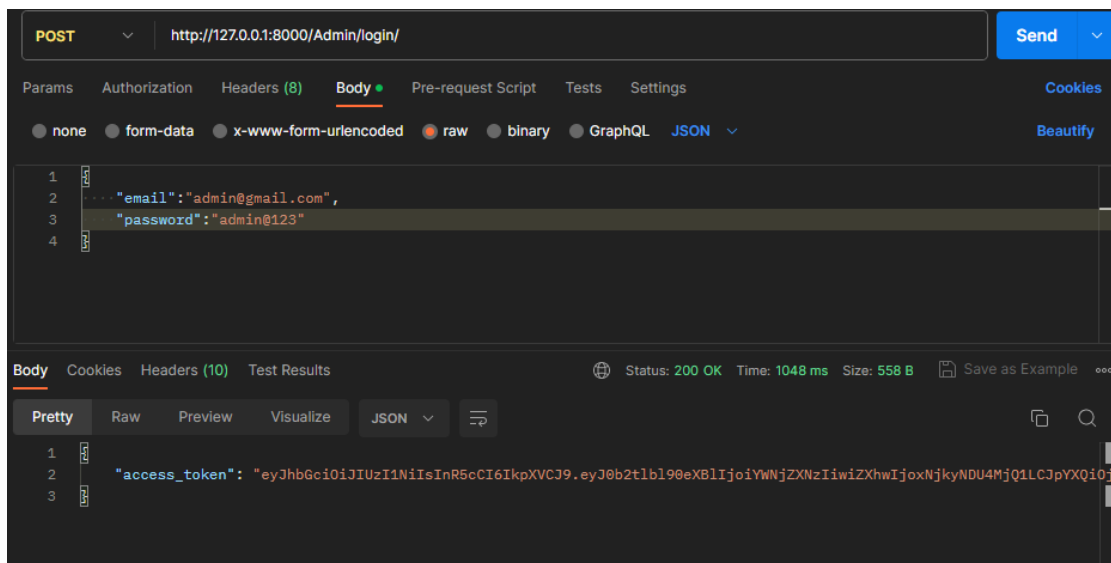# FINAL TASK -Ecommerce API

## →Admin Functionalities

→Admin login API

This API is for handling the authentication for admin users. Using email and password the admin can login to the system. If the credentials are valid and user is admin the API generates an access token for the user.
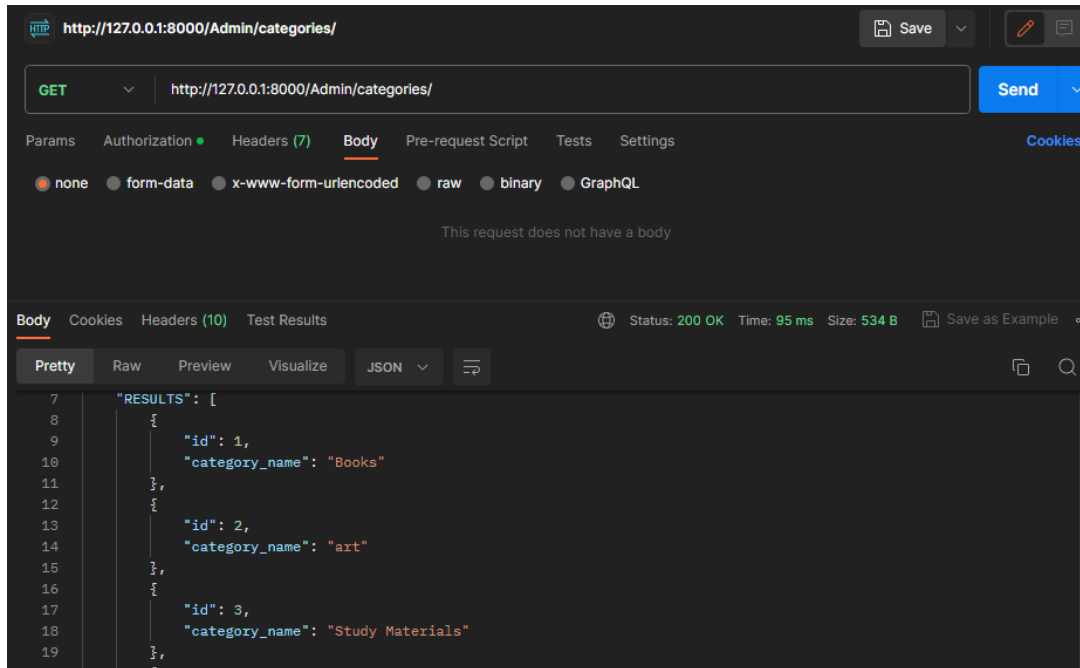

→Admin Category Add API

This API is for admin to add categories to the system .Admin can send a POST request with the name of the category he wants to add.
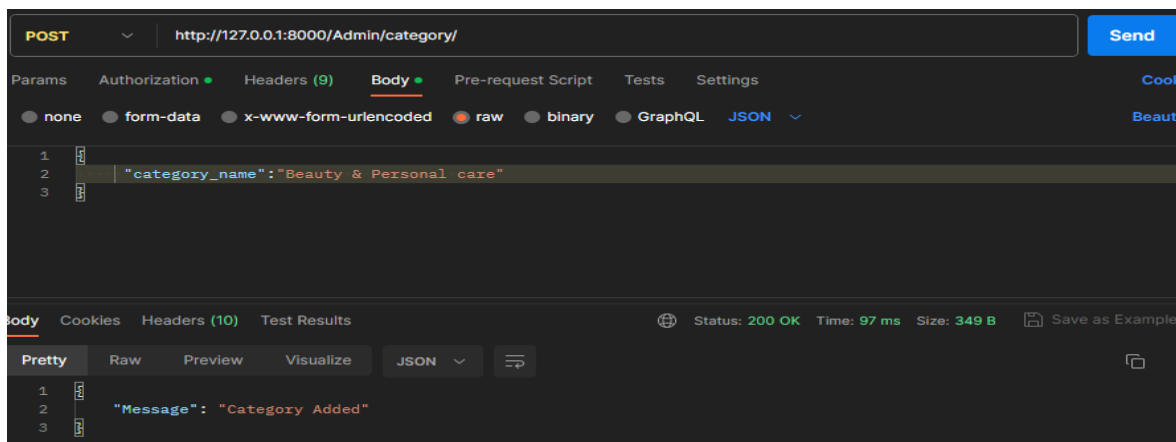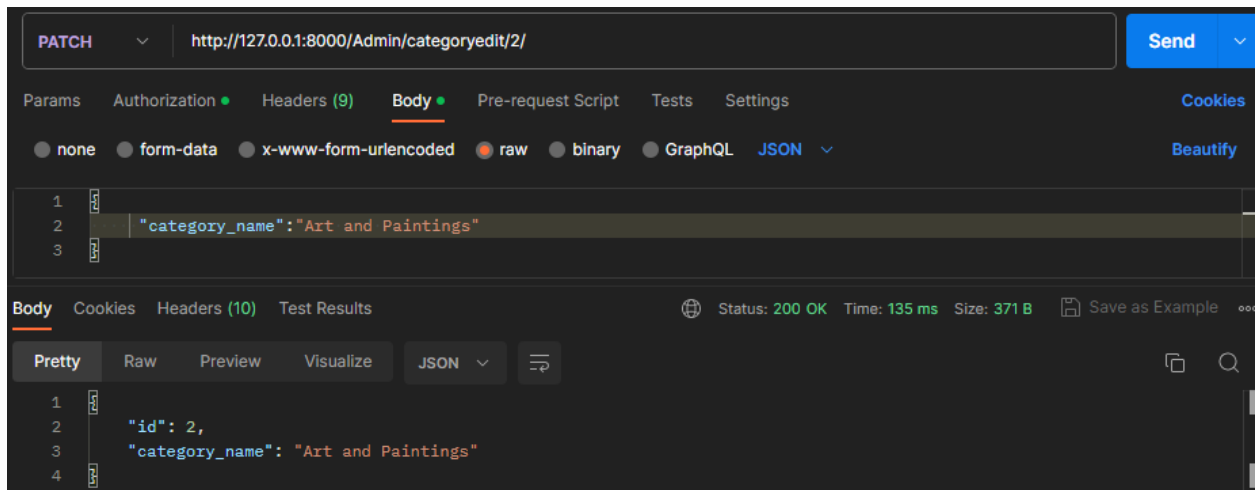
→Category view API

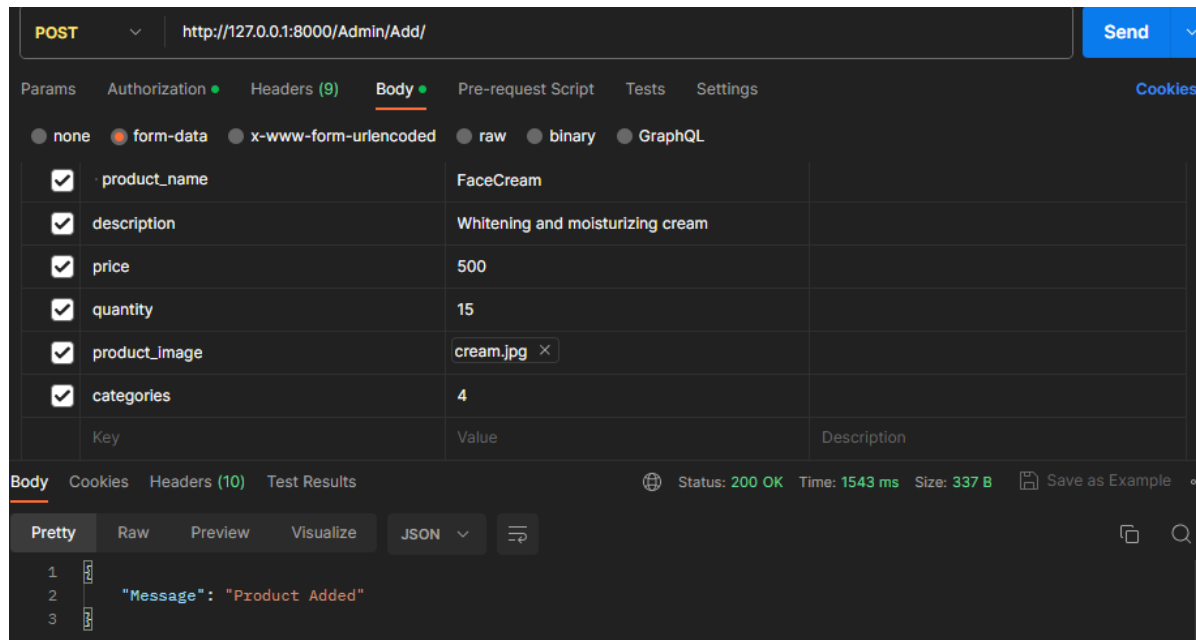This API is for admin to list the available categories

