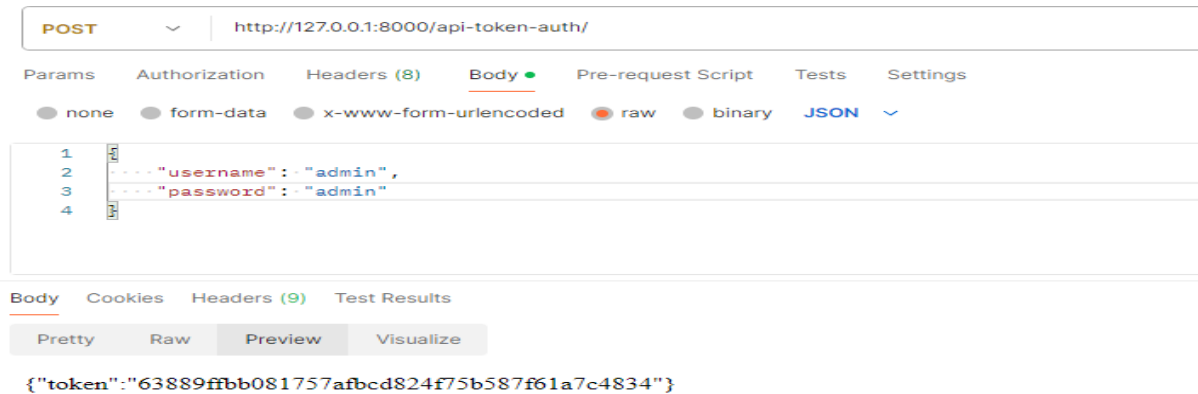


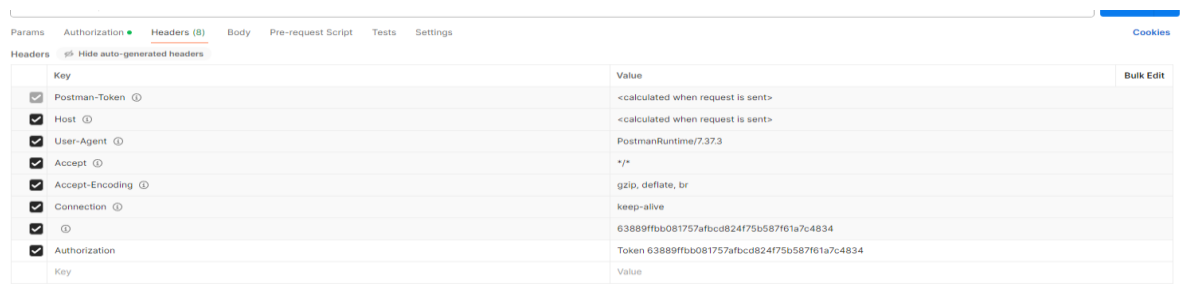
Testing Endpoints Step by Step

Authorization: We are using token based authentication. So first of all we have to get a token to make requests to all of our endpoints. We have to make sure our Django server is running. Then open postman.

We have to put a POST request at - <http://127.0.0.1:8000/api-token-auth/> , in body we are sending username and password. In response it will give us back a token that we can use.

