

MHB Dips

Service Layer

Ed Davis

March 22, 2024

Overview:

The backend service for my web application will be built using Django. It will use a RESTful API which the front will use to create, request, update, and delete data.

The backend service will consist of the following three layers:

1. URLs: This layer will define the specific paths that are available and can be used to reach specific endpoints.
2. Controller: This layer will consist of the logic that will handle the HTTP methods that are being requested from the different paths.
3. Models – The service layer will be responsible for establishing the type of data that will be sent to the database as well as imposing requirements to protect the data's integrity.

Specifications:

Below is a list of routes that the application will consist of. Please note that I am using the localhost development server and a localhost Postgres server to test the endpoints. The service will be hosted on AWS when deployed.

Products:

Display products:

Method: GET

URL: 127.0.0.1:8000/api/products/

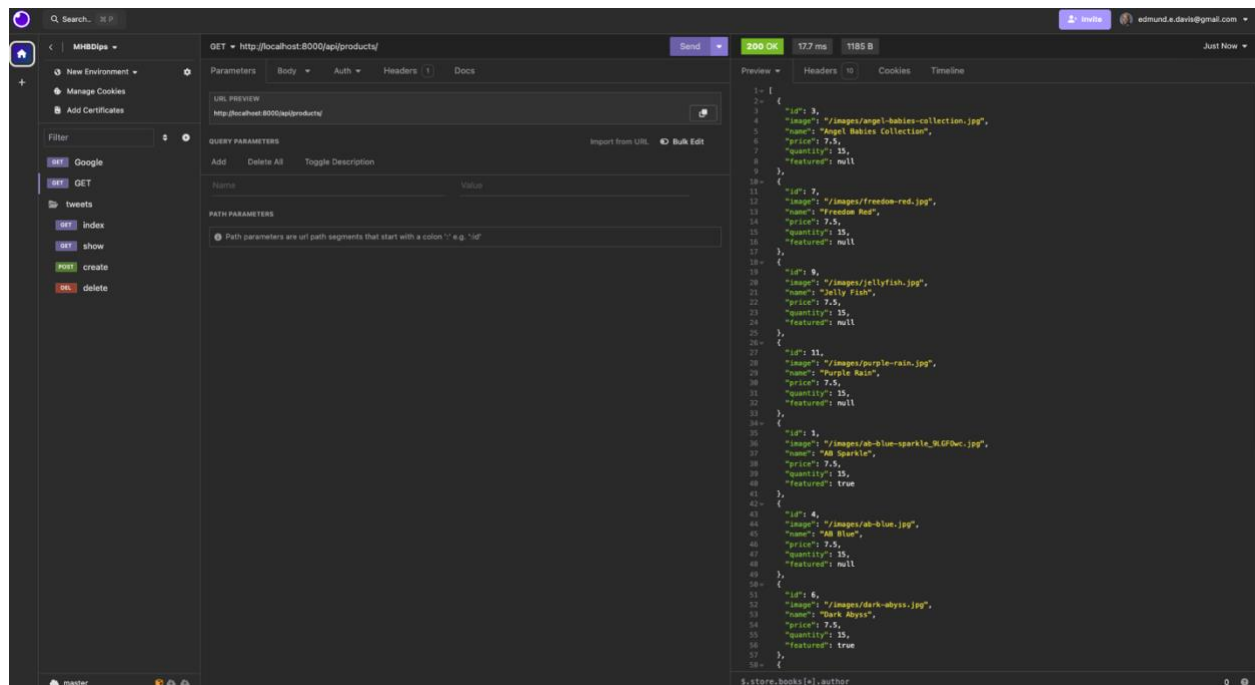
Purpose:

The URL will be used to retrieve products to be displayed on the home page, the product pages, and the cart page.

Example requests:

curl --request GET --url <http://localhost:8000/api/products/>

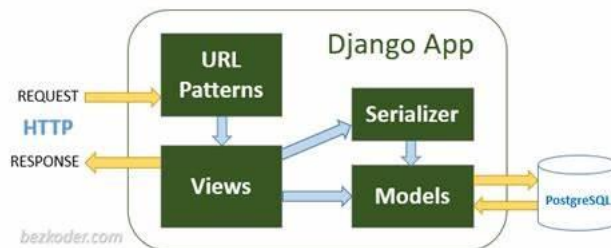
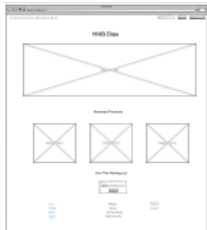
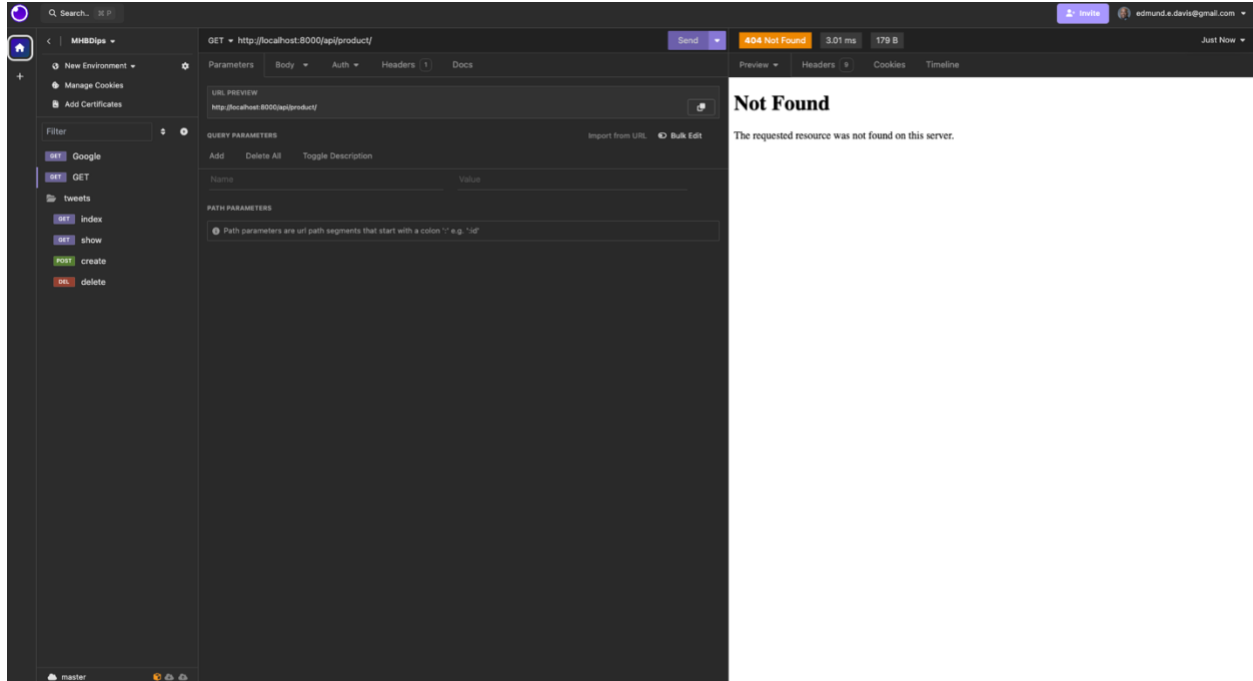
Success Response:



