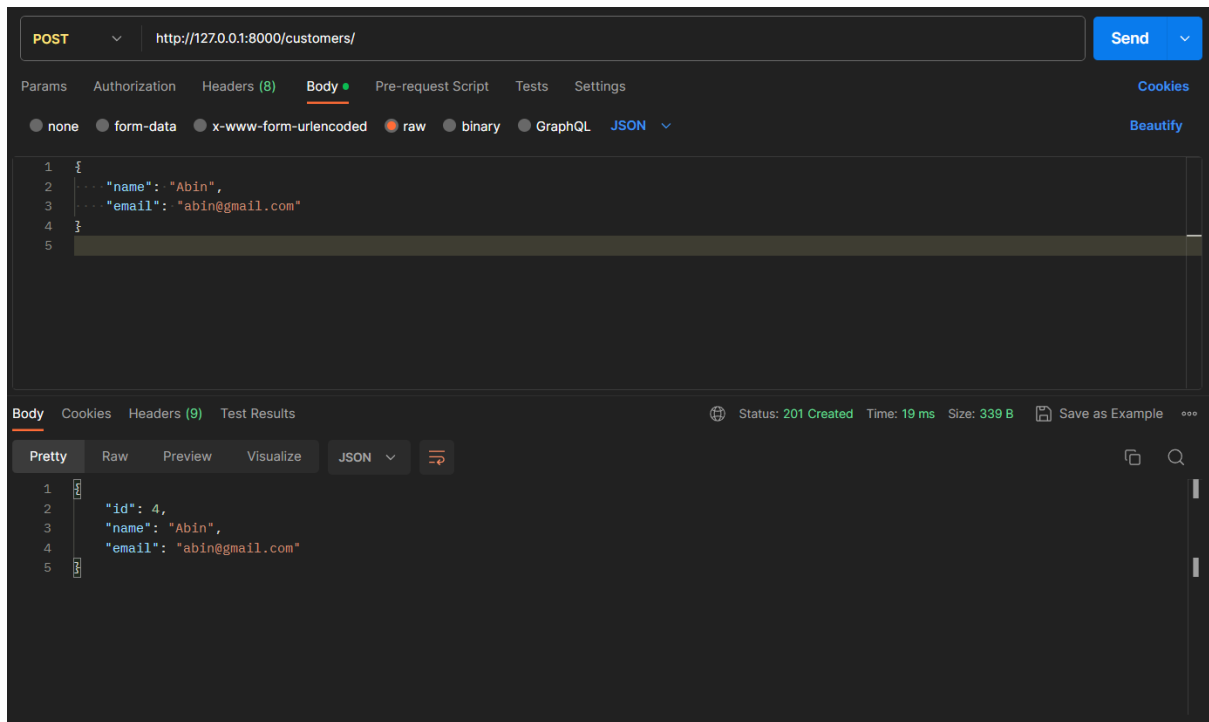
- `python manage.py makemigrations`
- `python manage.py migrate`