→Category Edit API

This API allows only admins to retrieve and update categories using a put or patch request.
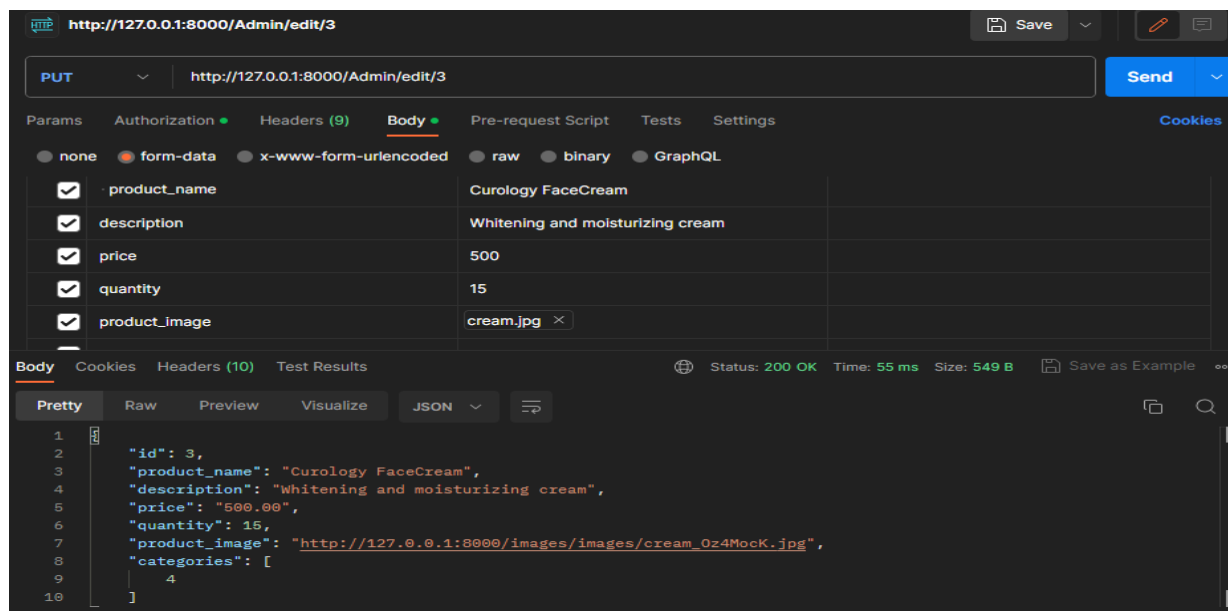
→Admin Product Add API

This API is allows admin to add new product to the system, by sending a POST request with details of the product they want to create.
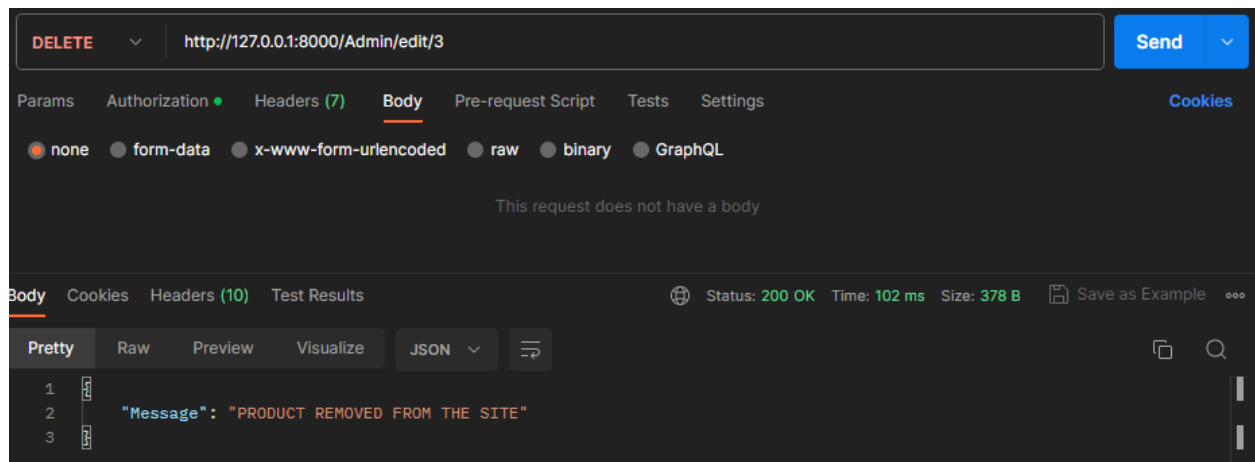

→Admin Product Edit API

This API allows the admin to update and delete product. The product details can be updated using PUT or PATCH request.
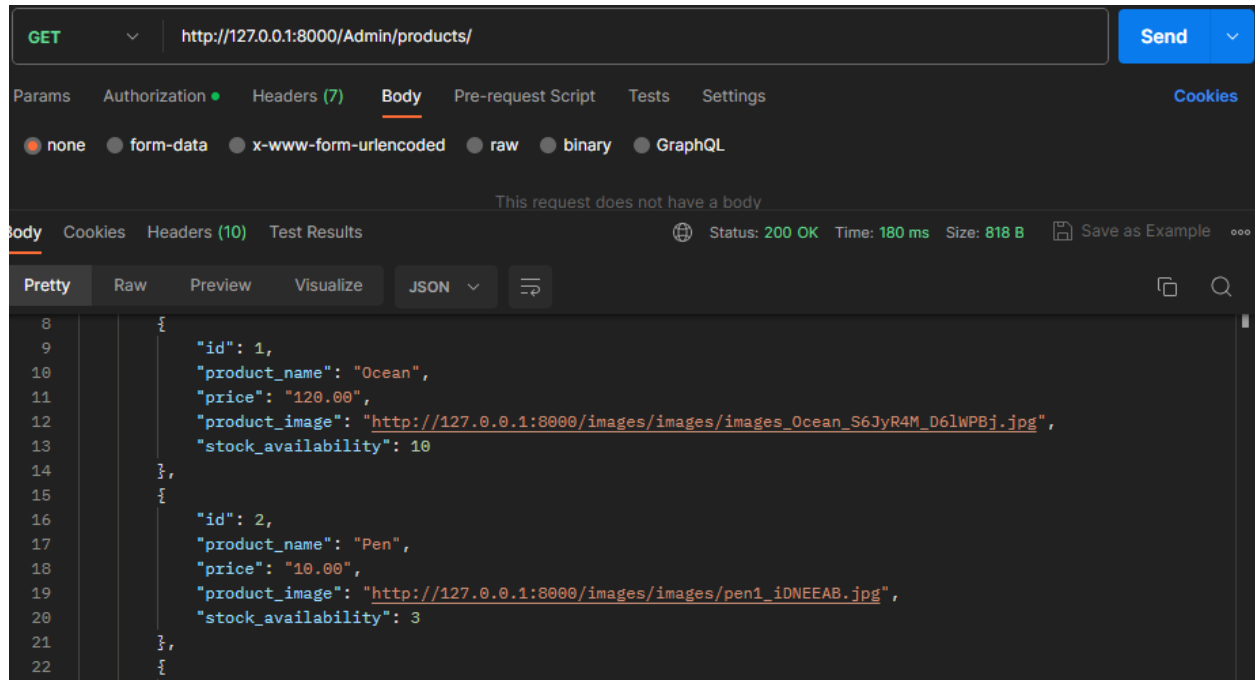
→Deleting a product from system by admin


```
{
    "Message": "PRODUCT REMOVED FROM THE SITE"
}
```

→Admin Product View API

This API allows the admin to get the list of all available products and apply filters based on categories, names and price.


```
{
    "id": 1,
    "product_name": "Ocean",
    "price": "120.00",
    "product_image": "http://127.0.0.1:8000/images/images/images_Ocean_S6JyR4M_D6lWPBj.jpg",
    "stock_availability": 10
},
{
    "id": 2,
    "product_name": "Pen",
    "price": "10.00",
    "product_image": "http://127.0.0.1:8000/images/images/pen1_iDNEEAB.jpg",
    "stock_availability": 3
},
{
```