The screenshot shows a web browser interface with a dark theme. The address bar displays the URL `http://localhost:8000/api/products/`. The browser's developer tools are open, showing the 'Network' tab. A single request is listed, which is a GET request to the specified URL. The response status is '200 OK' with a response time of '17.7 ms' and a size of '1185 B'. The 'Preview' pane on the right displays the JSON response, which is an array of 6 product objects. Each object contains fields for 'id', 'image', 'name', 'price', 'quantity', and 'featured'.

```
1- [
2-   {
3-     "id": 1,
4-     "image": "/images/angel-babies-collection.jpg",
5-     "name": "Angel Babies Collection",
6-     "price": 7.5,
7-     "quantity": 15,
8-     "featured": null
9-   },
10-   {
11-     "id": 2,
12-     "image": "/images/freedom-red.jpg",
13-     "name": "Freedom Red",
14-     "price": 7.5,
15-     "quantity": 15,
16-     "featured": null
17-   },
18-   {
19-     "id": 3,
20-     "image": "/images/jellyfish.jpg",
21-     "name": "Jelly Fish",
22-     "price": 7.5,
23-     "quantity": 15,
24-     "featured": null
25-   },
26-   {
27-     "id": 4,
28-     "image": "/images/purple-rain.jpg",
29-     "name": "Purple Rain",
30-     "price": 7.5,
31-     "quantity": 15,
32-     "featured": null
33-   },
34-   {
35-     "id": 5,
36-     "image": "/images/ab-blue-sparkle-rgb.jpg",
37-     "name": "AB Sparkle",
38-     "price": 7.5,
39-     "quantity": 15,
40-     "featured": true
41-   },
42-   {
43-     "id": 6,
44-     "image": "/images/ab-blue.jpg",
45-     "name": "AB Blue",
46-     "price": 7.5,
47-     "quantity": 15,
48-     "featured": null
49-   },
50-   {
51-     "id": 7,
52-     "image": "/images/dark-abyss.jpg",
53-     "name": "Dark Abyss",
54-     "price": 7.5,
55-     "quantity": 15,
56-     "featured": true
57-   }
58- ]
```

Error Response



Display product by ID:

Method: GET

URL: `http://localhost:8000/api/products/:id/`

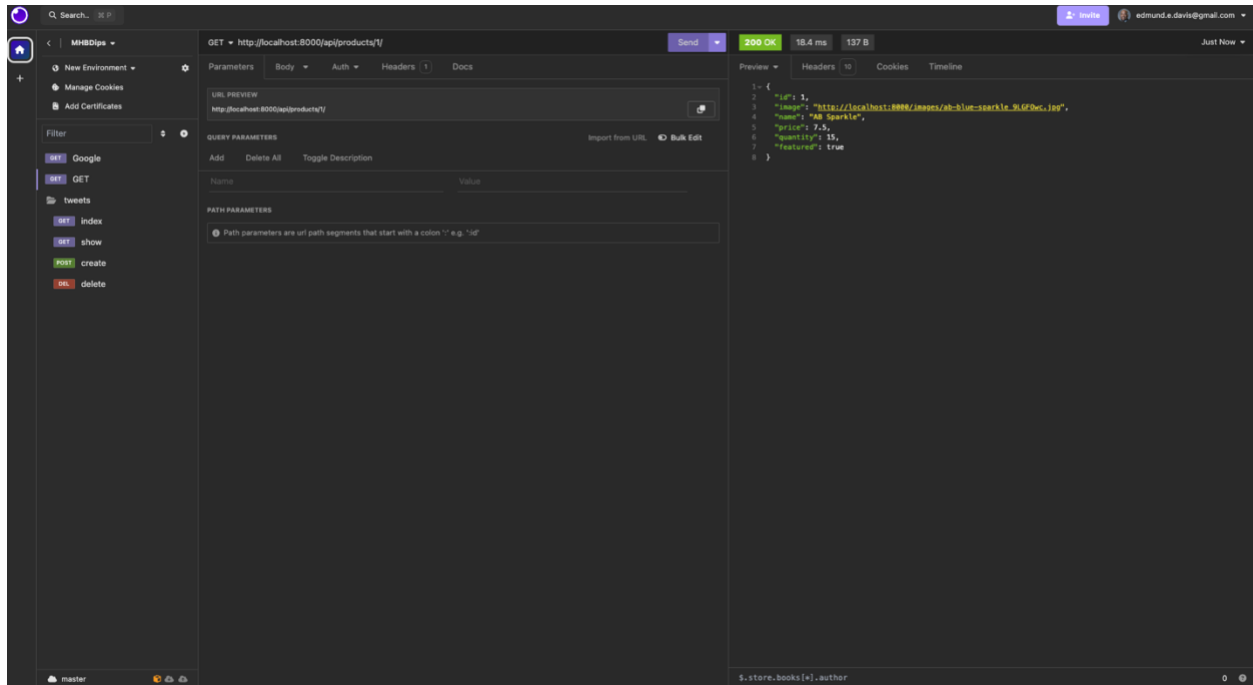
Purpose:

When a user clicks on a product image the endpoint will be called by the frontend. The user will be redirected to the product page that matches that ID.

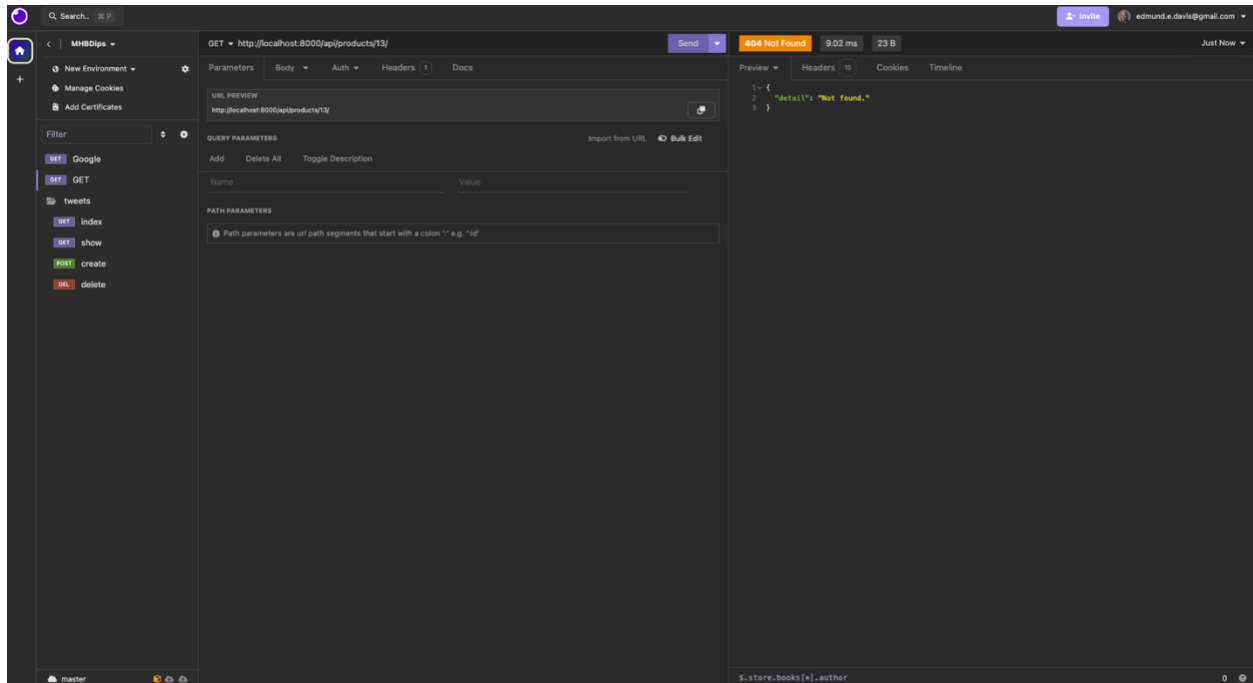
Example requests:

curl --request GET --url <http://localhost:8000/api/products/1/>

Success Response:



Error Response:



Accounts

Display products:

Method: POST

URL: 127.0.0.1:8000/api/accounts/

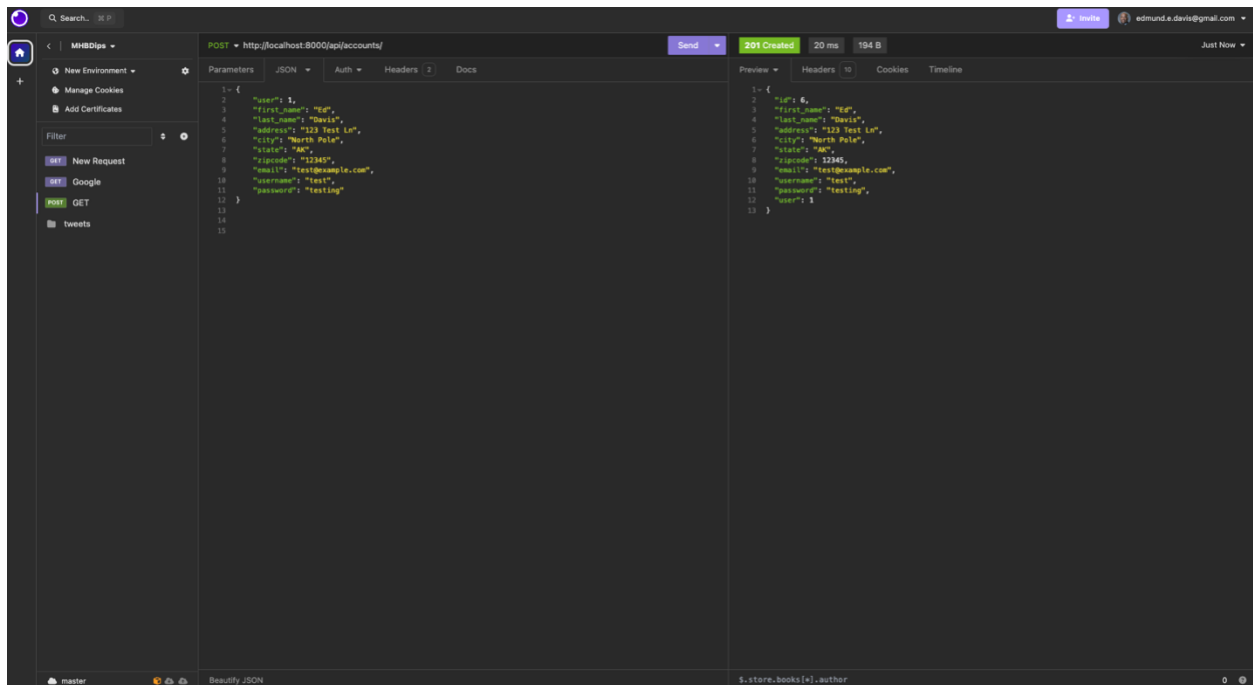
Purpose:

The URL will be used to add new customer accounts so they can login and make purchases/.

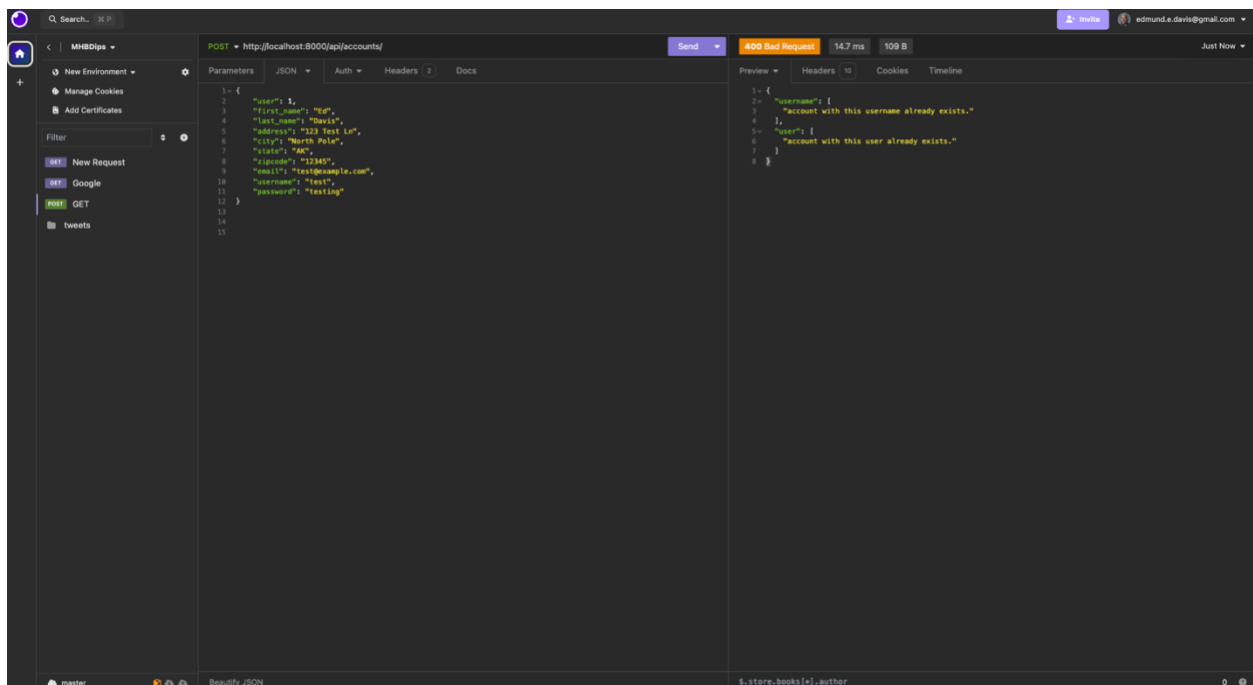
Example requests:

curl --request POST --url <http://localhost:8000/api/products/>

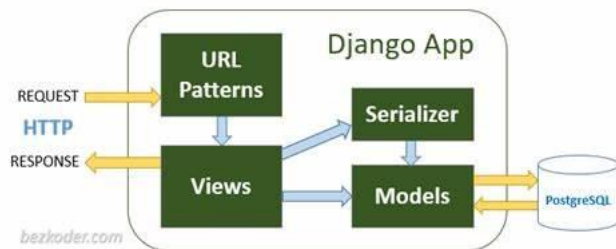
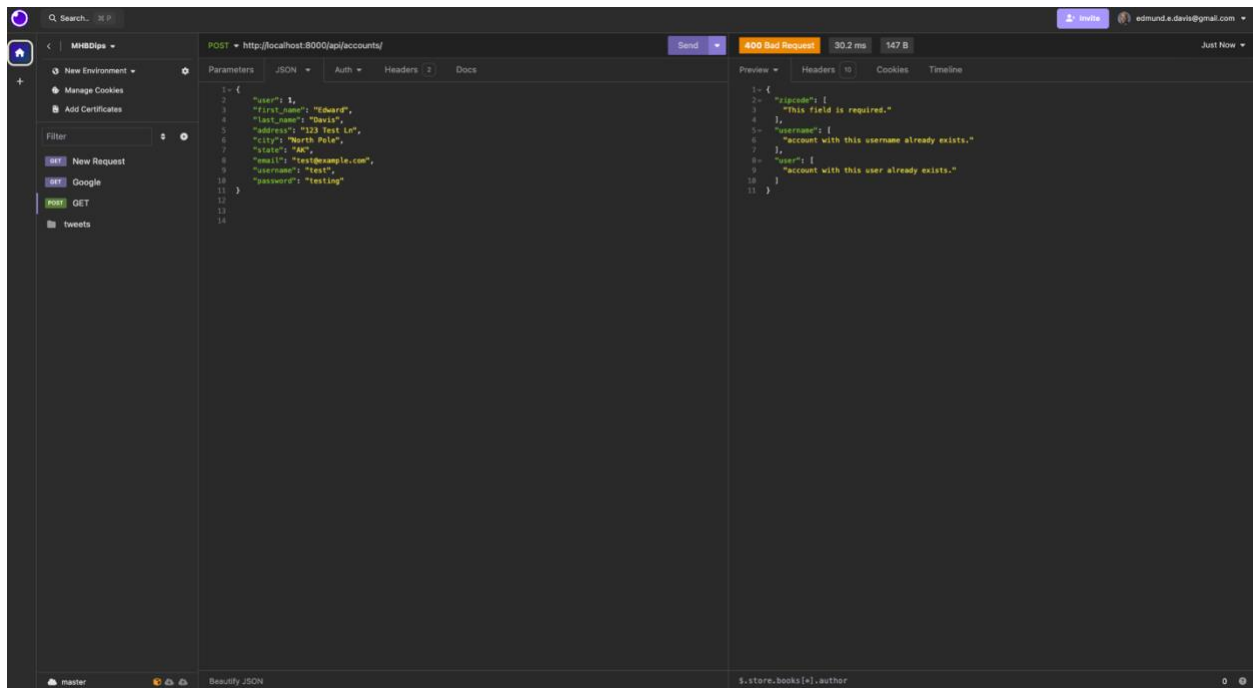
Success Response:



Error Response if User Exist:



Error Response:



Display Account by ID:

Method: GET

URL: <http://localhost:8000/api/accounts/:id/>

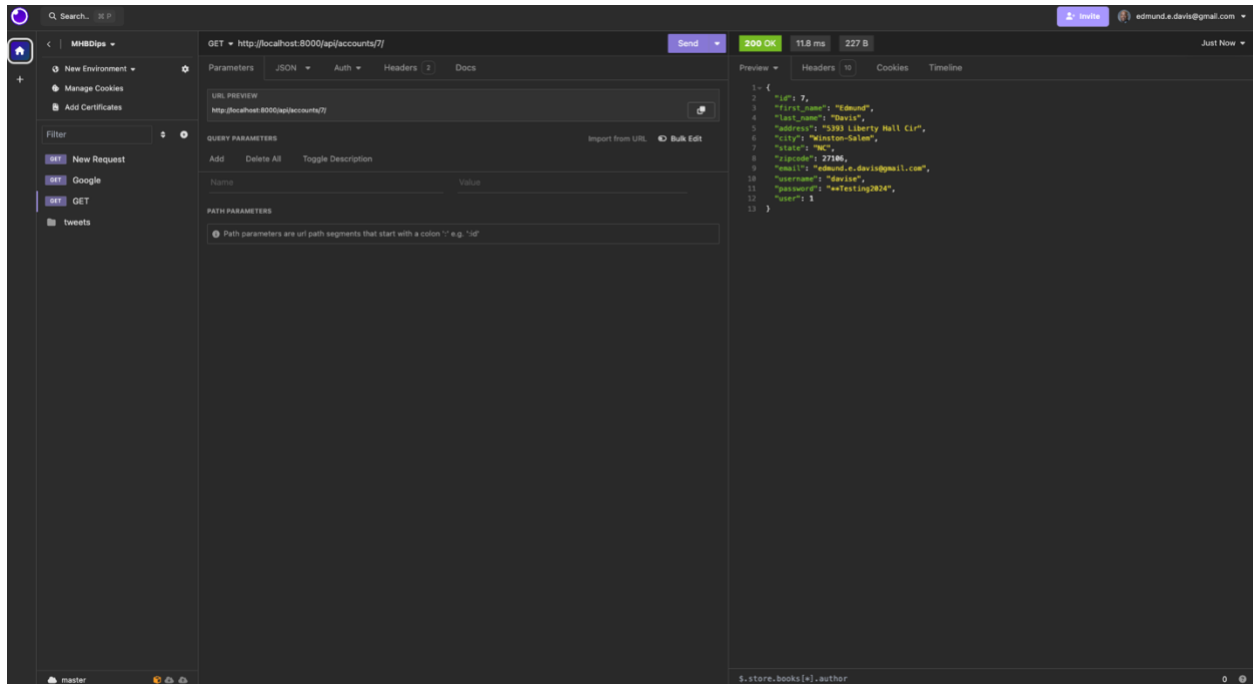
Purpose:

When a user logs in to the site. A GET request will be sent to the backend to fetch the user's information.

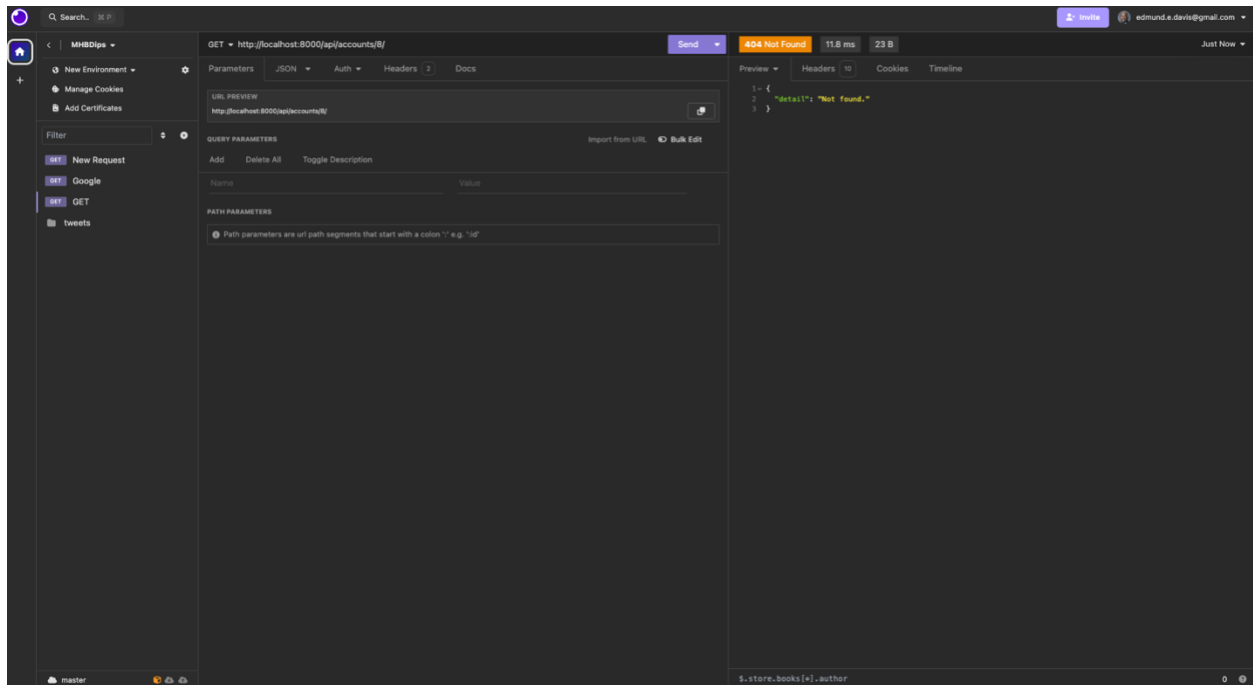
Example requests:

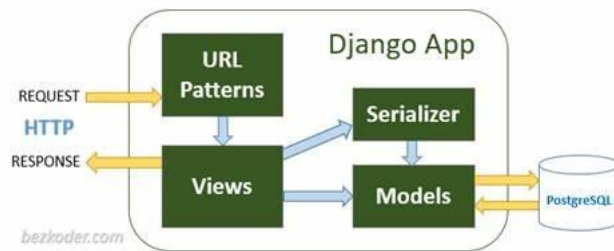
curl --request GET --url <http://localhost:8000/api/accounts/1/>

Success Response:



Error Response:





Reviews:

Display Reviews:

Method: POST

URL: 127.0.0.1:8000/api/reviews/

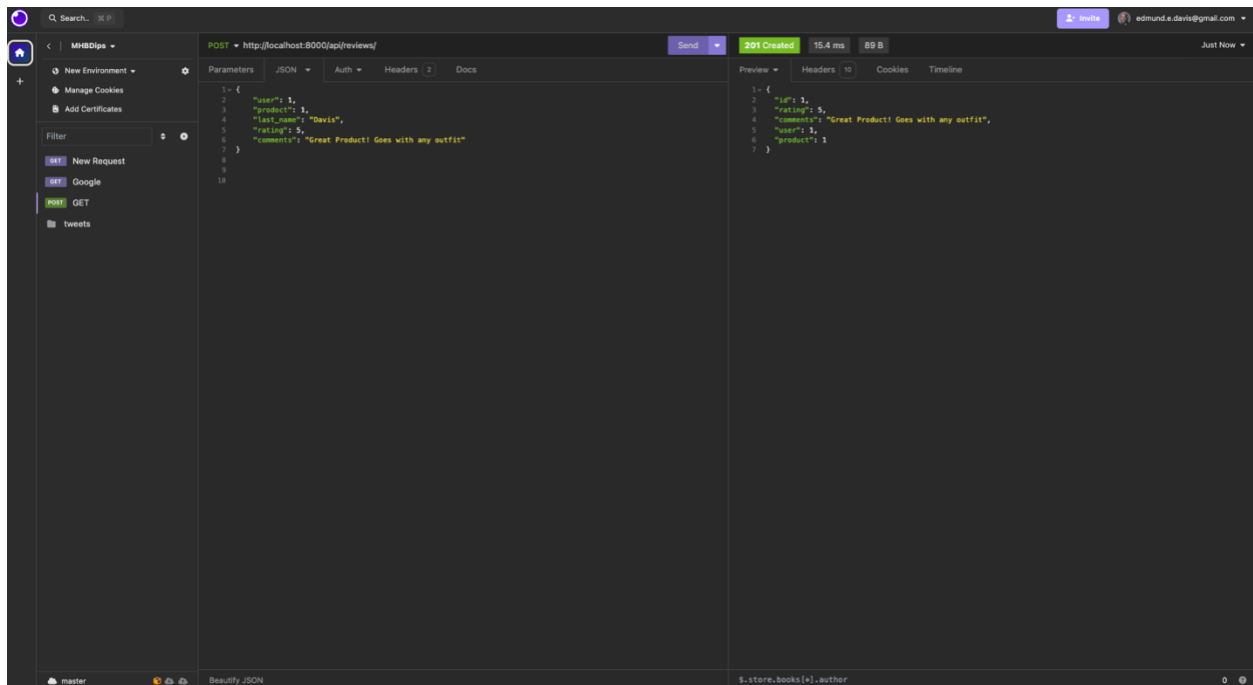
Purpose:

The URL will be used to post comments about products.

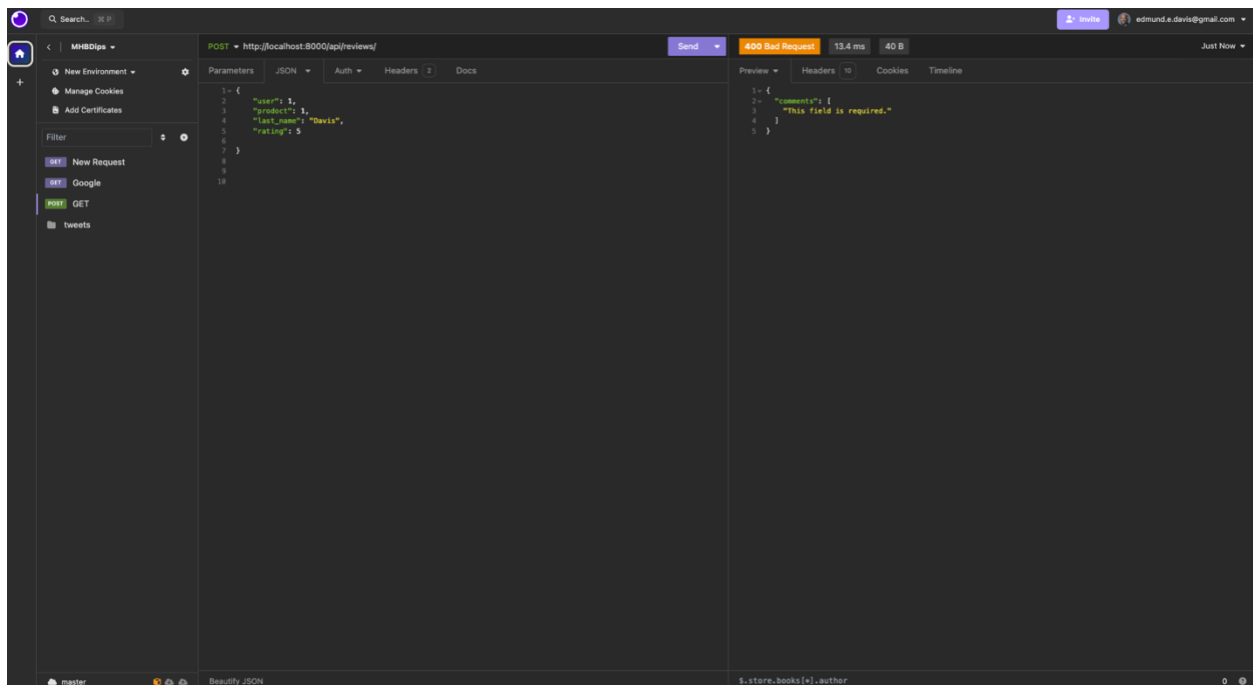
Example requests:

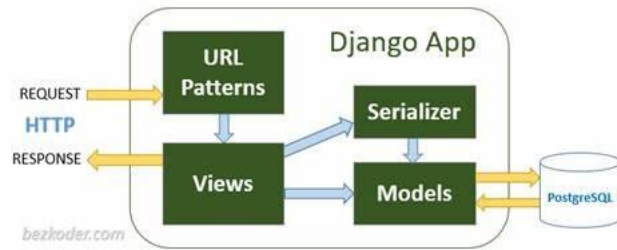
curl --request POST --url <http://localhost:8000/api/reviews/>

Success Response:



Error Response





Display Reviews:

Method: GET

URL: <http://localhost:8000/api/reviews/>

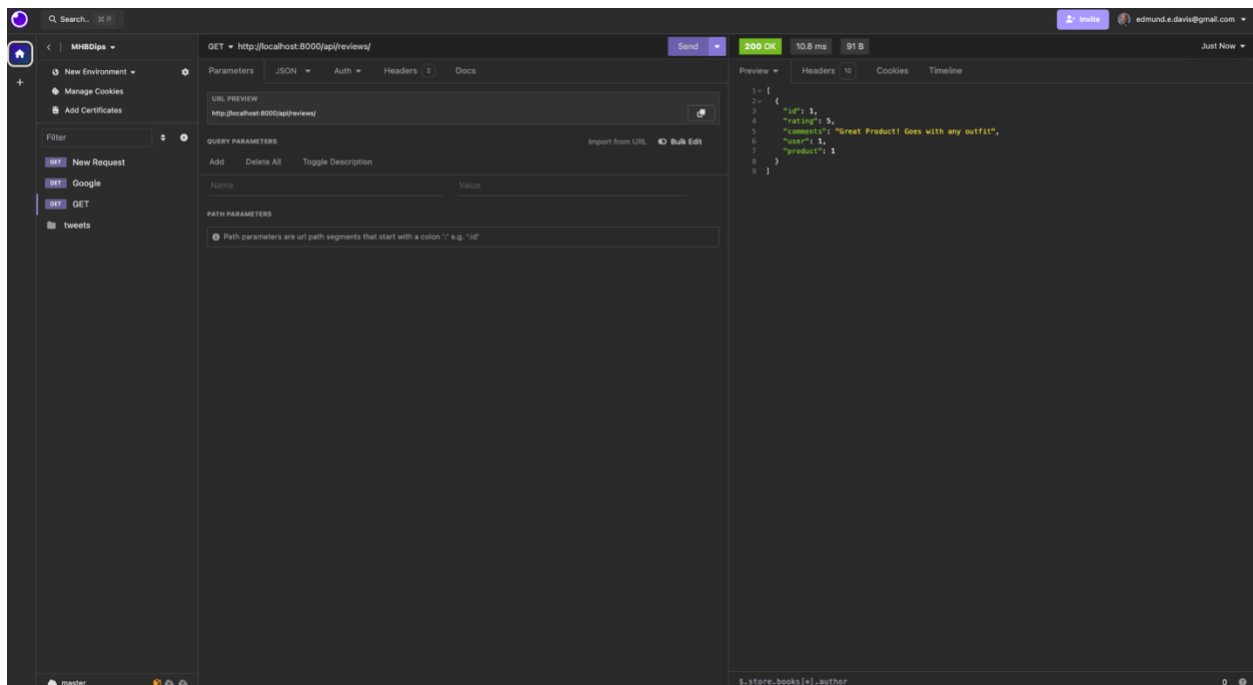
Purpose:

When a user visits a product page. A GET method will be sent to the backend to request the reviews for that product.