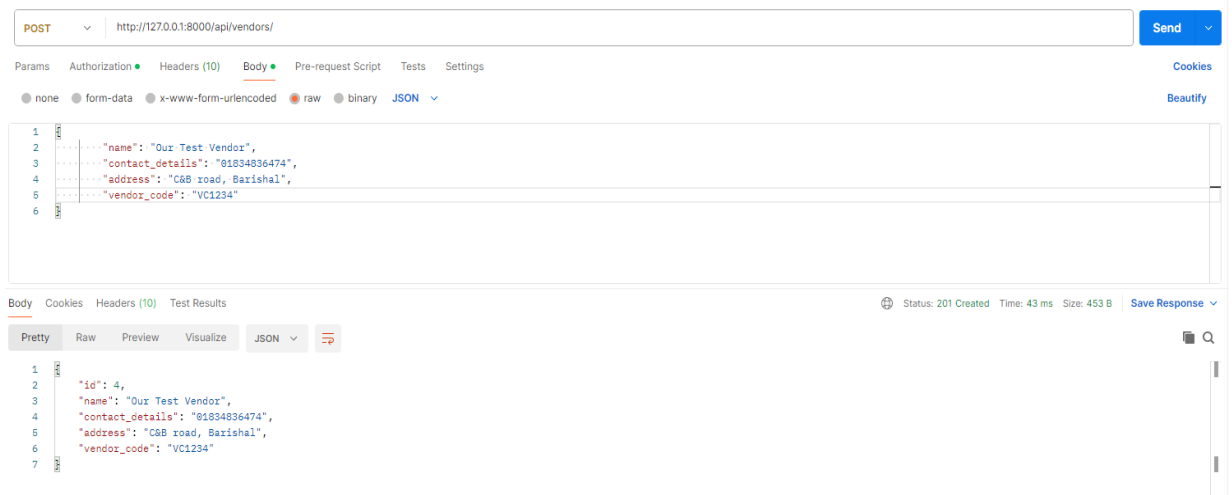


Now we have to go to header section on Postman, and put authorization and token like the below example. We are all set up to make requests to our endpoints now!



Endpoints

1. Create a new vendor



If we make a POST request at <http://127.0.0.1:8000/api/vendors/> it with the data in body like above, it will create a vendor for us.

2. List all vendors:

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:8000/api/vendors/`. The response is displayed in JSON format, showing a list of three vendors. The first two vendors have IDs 2 and 3, and the third has ID 4. Each vendor object contains fields for `id`, `name`, `contact_details`, `address`, and `vendor_code`.

```
1 [
2   {
3     "id": 2,
4     "name": "Vendor Name 2",
5     "contact_details": "Contact Details of Vendor",
6     "address": "Address of Vendor",
7     "vendor_code": "Uniqu 2e"
8   },
9   {
10    "id": 3,
11    "name": "Vendor Name 3",
12    "contact_details": "Contact Details of Vendor",
13    "address": "Address of Vendor",
14    "vendor_code": "Uniqu 2e3"
15  },
16  {
17    "id": 4,
18    "name": "Our Test Vendor",
19    "contact_details": "01834836474",
20    "address": "C&B road, Barishal",
21    "vendor_code": "VC1234"
22  }
23 ]
```

We send a GET request at <http://127.0.0.1:8000/api/vendors/> and it gives us all the vendors.

3. Retrieve a specific vendor's details:

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:8000/api/vendors/4`. The response is displayed in JSON format, showing the details of a single vendor with ID 4. The response object contains fields for `id`, `name`, `contact_details`, `address`, and `vendor_code`.

```
1 {
2   "id": 4,
3   "name": "Our Test Vendor",
4   "contact_details": "01834836474",
5   "address": "C&B road, Barishal",
6   "vendor_code": "VC1234"
7 }
```

We make a GET request at <http://127.0.0.1:8000/api/vendors/4> and get details of the vendor with id 4.

4. Update a vendor's details:

The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:8000/api/vendors/4`. The 'Body' tab is selected, and the request body is in JSON format. The JSON object contains the following fields: `id` (4), `name` (Our Test Vendor), `contact_details` (01834836474), `address` (C&B road, Barishal updated), and `vendor_code` (VC1234). The 'Test Results' tab is also visible, showing the response body in JSON format: `{ "id": 4, "name": "Our Test Vendor", "contact_details": "01834836474", "address": "C&B road, Barishal updated", "vendor_code": "VC1234" }`.

```
1 {
2   "id": 4,
3   "name": "Our Test Vendor",
4   "contact_details": "01834836474",
5   "address": "C&B road, Barishal updated",
6   "vendor_code": "VC1234"
7 }
```

```
{ "id": 4, "name": "Our Test Vendor", "contact_details": "01834836474", "address": "C&B road, Barishal updated", "vendor_code": "VC1234" }
```

We made a PUT request at <http://127.0.0.1:8000/api/vendors/4> and we are just updating the address of the vendor. It updates successfully.

5. Delete a vendor:

The screenshot shows a REST client interface with a DELETE request to `http://127.0.0.1:8000/api/vendors/4`. The 'Body' tab is selected, and the request body is empty. The 'Test Results' tab is also visible, showing the response body in JSON format: `{ "id": 4, "name": "Our Test Vendor", "contact_details": "01834836474", "address": "C&B road, Barishal updated", "vendor_code": "VC1234" }`.

```
{ "id": 4, "name": "Our Test Vendor", "contact_details": "01834836474", "address": "C&B road, Barishal updated", "vendor_code": "VC1234" }
```

If we put a DELETE request, it will delete the vendor. For now, we are not deleting it for testing next endpoints.