→Admin Product Retrieve API

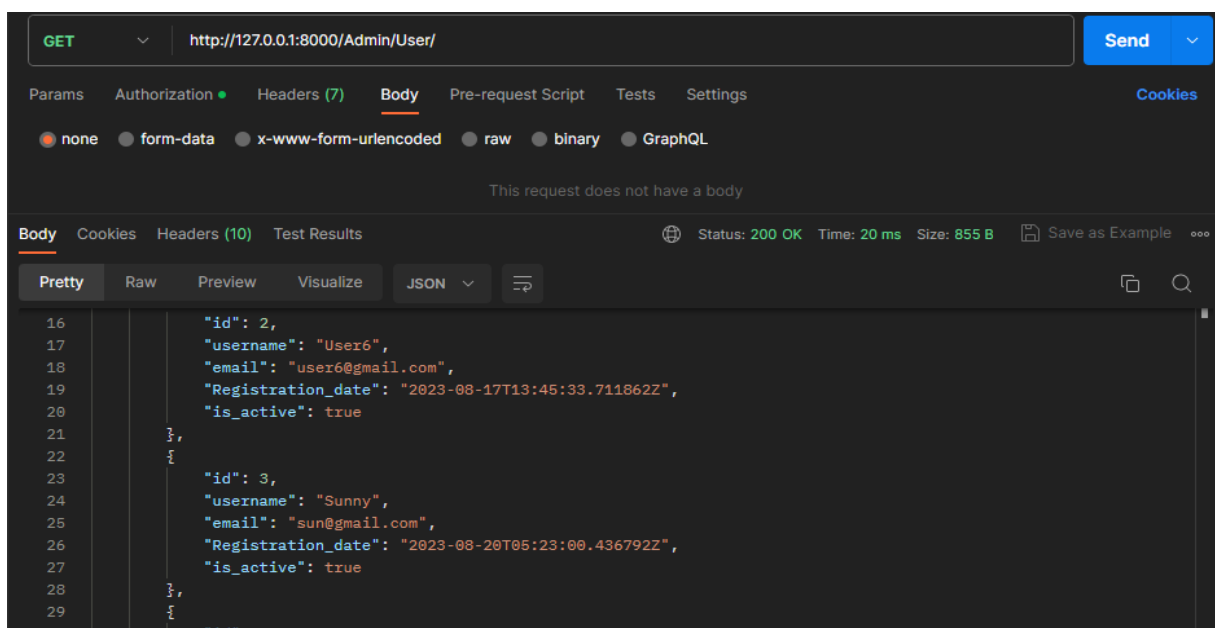This API allows the admin to retrieve detailed information about a specific product. Admins can access the detailed information of a product by providing the product's id in the request.
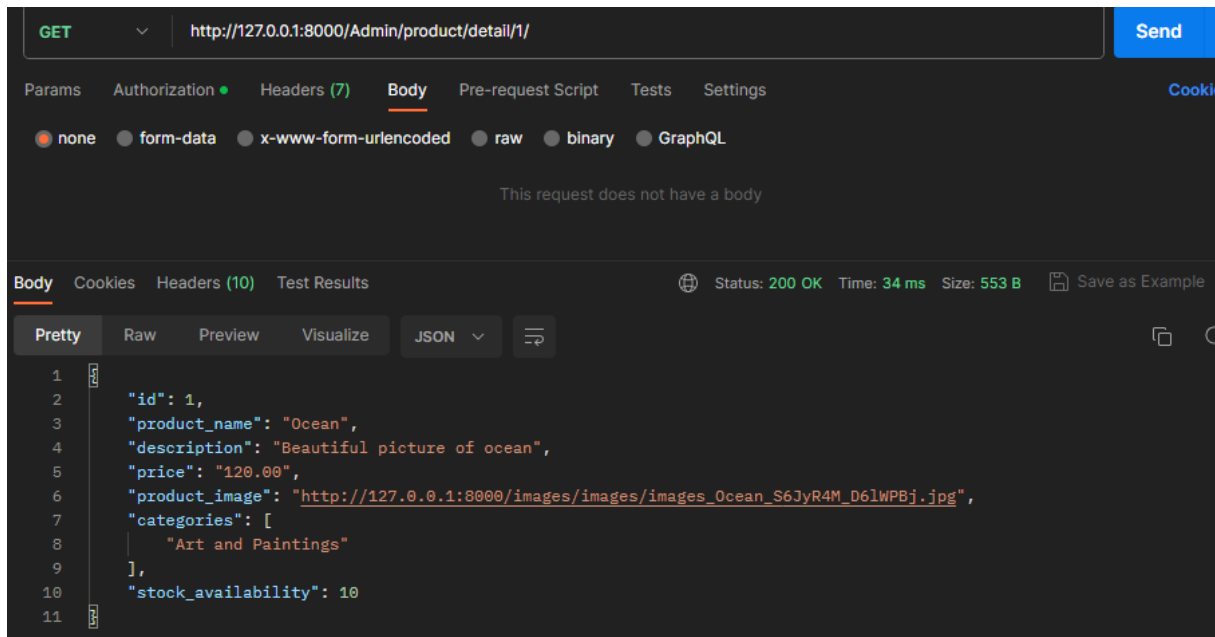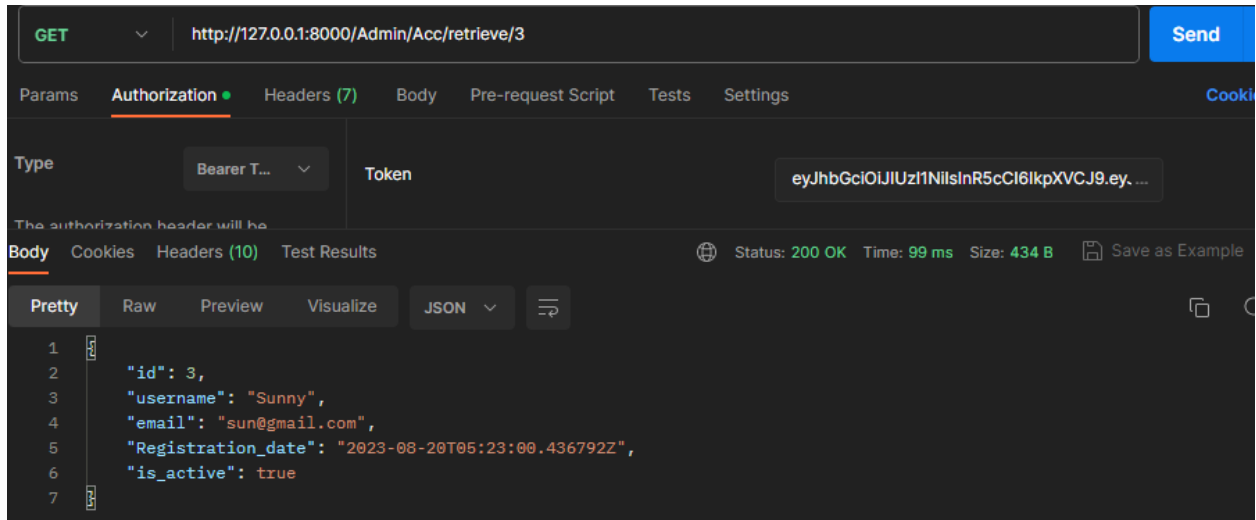

→User List View API

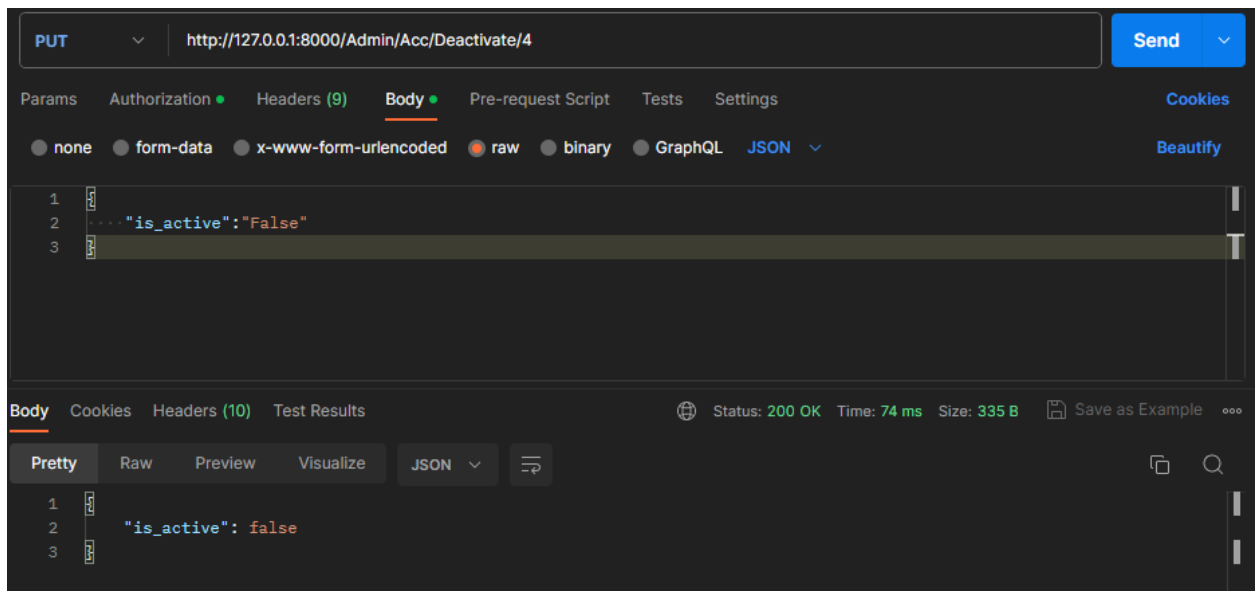This API allows only the Admin to view the list of user accounts.

→User Retrieve API

This API allows admin to retrieve details of a specific user.The admin can get the details of a specific user by providing the user's id in the request.

→User Deactivate API

This API allows admins to deactivate a specific user account. Admins can set the 'is_active' attribute of user account to False for deactivating the account.
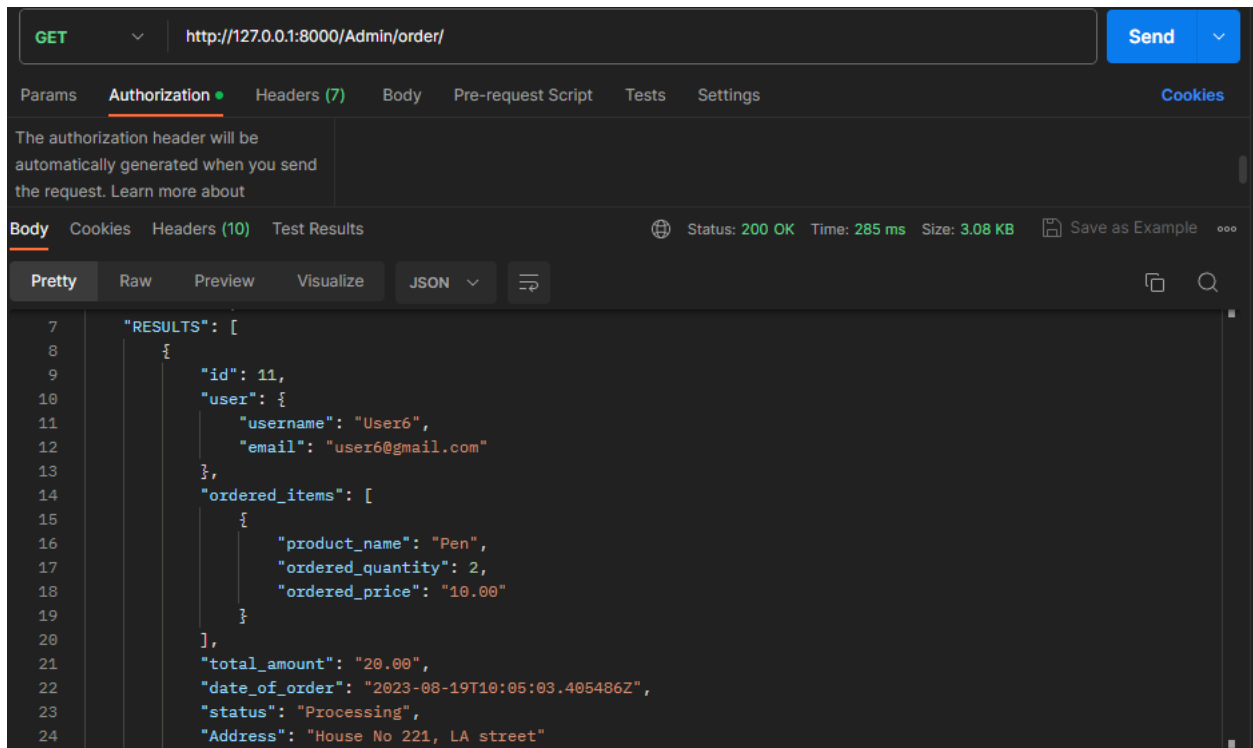
→User Delete API