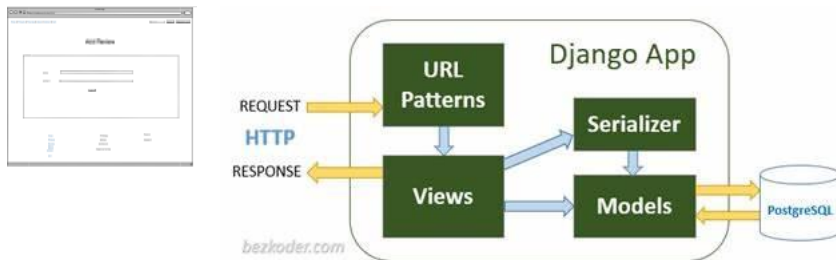
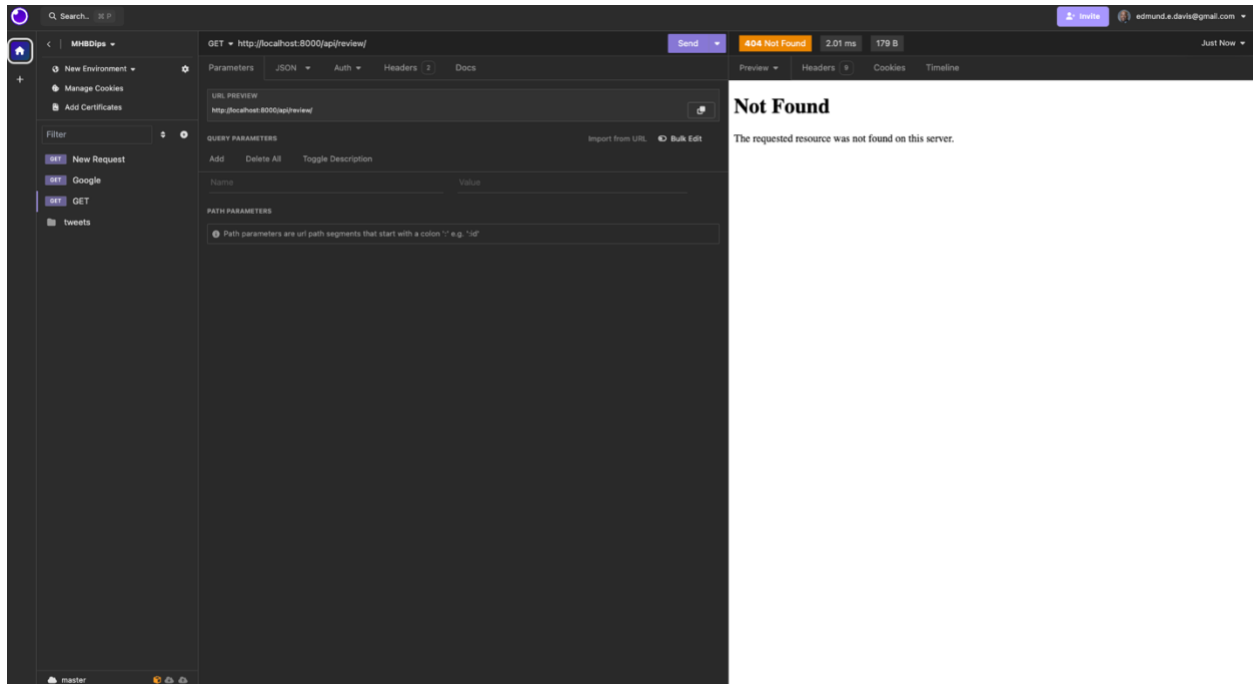
Example requests:

`curl --request GET --url http://localhost:8000/api/reviews/1`

Success Response:



Error Response:



OrderDetails:

Add Order Details:

Method: POST

URL: <http://localhost:8000/api/orderdetails/>

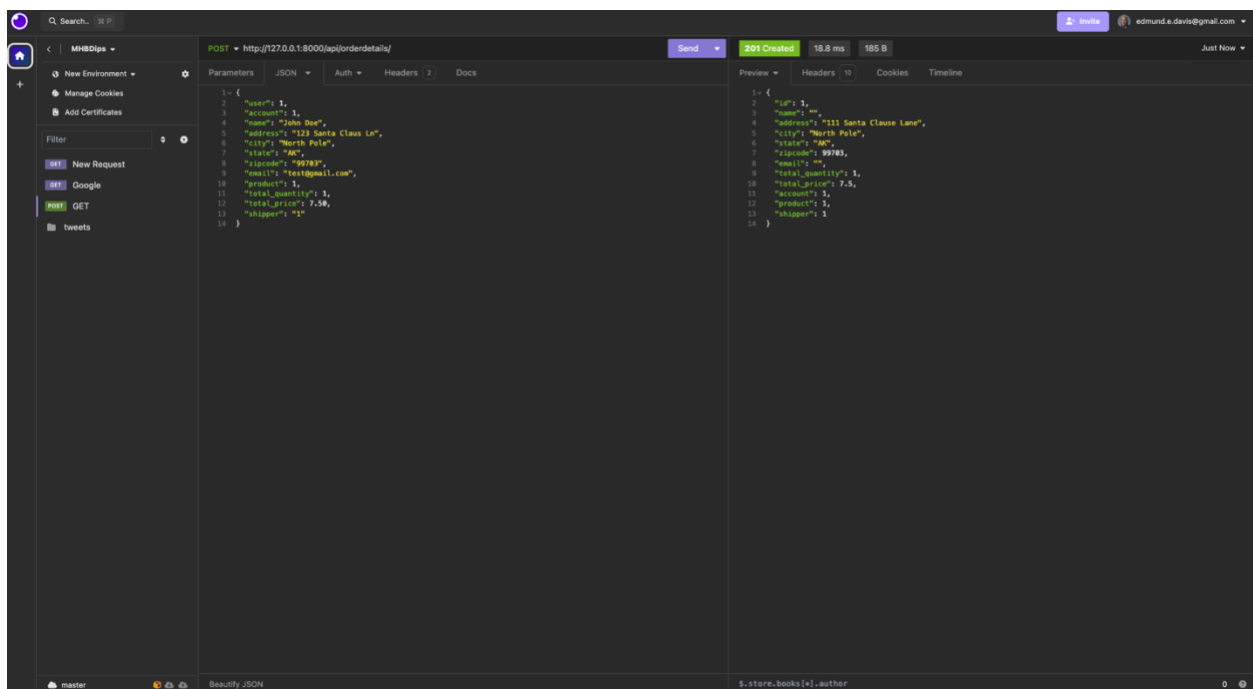
Purpose:

When user clicks add to cart. Their order details will be sent to the order details table. This information will eventually be used in a GET request to display to the cart. Which has not been implemented yet.