Step 7: Start the Server

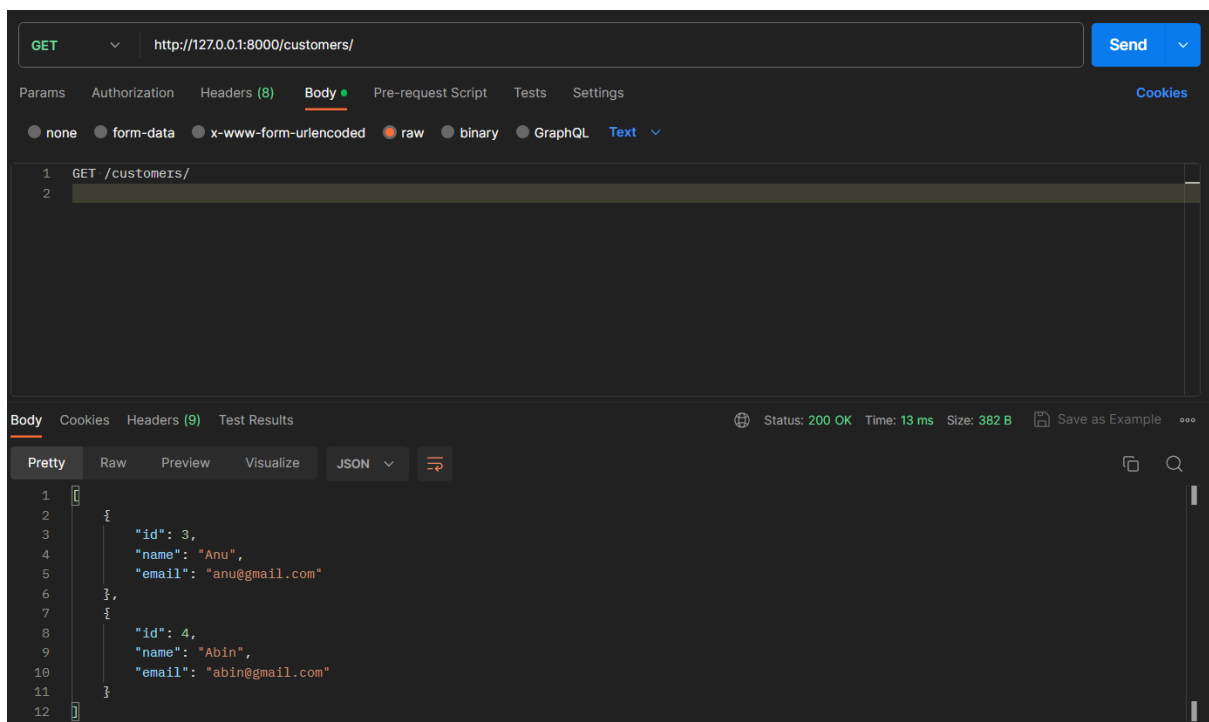
➤ `python manage.py runserver`

Step 8: Test the Developed API using Postman

1. Create a New User



2. Get Customers



3. Get Specific Customer

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://127.0.0.1:8000/customers/3/
- Body:** The request body is empty.
- Status:** 200 OK
- Time:** 12 ms
- Size:** 334 B
- Response Body (JSON):**

```
{  "id": 3,  "name": "Anu",  "email": "anu@gmail.com"}
```

4. Update Customer

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** http://127.0.0.1:8000/customers/3/
- Body:** The request body contains a JSON object:

```
{  "name": "Anu Jose",  "email": "anujose@gmail.com"}
```
- Status:** 200 OK
- Time:** 26 ms
- Size:** 355 B
- Response Body (JSON):**

```
{  "id": 3,  "name": "Anu Jose",  "email": "anujose@gmail.com"}
```

5. Delete Customer

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** http://127.0.0.1:8000/customers/
- Body:** 1 DELETE /customers/2/
- Status:** 204 No Content
- Time:** 11 ms
- Size:** 276 B
- Buttons:** Send, Beautify, Save as Example

6. Add a new product

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://127.0.0.1:8000/products/
- Body:** 1 {
2 ... "customer": 2,
3 ... "name": "Xiaomi A3"
4 }
5
6
- Status:** 201 Created
- Time:** 14 ms
- Size:** 389 B
- Buttons:** Send, Beautify, Save as Example

7. Get List of all products

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:8000/products/1/`. The request is sent, and the response is displayed in the Body tab, formatted as JSON. The response status is 200 OK, with a time of 10 ms and a size of 382 B.

```
1 GET /products/
2
```

```
{
  "id": 3,
  "name": "Redmi",
  "active": true,
  "created_at": "2023-06-19T16:19:11.886083Z",
  "customer": 3
}
```

Status: 200 OK Time: 10 ms Size: 382 B Save as Example

8. Get Specefic Product

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:8000/products/4/`. The request is sent, and the response is displayed in the Body tab, formatted as JSON. The response status is 200 OK, with a time of 16 ms and a size of 389 B.

```
1 GET /products/4/
2
```

```
{
  "id": 4,
  "name": "Samsung S22",
  "active": true,
  "created_at": "2023-06-19T17:31:24.618599Z",
  "customer": 3
}
```

Status: 200 OK Time: 16 ms Size: 389 B Save as Example

9. Update a product

The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:8000/products/3/`. The request body is a JSON object: `{ "customer": 3, "name": "Redmi A3" }`. The response status is 200 OK, with a time of 14 ms and a size of 397 B. The response body is a JSON object: `{ "id": 3, "name": "Redmi A3", "active": true, "created_at": "2023-06-19T16:19:11.886083Z", "customer": 3 }`.

```
PUT http://127.0.0.1:8000/products/3/

{
  "customer": 3,
  "name": "Redmi A3"
}
```

Status: 200 OK Time: 14 ms Size: 397 B Save as Example

```
{
  "id": 3,
  "name": "Redmi A3",
  "active": true,
  "created_at": "2023-06-19T16:19:11.886083Z",
  "customer": 3
}
```

10. Delete a product

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:8000/products/`. The request body is a JSON object: `{ "customer": 2, "name": "Xiaomi A3" }`. The response status is 201 Created, with a time of 14 ms and a size of 389 B. The response body is a JSON object: `{ "id": 2, "name": "Xiaomi A3", "active": true, "created_at": "2023-06-19T16:06:34.213691Z", "customer": 2 }`.

```
POST http://127.0.0.1:8000/products/

{
  "customer": 2,
  "name": "Xiaomi A3"
}
```

Status: 201 Created Time: 14 ms Size: 389 B Save as Example

```
{
  "id": 2,
  "name": "Xiaomi A3",
  "active": true,
  "created_at": "2023-06-19T16:06:34.213691Z",
  "customer": 2
}
```

11. Status of product

The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:8000/products/4/`. The request body is a JSON object: `{ "name": "Samsung S22", "customer": 3, "is_active": false }`. The response status is 200 OK, with a time of 37 ms and a size of 400 B. The response body is a JSON object: `{ "id": 4, "name": "Samsung S22", "active": true, "created_at": "2023-06-19T17:31:24.618599Z", "customer": 3 }`.

PUT `http://127.0.0.1:8000/products/4/` Send

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

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** Beautify

```
1 {
2   "name": "Samsung S22",
3   "customer": 3,
4   "is_active": false
5 }
6
```

Body Cookies Headers (9) Test Results Status: 200 OK Time: 37 ms Size: 400 B Save as Example

Pretty Raw Preview Visualize **JSON**

```
1 {
2   "id": 4,
3   "name": "Samsung S22",
4   "active": true,
5   "created_at": "2023-06-19T17:31:24.618599Z",
6   "customer": 3
7 }
```