This API allows Admins to delete a specific user account. User id should be provided in the url to identify the user account that needs to be deleted.
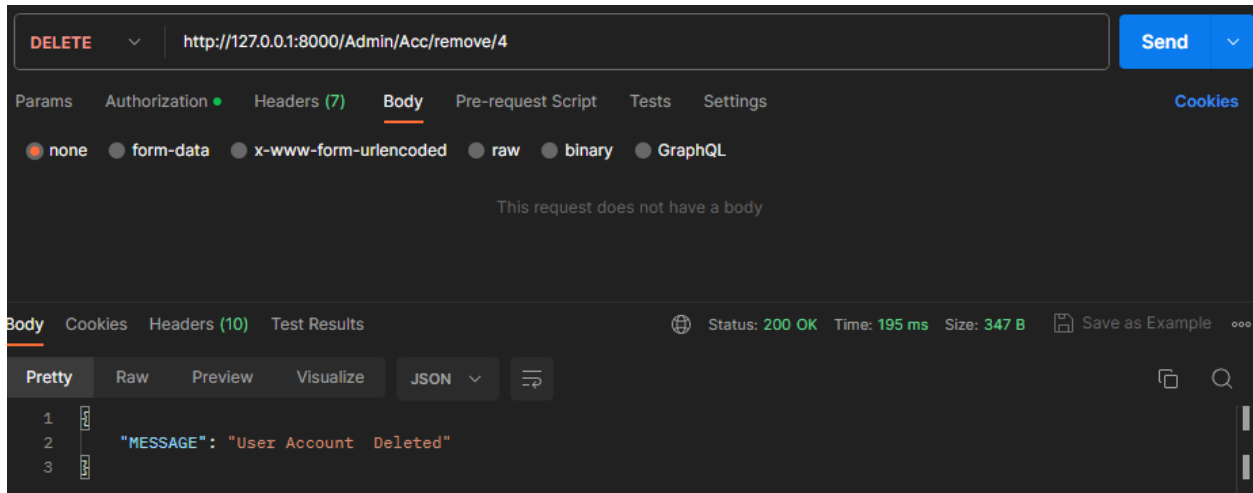
→Admin Order View API

This API allows the admin to view the list of orders placed by the users. The orders are sorted by the date of order in ascending order.

→Order Status Edit API

This API allows admin to update the status of an order using PATCH method. After updating the status the API sends an email notification to the mail id of the user who placed the order. Order id should be provided in the URL to identify the specific order for which the status need to be updated.
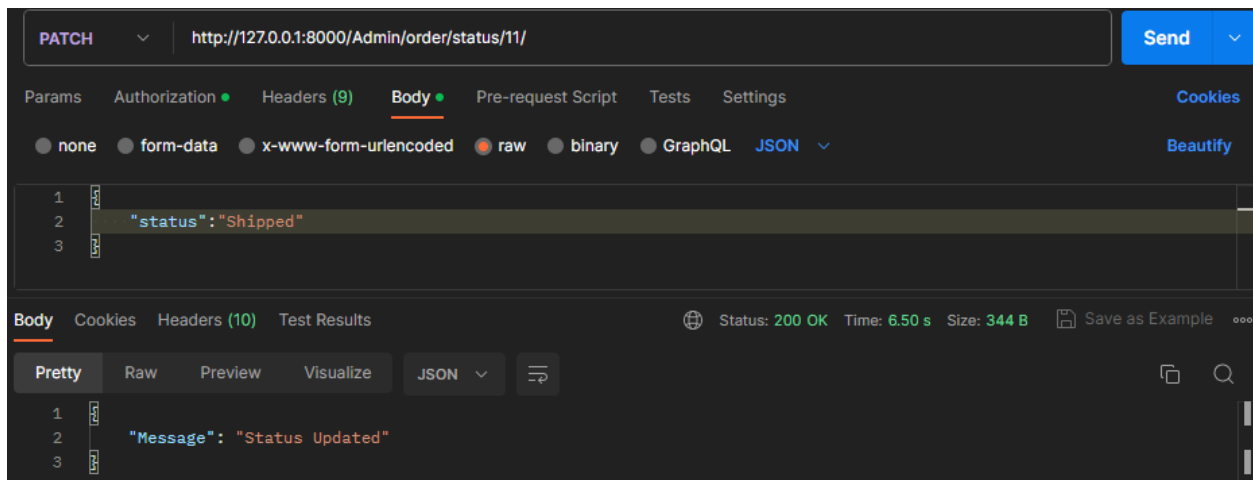
→Email received for status update
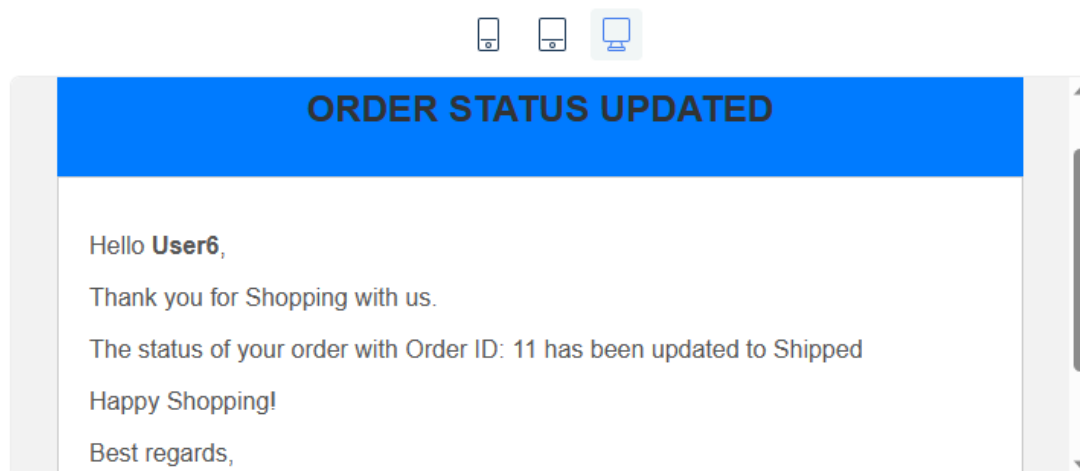
→Promotional Email View API

This API allows the admin to send promotional emails to the registered users using POST method. The email content is taken from serializers validated data.The email is sent using send_email function.
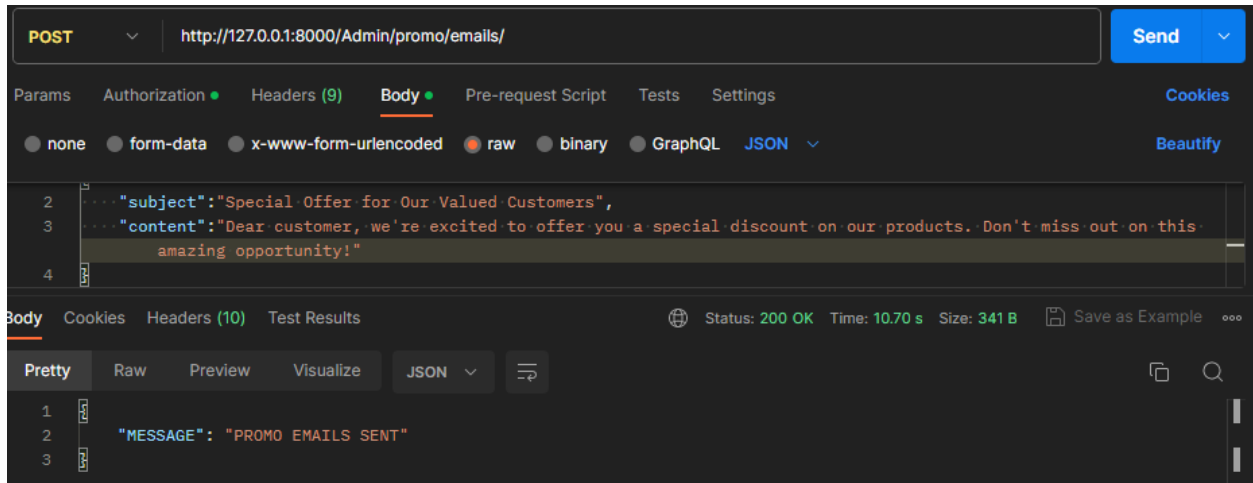
→User functionalities