6. Create a purchase order:

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:8000/api/purchase_orders/`. The 'Body' tab is selected, and the request body is in JSON format. The JSON object contains the following fields: `po_number` (PO1255), `order_date` (2024-04-27T10:00:00Z), `delivery_date` (2024-05-10T12:00:00Z), `items` (an array of two items: Item 1 with quantity 10 and price 20.5, and Item 2 with quantity 5 and price 15.75), `quantity` (2), `status` (pending), `quality_rating` (0.0), `issue_date` (2024-04-27T10:00:00Z), and `vendor` (4).

```
1 {
2   "po_number": "PO1255",
3   "order_date": "2024-04-27T10:00:00Z",
4   "delivery_date": "2024-05-10T12:00:00Z",
5   "items": [
6     {
7       "name": "Item 1",
8       "quantity": 10,
9       "price": 20.5
10    },
11    {
12      "name": "Item 2",
13      "quantity": 5,
14      "price": 15.75
15    }
16  ],
17   "quantity": 2,
18   "status": "pending",
19   "quality_rating": 0.0,
20   "issue_date": "2024-04-27T10:00:00Z",
21   "vendor": 4
22 }
```

We send a POST request at http://127.0.0.1:8000/api/purchase_orders/ and make a PO for vendor 4
After the PO object is created, Django signal creates an object of VendorPerformance for vendor 4 and initializes it.(Prove from Django-admin)

VendorPerformance object (6)

Vendor:

Vendor object (4) ▾

Date:

Date:

2024-04-28

 Today | 📅

Time:

18:49:05

 Now | 🕒

Note: You are 6 hours ahead of server time.

On time delivery rate:

0.0

Quality rating avg:

0.0

Average response time:

0.0

Fulfillment rate:

0.0

SAVE

Save and add another

Save and continue editing

7. List all purchase orders with an option to filter by vendor

GET ▾

http://127.0.0.1:8000/api/purchase_orders/

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

none

form-data

x-www-form-urlencoded

raw

binary

This request does not have a body

Body Cookies Headers (10) Test Results

⚙️ Status: 200 OK Time: 38 ms Size: 1022 B

Pretty

Raw

Preview

Visualize

[{"id":1,"po_number":"PO12345","order_date":"2024-04-27T10:00:00Z","delivery_date":"2024-05-10T12:00:00Z","items":[{"name":"Item 1","quantity":10,"price":20.5}, {"name":"Item 2","quantity":5,"price":15.75}], "quantity":2,"status":"completed","quality_rating":4.2,"issue_date":"2024-04-27T10:00:00Z","acknowledgment_date":"2024-04-28T18:12:52.542849Z","vendor":2}, {"id":2,"po_number":"PO1255","order_date":"2024-04-27T10:00:00Z","delivery_date":"2024-05-10T12:00:00Z","items":[{"name":"Item 1","quantity":10,"price":20.5}, {"name":"Item 2","quantity":5,"price":15.75}], "quantity":2,"status":"pending","quality_rating":0.0,"issue_date":"2024-04-27T10:00:00Z","acknowledgment_date":null,"vendor":4}]

We send a GET request at http://127.0.0.1:8000/api/purchase_orders/ and it returned all the POs

GET ▾

http://127.0.0.1:8000/api/purchase_orders?vendor=4

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

none

form-data

x-www-form-urlencoded

raw

binary

This request does not have a body

Body Cookies Headers (10) Test Results

⚙️ Status: 200 OK Time:

Pretty

Raw

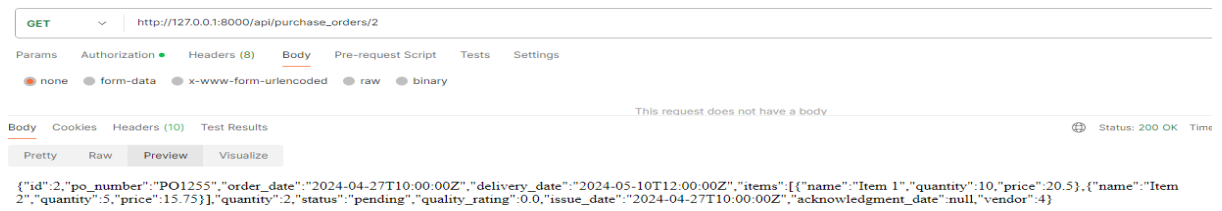
Preview

Visualize

[{"id":2,"po_number":"PO1255","order_date":"2024-04-27T10:00:00Z","delivery_date":"2024-05-10T12:00:00Z","items":[{"name":"Item 1","quantity":10,"price":20.5}, {"name":"Item 2","quantity":5,"price":15.75}], "quantity":2,"status":"pending","quality_rating":0.0,"issue_date":"2024-04-27T10:00:00Z","acknowledgment_date":null,"vendor":4}]