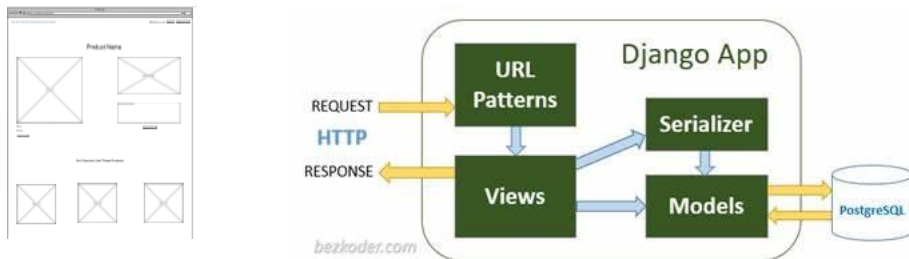
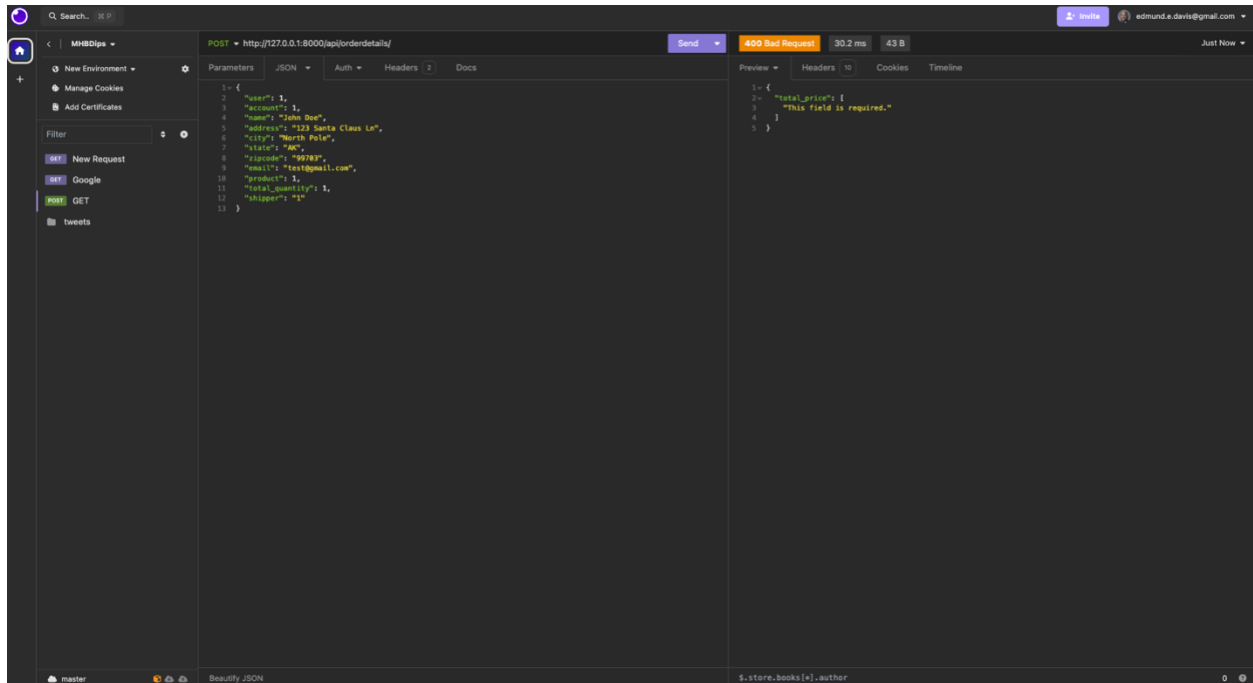
Example requests:

```
curl --request POST --url http://localhost:8000/api/orderdetails/
```

Success Response:



Error Response:



Display Order Details:

Method: GET

URL: `http://localhost:8000/api/orderdetails/`

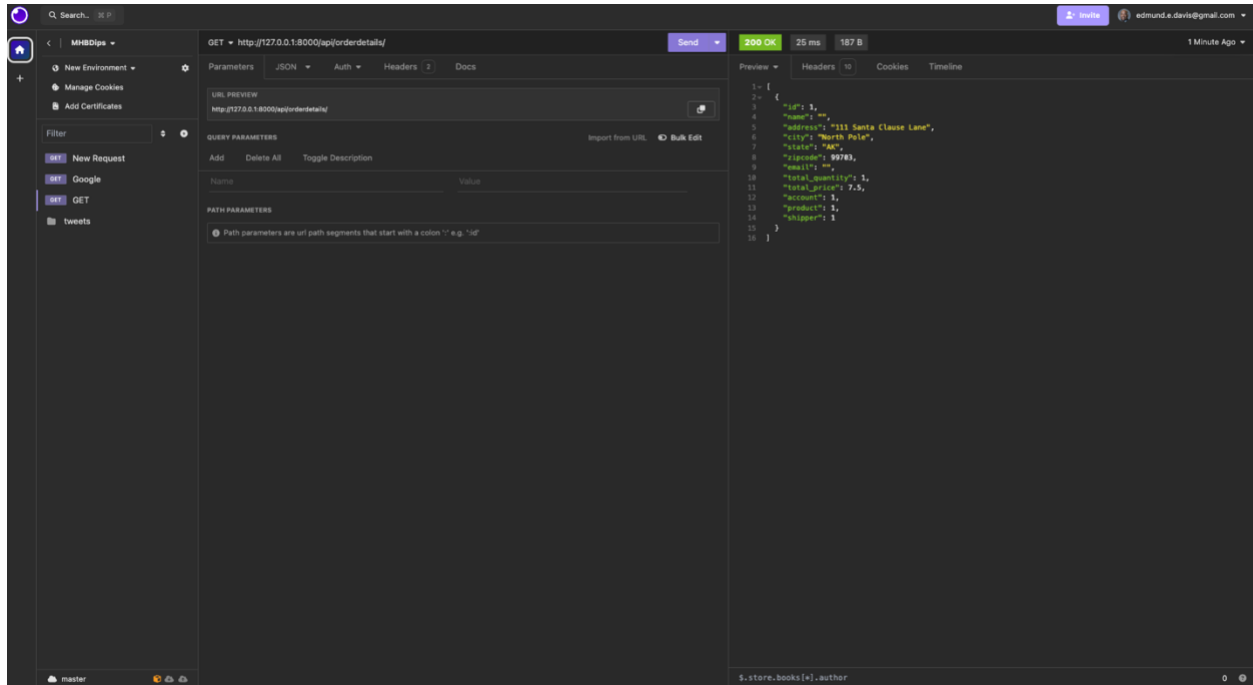
Purpose:

When a user visits the cart page. The order details will send a GET request to the backend for details about the order. The name will be populated for the user, but they will have a chance to update the address.

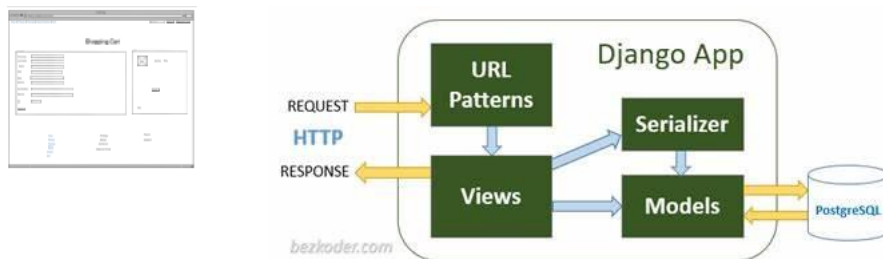
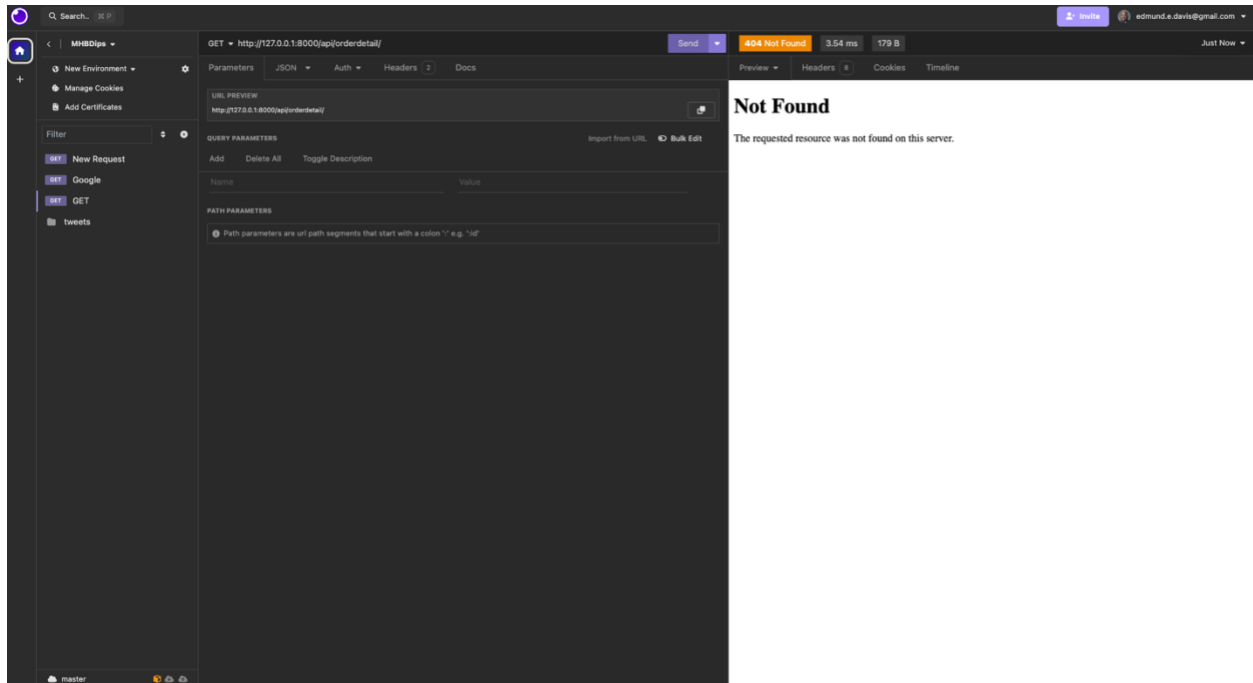
Example requests:

curl --request GET --url http://localhost:8000/api/orderdetails/

Success Response:



Error Response:



Login:

Login into account:

Method: POST

URL: <http://localhost:8000/api/login/>

Purpose:

When a user clicks login. They will be taken to a page to verify their profile. Once it is submitted, users will be taken back to the home page. Please note at the moment I am having issues with my CSRF token. That is likely due to me placing the files in the wrong place and then having to move them to the right folder structure level. My computer is taking a while to move the files, but it should be corrected by the time I put the project into production.

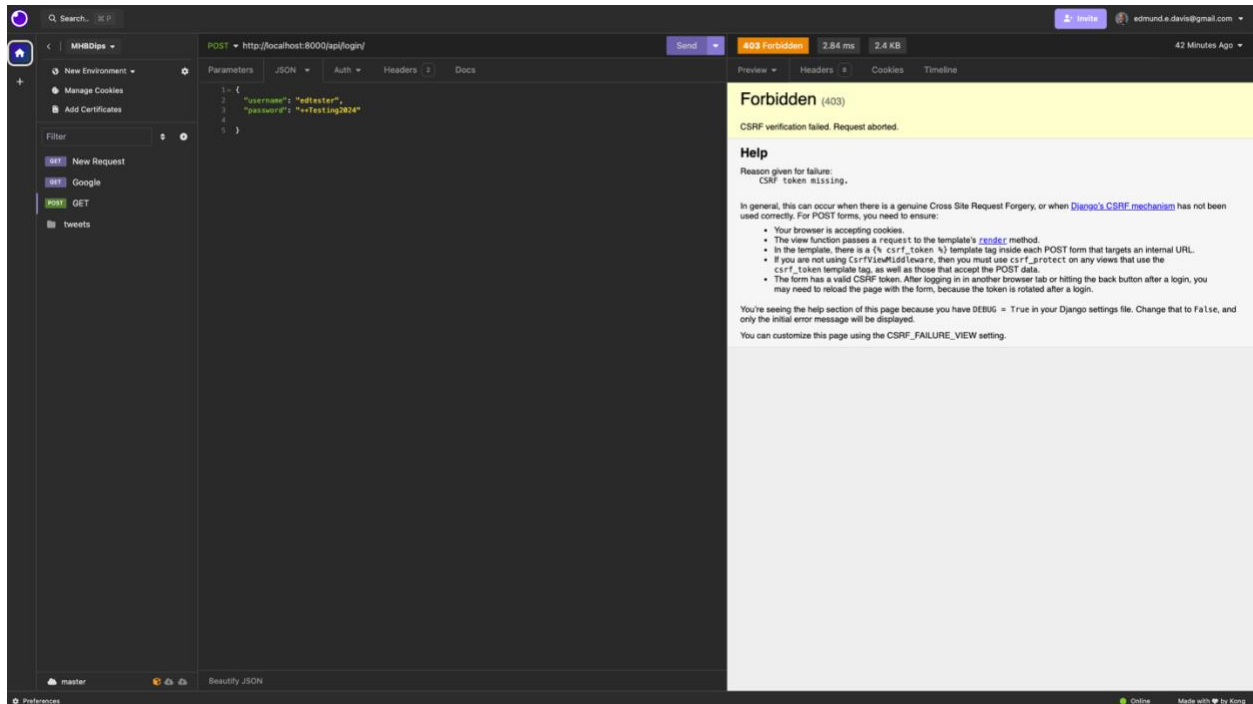
Example requests:

```
curl --request POST --url http://localhost:8000/api/login/
```

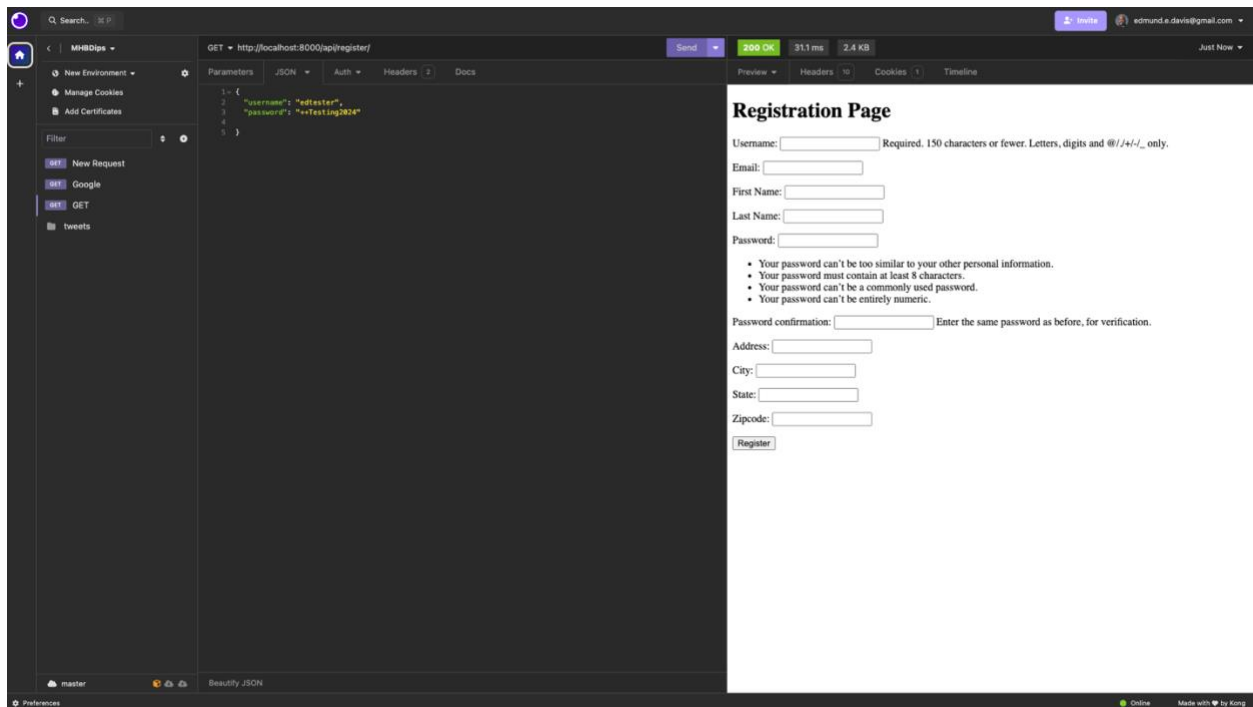
Success Response:

Currently not available due to the token issue I am working on.

Error Response:



The token issue is happening on all my forms. I have no issues retrieving the form as show below with my register.html, but I keep getting the Forbidden Error when posting all my forms. Hopefully, my computer will finish moving the files soon.



This is the form in the template:

```
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Login</button>
</form>
```

Once I can resolve this issue all of my endpoints will be done. The endpoints currently affected by this issue is my register, login, and account_update.