→Signup API

This API handles user registration. Upon successful registration this API sends a email to the registered mail id of the user. Email, password and username are needed for registration.
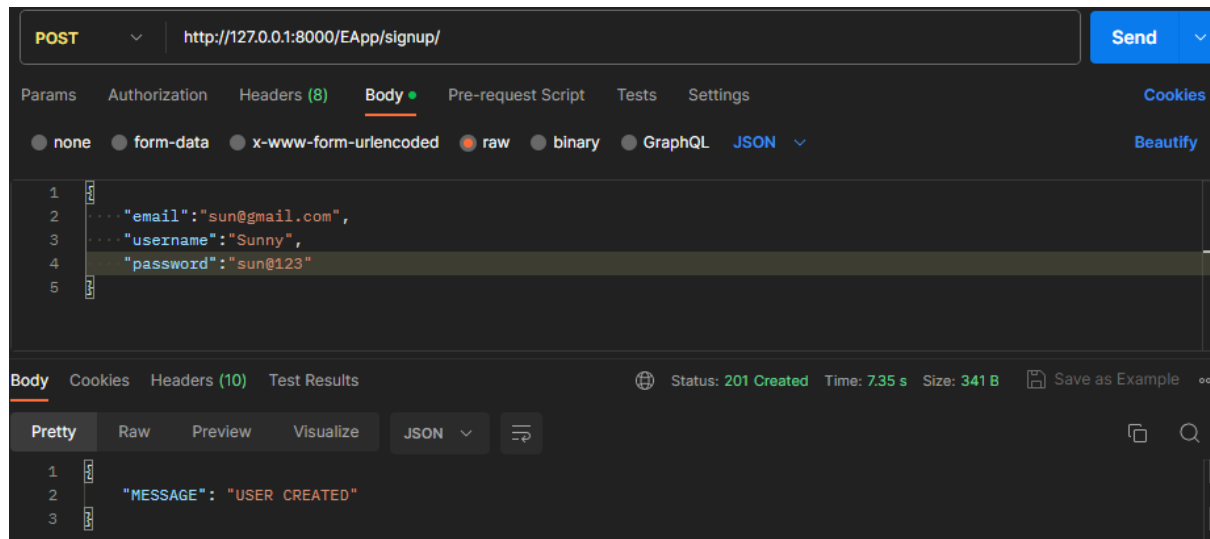
→User Password Reset API

This API handles the process of sending a password reset email to a user's email address. A POST request is made using the email. If the email is registered then default token generator generates a token, using this token and user's primary key reset URL is created.
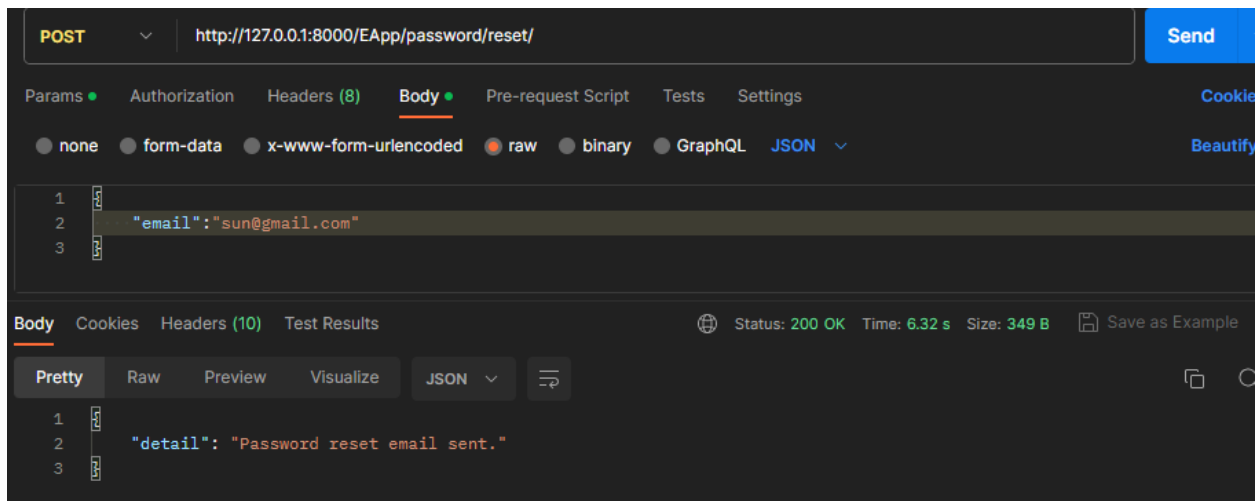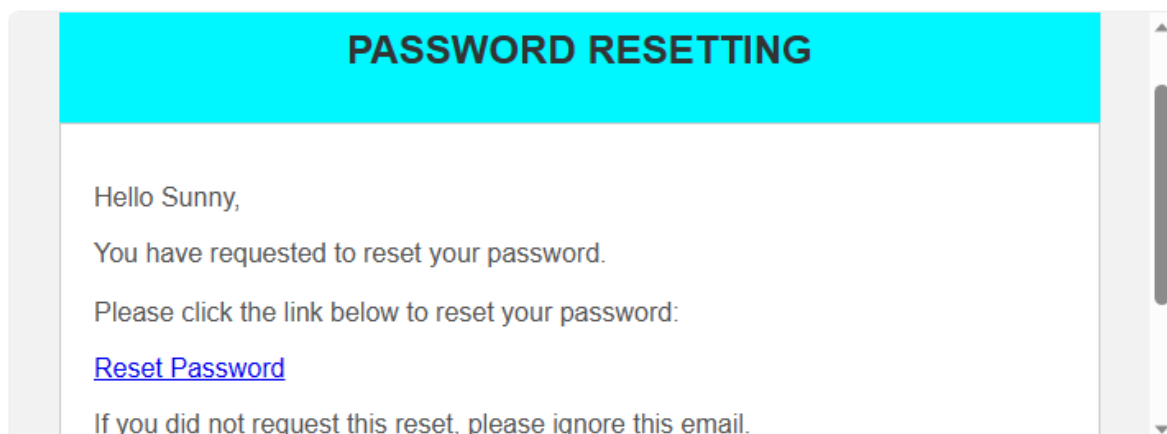
→Password Reset Confirm API

This API handles the confirmation of password reset for the user. A POST request is made with UID and token from the reset link and new_password and confirm_password fields. UID is decoded to get the pk. If the token is valid and the user is found using pk then API checks whether the new_password and confirm_password match. If they match new password is saved.
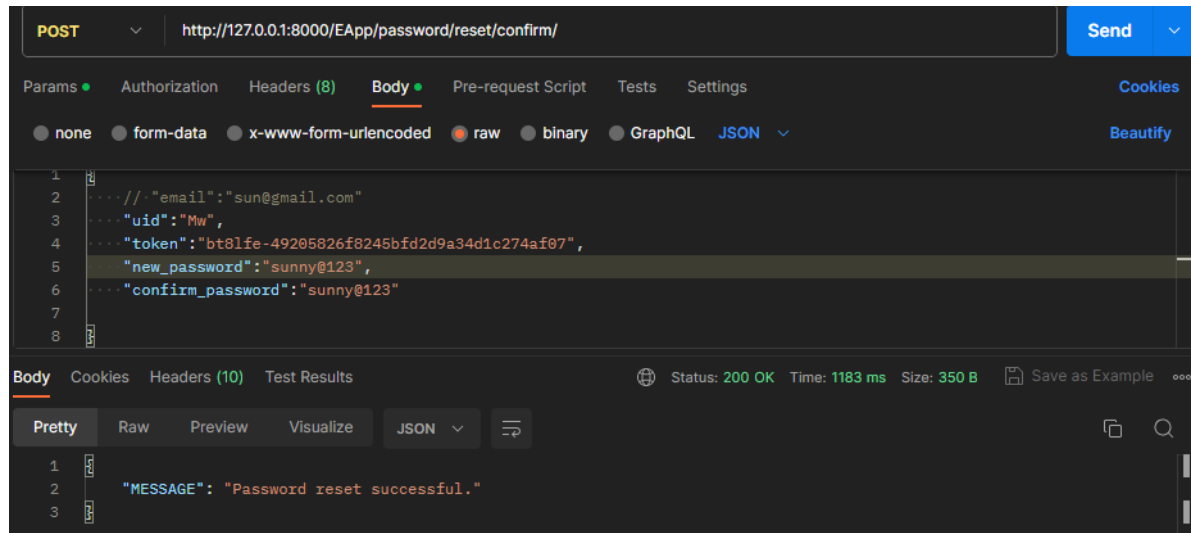
→Product List API

