E-commerce API Documentation

PP Sahad Abdul Karim

August 16, 2023

Contents

1 Introduction

The E-commerce API project involves building a comprehensive Python Django-based API for online shopping. By integrating JWT authentication, order tracking, and email notifications, the project aims to create a well-rounded E-commerce platform catering to both customers and administrators.

2 Admin Functionalities

2.1 Admin Registration

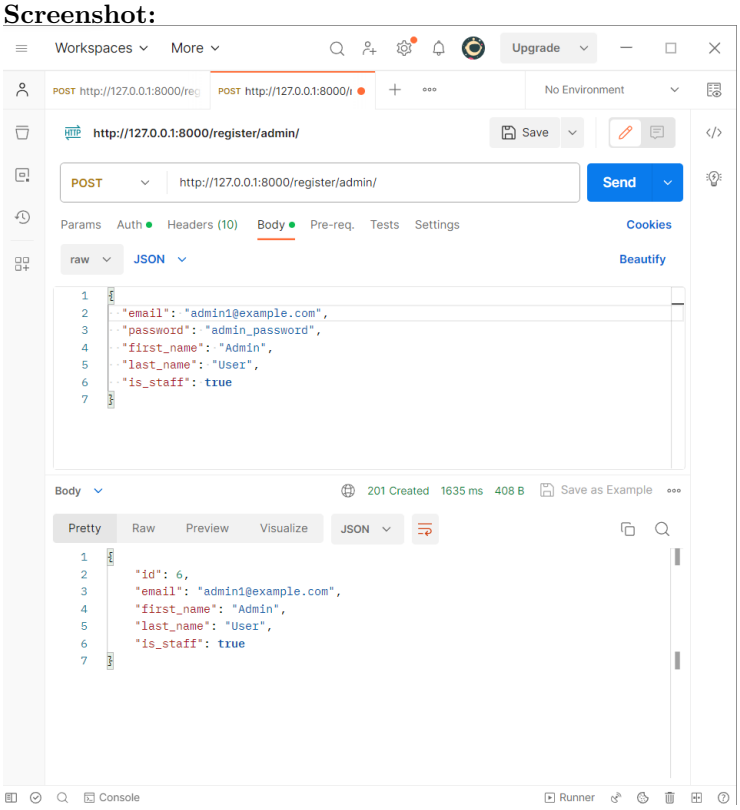
Admins can register for an account with elevated privileges to access the admin dashboard. Admin registration can only be performed by another admin.
Endpoint: `POST /register/admin/`

Functionality: Register admin users with necessary information and permissions.

Example Request:

```
POST http://localhost:8000/register/admin/

{
  "email": "admin@example.com",
  "password": "admin_password",
  "first_name": "Admin",
  "last_name": "User",
  "is_staff": true
}
```

Screenshot:



2.2 Admin Authentication

Explain how admins can log in using JWT authentication Admins can log in using JWT (JSON Web Token) authentication, which provides a secure way to access the admin dashboard.

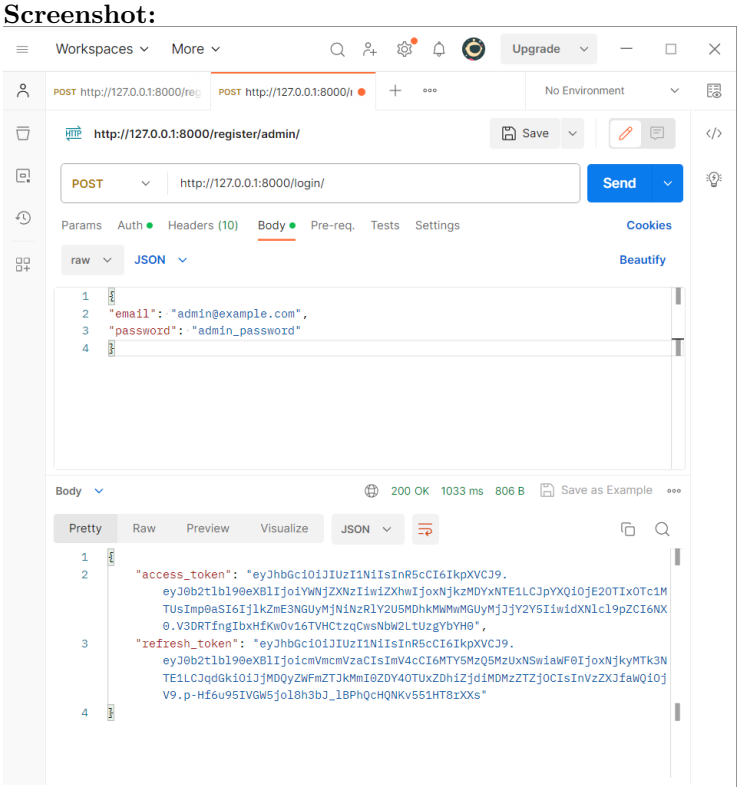
Endpoint: `POST /login/`

Functionality: Authenticate admin users and generate JWT tokens for access.