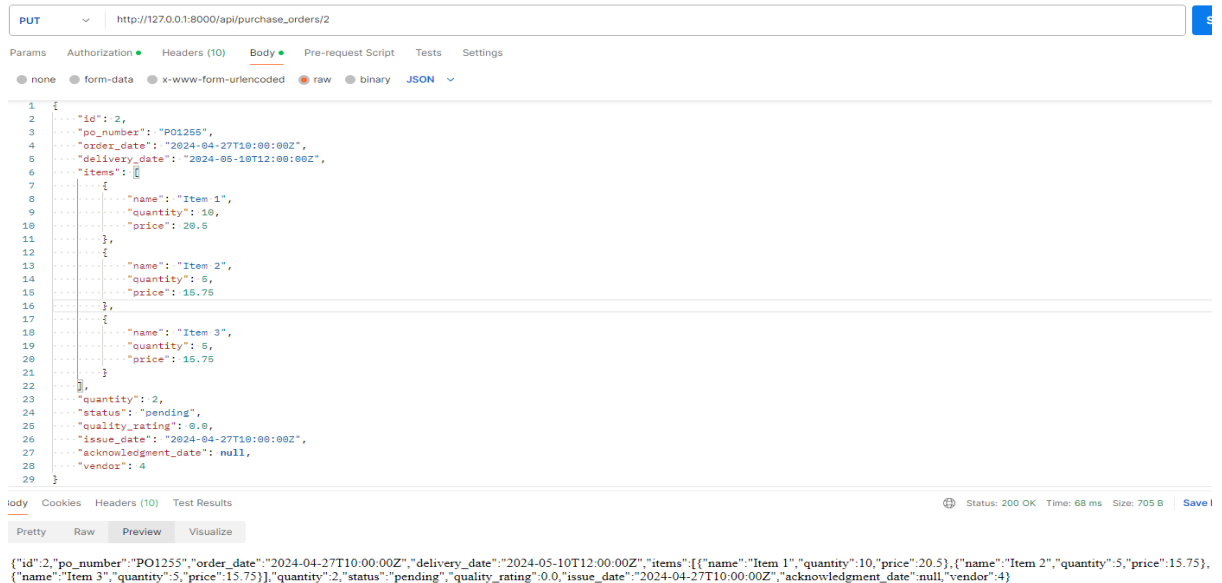
We send GET request at http://127.0.0.1:8000/api/purchase_orders?vendor=4 and it returned all the POs for the vendor 4. So the option to filter by vendor also works.

8. Retrieve details of a specific purchase order:



We send a GET request at http://127.0.0.1:8000/api/purchase_orders/2 to get the PO with the id 2.

9. Update a purchase order:



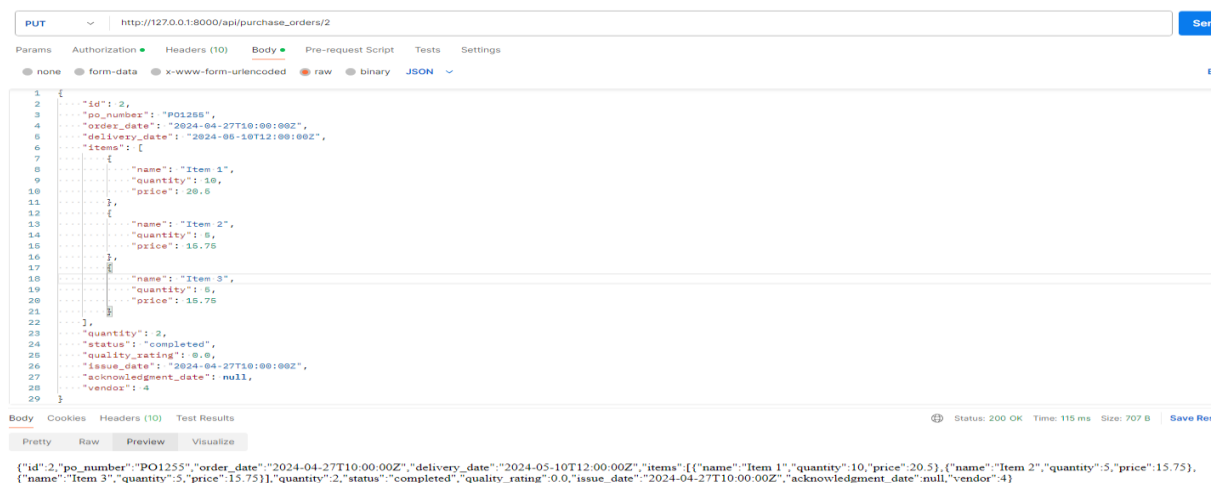
We send a PUT request at http://127.0.0.1:8000/api/purchase_orders/2 to add a new item to the PO with id 2. It is successfully updated.