This API handles the listing of available products. Filtering is also applied. The user can filter the products based on category name and price and product name.
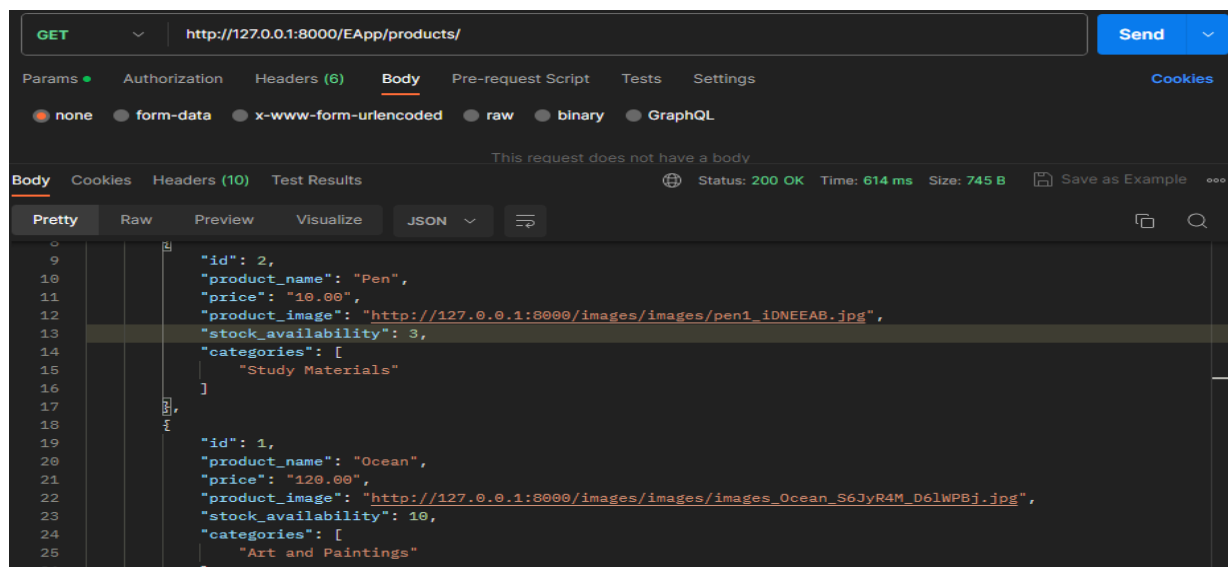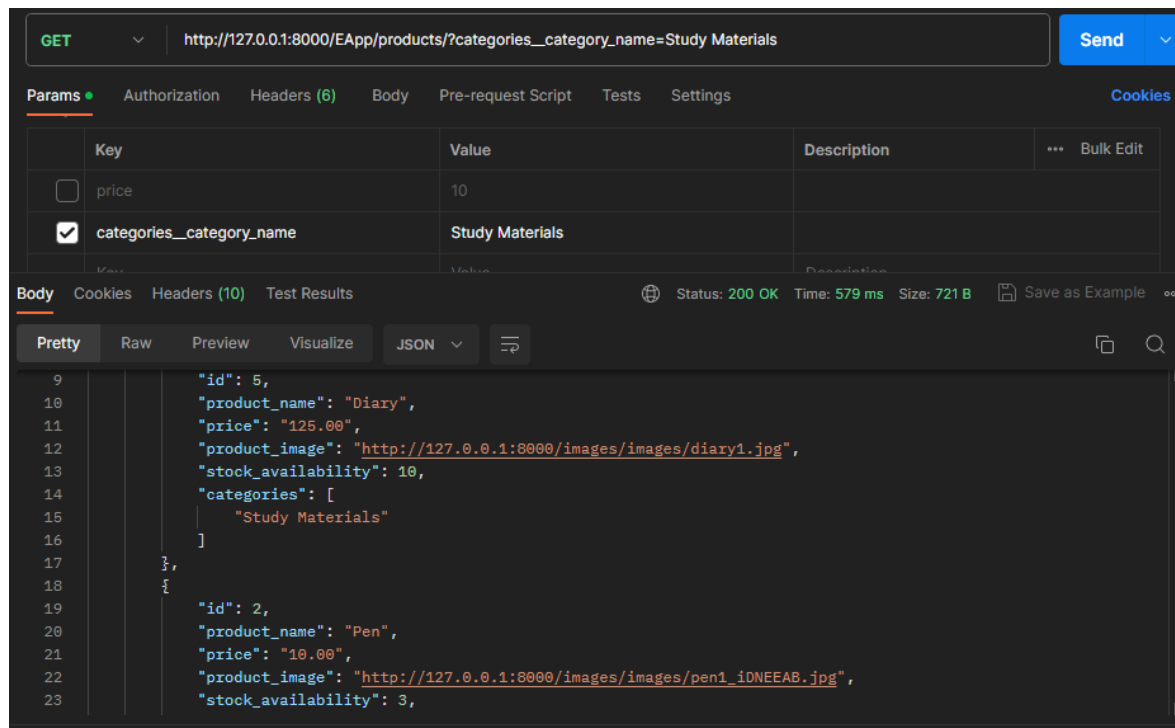
→Product Retrieve API

This API allows users to retrieve information about a single product. The product id needs to be passed along with the URL in the get method for retrieving product information.
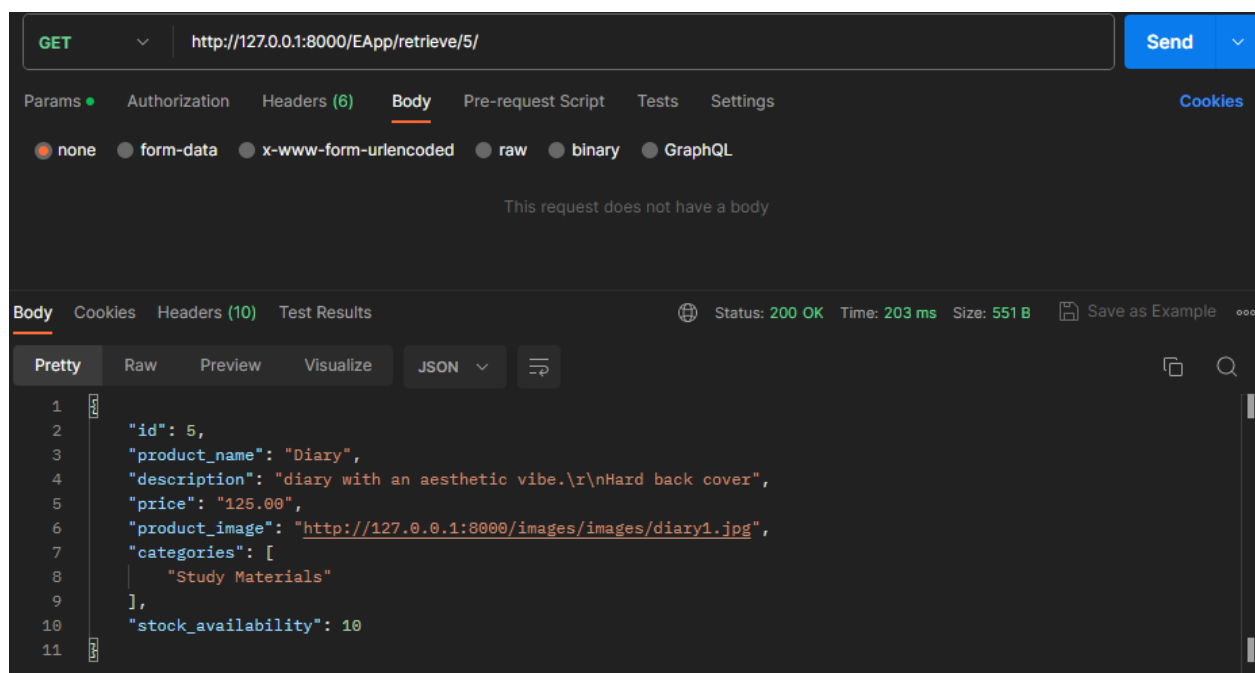
→Add Item to Cart API

This API allows authenticated users to add items to their shopping carts. The product and Number Of items (optional) need to provide along with the GET method. The API check whether the product is in stock, and whether is it available in the cart, if the product exists in the cart, it won't be added again else it will be added to the cart, later the user can edit the count of items required.
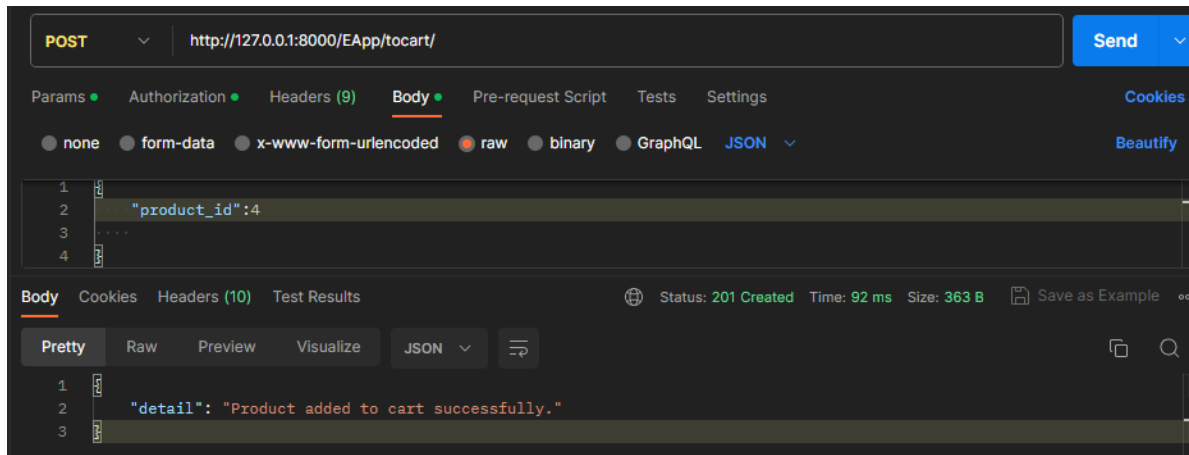
→Cart Item List API

This API handles the listing of the items in the cart of an authenticated user.
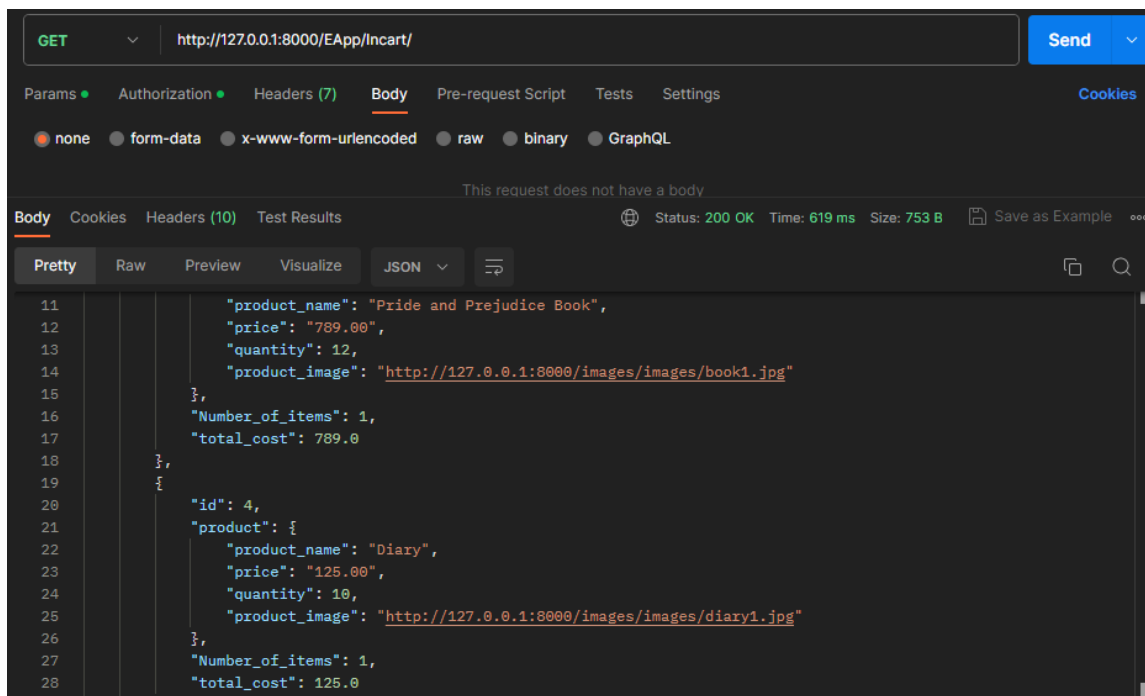
→Cart Item Retrieve API

This API allows retrieving detailed information about a specific item in the user's cart. Cart Item id needs to be passed in the URL along in the Get Method
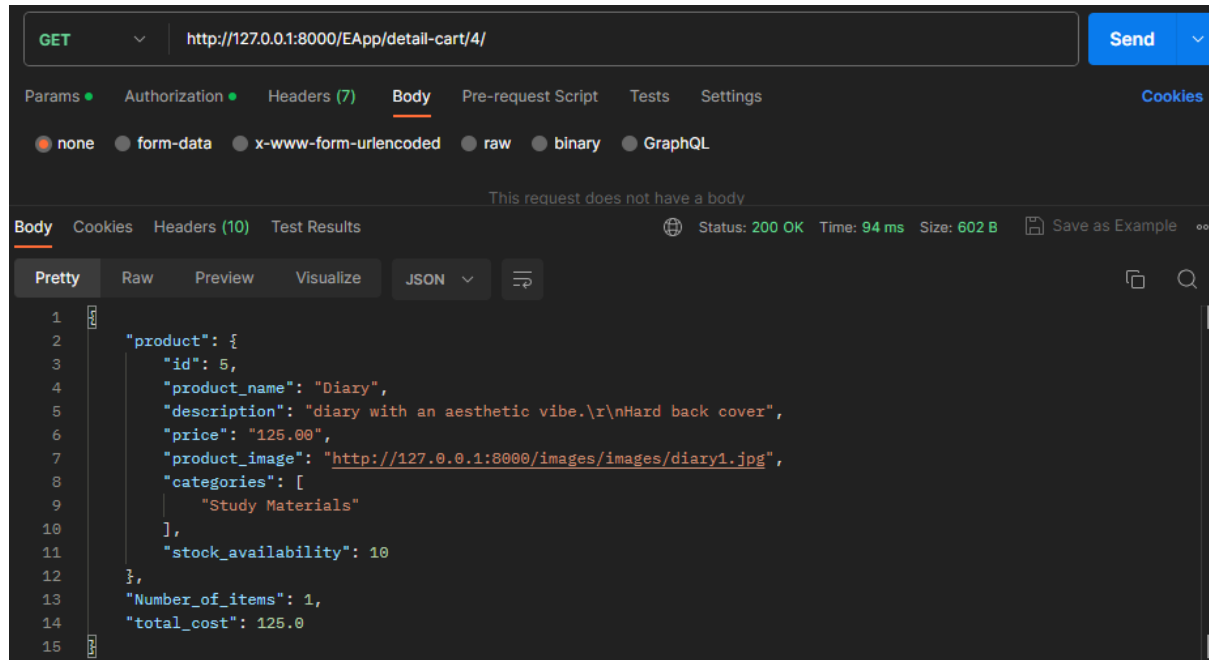
→Cart Item Update API

This API allows to update the Number of items for a specific item in the user's shopping cart using the PATCH method.
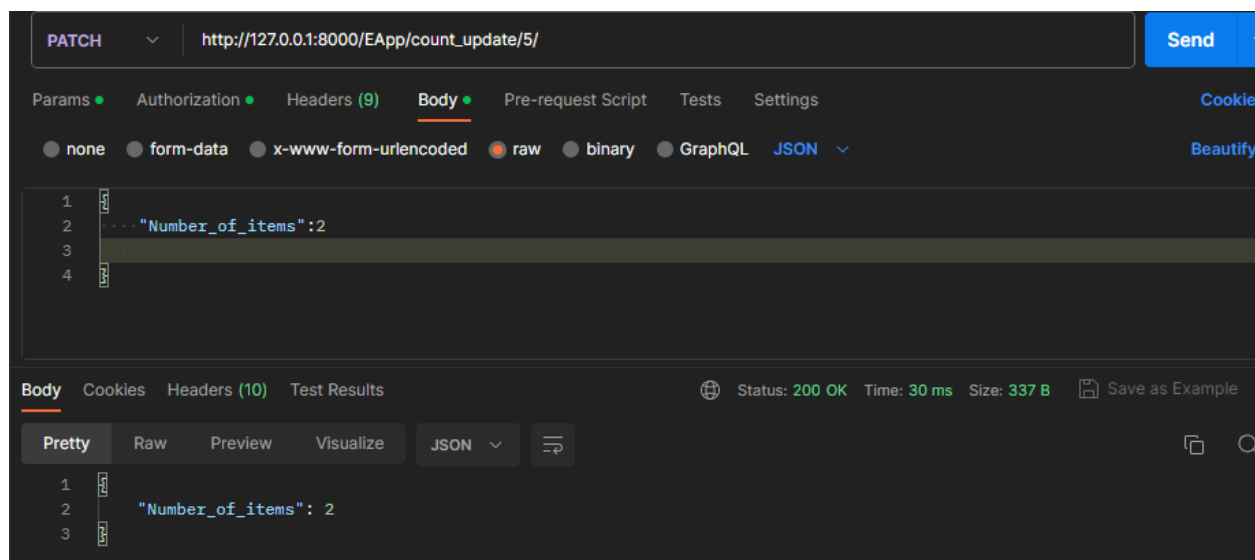
→Cart Item Delete API

This API allows the user to remove the items from their own cart.
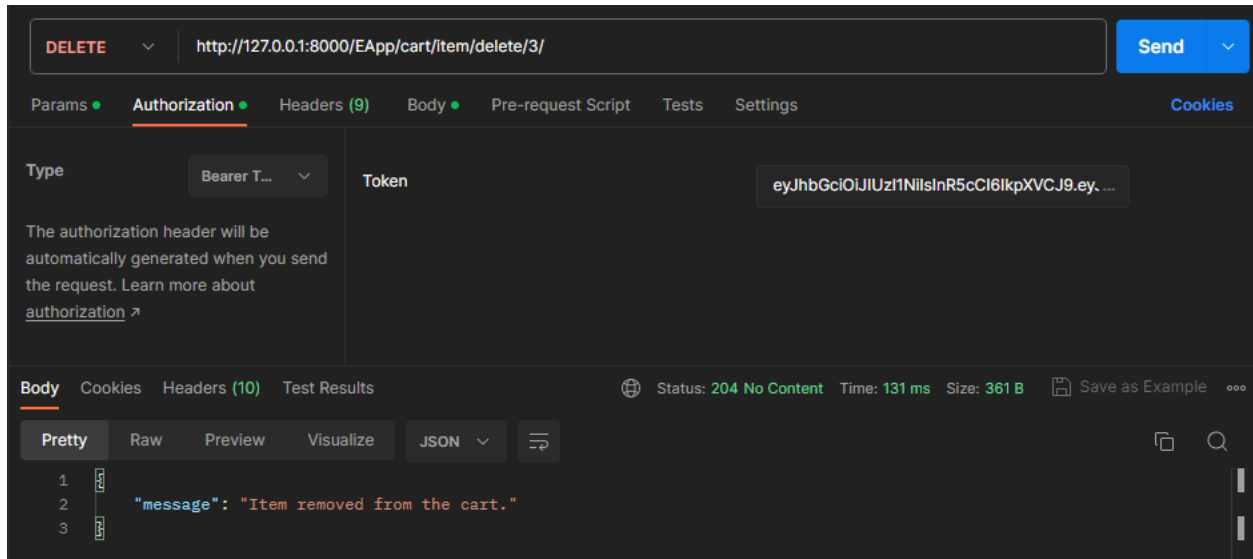
→Order Placement API

This API allows the user to place an order for the items available in the cart. When a user submits an order request, the API first validates the provided data. This includes the list of cart item IDs, shipping address, and payment method. It then checks if the cart items belong to the user's cart. If all items are present in the cart, the API calculates the total order amount by multiplying the quantity of each item by its price. It also sends confirmation emails to both the admin and the user when an order is placed.

```
POST    http://127.0.0.1:8000/EApp/order/place/          Send

Params   Authorization •   Headers (9)   Body •   Pre-request Script   Tests   Settings        Cookies
none   form-data   x-www-form-urlencoded   raw   binary   GraphQL   JSON        Beautify

1  {
2      "cartitem_id":[5,6],
3      "Address":"House No 321,garden street NYC "
4  }

Body   Cookies   Headers (10)   Test Results        Status: 200 OK   Time: 9.77 s   Size: 336 B   Save as Example

Pretty   Raw   Preview   Visualize   JSON

1  {
2      "Message": "order placed"
3  }
```

Email received by user

Order Confirmation

From: <ecommerec@gmail.com>   2023-08-20 13:21, 3.8 KB
To: <amy@gmail.com>

Show Headers

HTML   HTML Source   Text   Raw   Spam Analysis   HTML Check 5   Tech Info

ORDER CONFIRMATION

Hello **Amy**,

Your new order has been successfully placed.

Order ID: 15

Total Amount: 1039.00

Ordered Products:

Order Confirmation

HTML	HTML Source	Text	Raw	Spam Analysis	HTML Check 5	Tech Info

Product ordered: Diary

Price of the product: 125.00

Number of Diary ordered: 2

Product ordered: Pride and Prejudice Book

Price of the product: 789.00

Number of Pride and Prejudice Book ordered: 1

Status: Processing

Email received by admin

Order Confirmation

HTML	HTML Source	Text	Raw	Spam Analysis	HTML Check 5	Tech Info

ORDER CONFIRMATION

Hello **Admin**,

A new order has been successfully placed by **Amy**.