10. Delete a purchase order:

For the sake of next endpoints testing we are not deleting.

Performance

First let us set the status of our recently created PO with id 2 and make it completed. It has some consequences. Any change to this PO will trigger Django signals and it will calculate the metrics.



Now let us put a POST request for acknowledgement.

POST ⌵ http://127.0.0.1:8000/api/purchase_orders/2/acknowledge

Params Authorization ● Headers (9) Body Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary

This request

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize

```
{\"message\": \"Purchase order acknowledged successfully.\"}
```

Now let's check the performance of vendor 4. And his recently created PO id is 2.

GET ⌵ http://127.0.0.1:8000/api/vendors/4/performance

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

● none ● form-data ● x-www-form-urlencoded ● raw ● binary

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON ⌵ ☰

```
1 {  
2   \"id\": 6,  
3   \"date\": \"2024-04-28T19:25:43.222652Z\",  
4   \"on_time_delivery_rate\": 1.0,  
5   \"quality_rating_avg\": 0.0,  
6   \"average_response_time\": 2005.7195943333331,  
7   \"fulfillment_rate\": 1.0,  
8   \"vendor\": 4  
9 }
```

So we can see the performance of the vendor 4. But his average rating is 0 because he has not received any rating yet. So let us give him a rating.

PUT http://127.0.0.1:8000/api/purchase_orders/2

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

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

```
2 ..... "id": 2,
3 ..... "po_number": "PO1255",
4 ..... "order_date": "2024-04-27T10:00:00Z",
5 ..... "delivery_date": "2024-05-10T12:00:00Z",
6 ..... "items": [
7 ..... {
8 .....   "name": "Item 1",
9 .....   "quantity": 10,
10 .....   "price": 20.5
11 ..... },
12 ..... {
13 .....   "name": "Item 2",
14 .....   "quantity": 5,
15 .....   "price": 15.75
16 ..... },
17 ..... {
18 .....   "name": "Item 3",
19 .....   "quantity": 5,
20 .....   "price": 15.75
21 ..... }
22 ..... ],
23 ..... "quantity": 2,
24 ..... "status": "completed",
25 ..... "quality_rating": 5.0,
26 ..... "issue_date": "2024-04-27T10:00:00Z",
27 ..... "acknowledgment_date": "2024-04-28T19:25:43.175660Z",
28 ..... "vendor": 4
29 .....
```

body Cookies Headers (10) Test Results Status: 200 OK Time: 166 ms Size: 732 B Save R

Pretty Raw Preview Visualize

```
{
  "id": 2,
  "po_number": "PO1255",
  "order_date": "2024-04-27T10:00:00Z",
  "delivery_date": "2024-05-10T12:00:00Z",
  "items": [
    {
      "name": "Item 1",
      "quantity": 10,
      "price": 20.5
    },
    {
      "name": "Item 2",
      "quantity": 5,
      "price": 15.75
    },
    {
      "name": "Item 3",
      "quantity": 5,
      "price": 15.75
    }
  ],
  "quantity": 2,
  "status": "completed",
  "quality_rating": 5.0,
  "issue_date": "2024-04-27T10:00:00Z",
  "acknowledgment_date": "2024-04-28T19:25:43.175660Z",
  "vendor": 4
}
```

Let us check his performance again:

GET http://127.0.0.1:8000/api/vendors/4/performance

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

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

2 "id": 2,

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 ..... {
2 .....   "id": 6,
3 .....   "date": "2024-04-28T19:28:30.004488Z",
4 .....   "on_time_delivery_rate": 1.0,
5 .....   "quality_rating_avg": 5.0,
6 .....   "average_response_time": 2005.7195943333331,
7 .....   "fulfillment_rate": 1.0,
8 .....   "vendor": 4
9 ..... }
```

So now we see that the quality rating is updated.

Note: On_time_delivery_rate and fulfilment_rate is 1.0 because we only have one PO for vendor 4. We are calculating average_response_time in seconds.

From now on this performance calculation will be updated every time a new PO is created or updated for vendor 4.