Order ID: 15

Total Amount: 1039.00

Ordered Products:

Order Confirmation

From: <ecommerec@gmail.com>
To: <admin@gmail.com>

2023-08-20 13:21, 3.8 KB
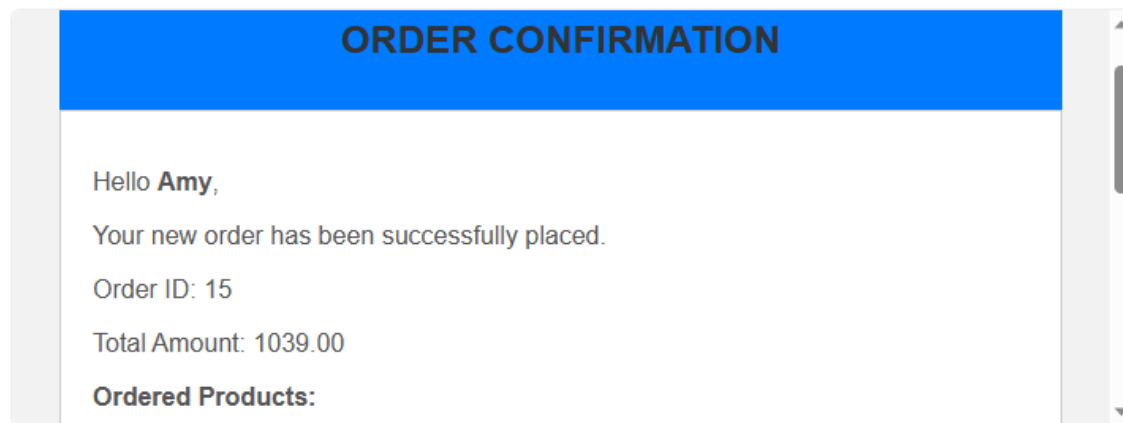
Show Headers

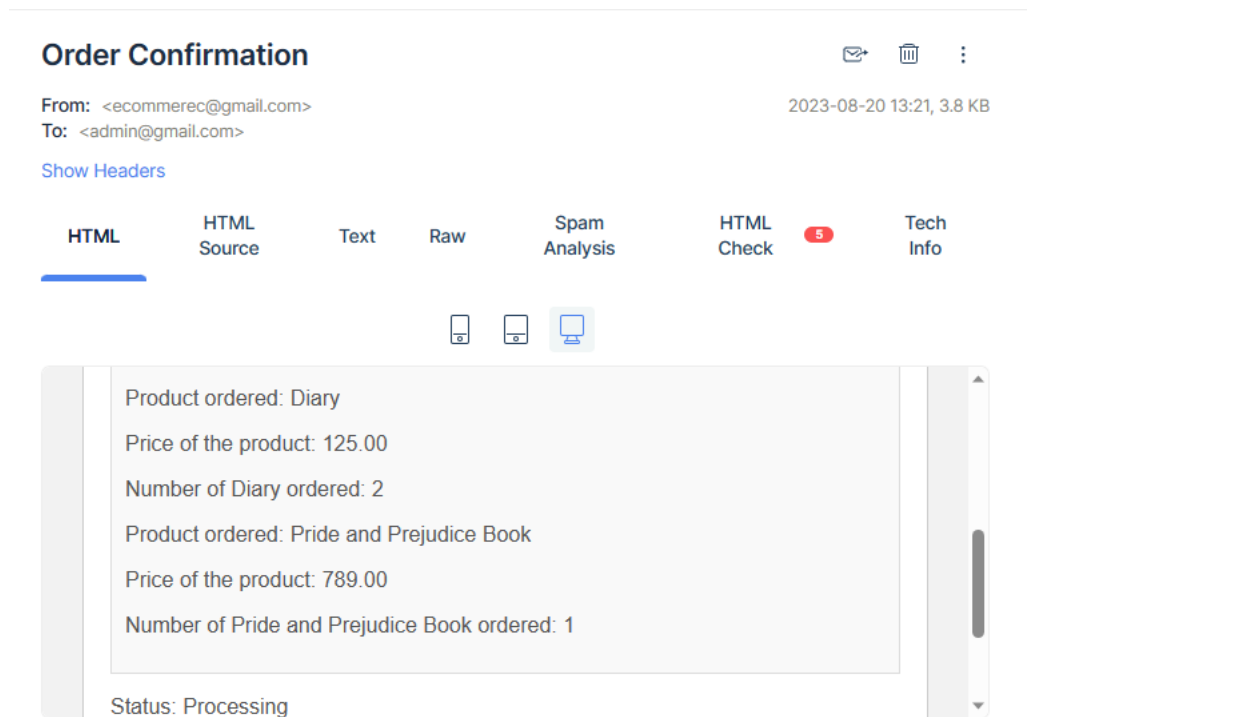| HTML | HTML Source | Text | Raw | Spam Analysis | HTML Check | 5 | Tech Info |

Product ordered: Diary

Price of the product: 125.00

Number of Diary ordered: 2

Product ordered: Pride and Prejudice Book

Price of the product: 789.00

Number of Pride and Prejudice Book ordered: 1

Status: Processing
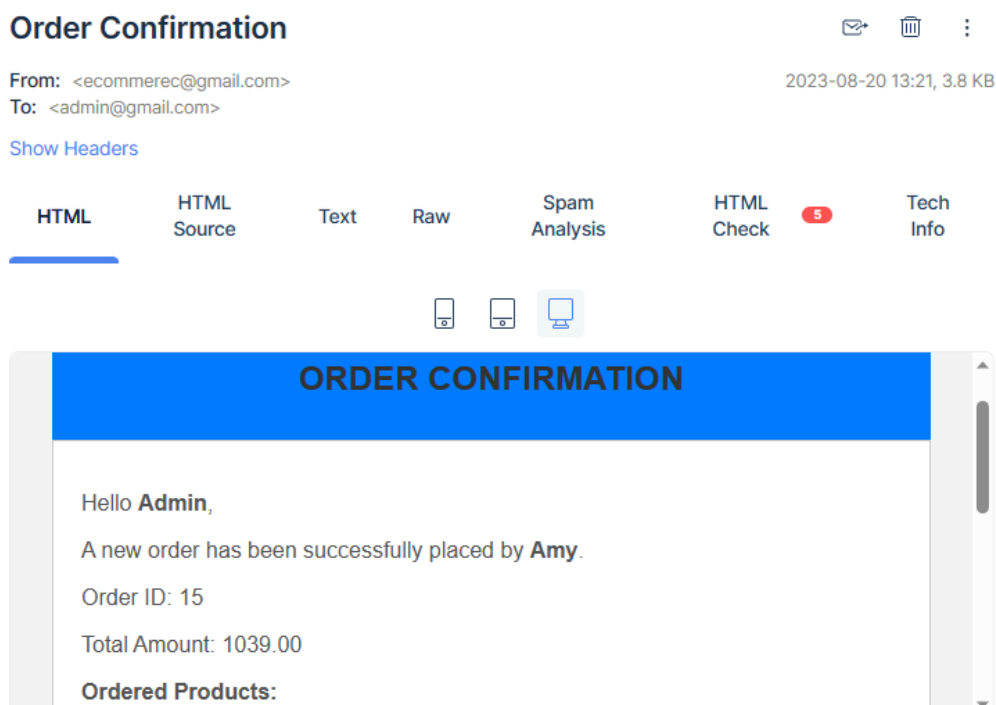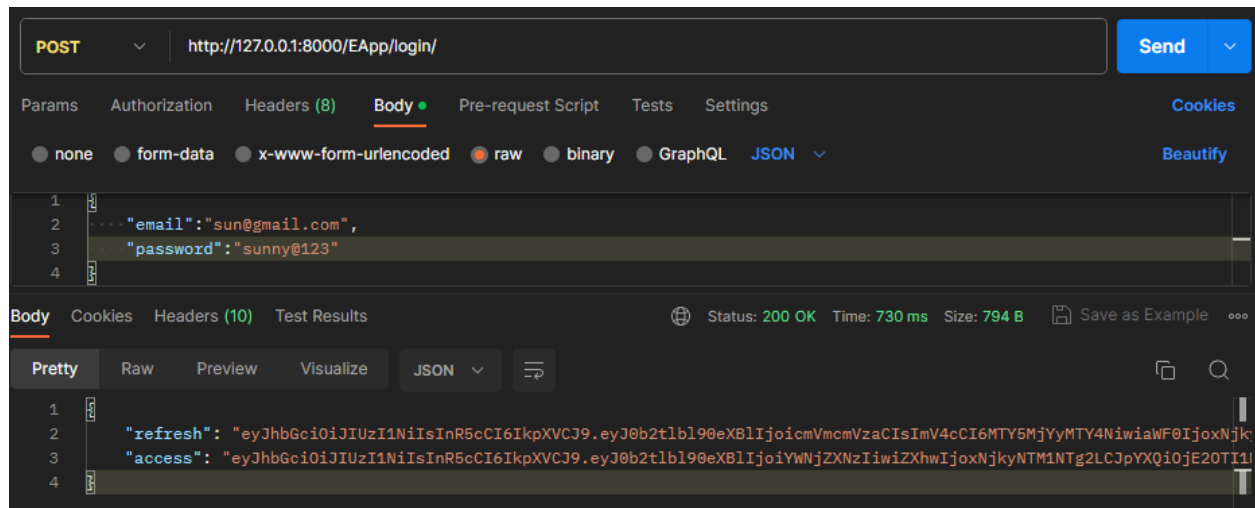
→Order History API

This API handles the listing of the orders made by the user. It shows the order history.

```
GET    http://127.0.0.1:8000/EApp/order/history/                    Send

Params   Authorization   Headers (7)   Body   Pre-request Script   Tests   Settings        Cookies

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

                              This request does not have a body

Body   Cookies   Headers (10)   Test Results      Status: 200 OK   Time: 318 ms   Size: 1 KB   Save as Example

Pretty   Raw   Preview   Visualize   JSON ∨

 8    {
 9        "id": 15,
10        "ordered_items": [
11            {
12                "product_name": "Diary",
13                "ordered_quantity": 2,
14                "ordered_price": "125.00"
15            },
16            {
17                "product_name": "Pride and Prejudice Book",
18                "ordered_quantity": 1,
19                "ordered_price": "789.00"
20            }
21        ],
22        "total_amount": "1039.00",
23        "date_of_order": "2023-08-20T13:21:25.894420Z",
24        "status": "Processing",
25        "Address": "House No 321,garden street NYC"
```

→User login

Project GITHUB repo URL: https://github.com/minta19/FinalTask.git

Pythonanywhere URL example: https://minta19.pythonanywhere.com/EApp/products/