Example Request:

```
POST http://localhost:8000/login/

{
"email": "admin@example.com",
"password": "admin_password"
}
```

Screenshot:



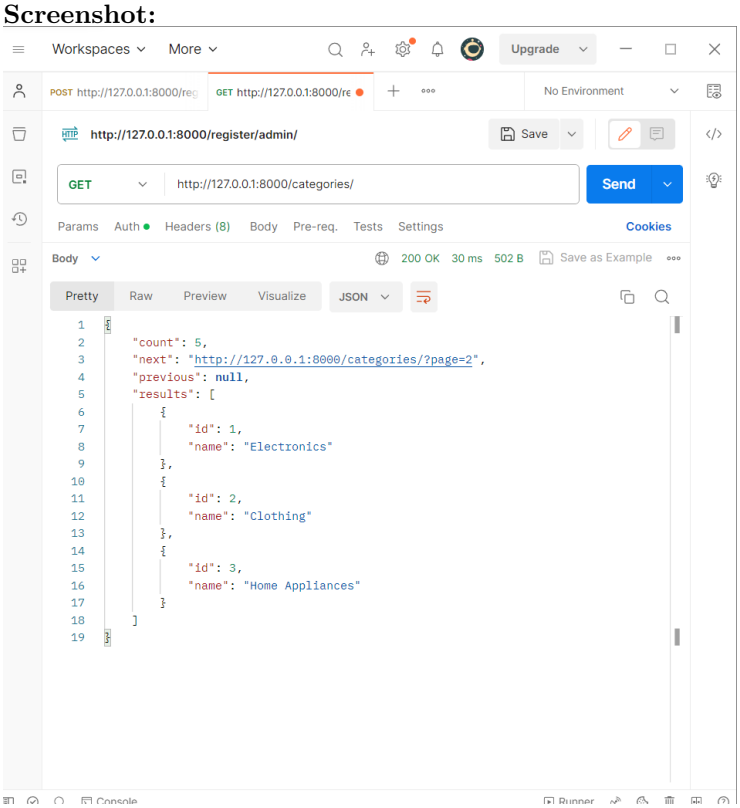
2.3 Admin Category Management

Admins can manage product categories by adding new categories and updating existing ones.

Endpoint: `GET /categories/`
Functionality: Retrieve a list of all product categories.
Example Request:

```
GET http://localhost:8000/categories/
```

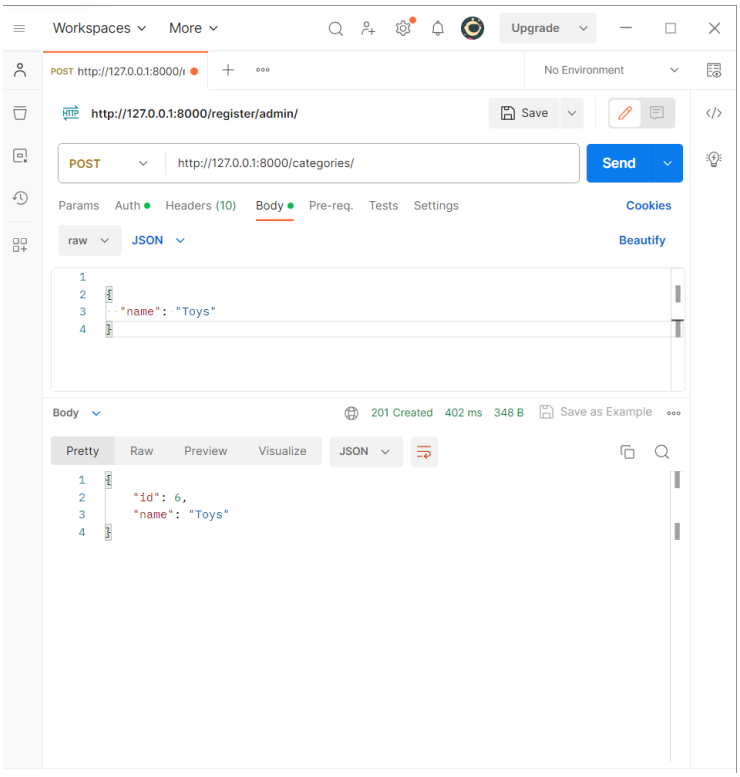
Screenshot:



Endpoint: `POST /categories/`
Functionality: Add a new product category.
Example Request:

```
POST http://localhost:8000/categories/

{
  "name": "Toys"
}
```

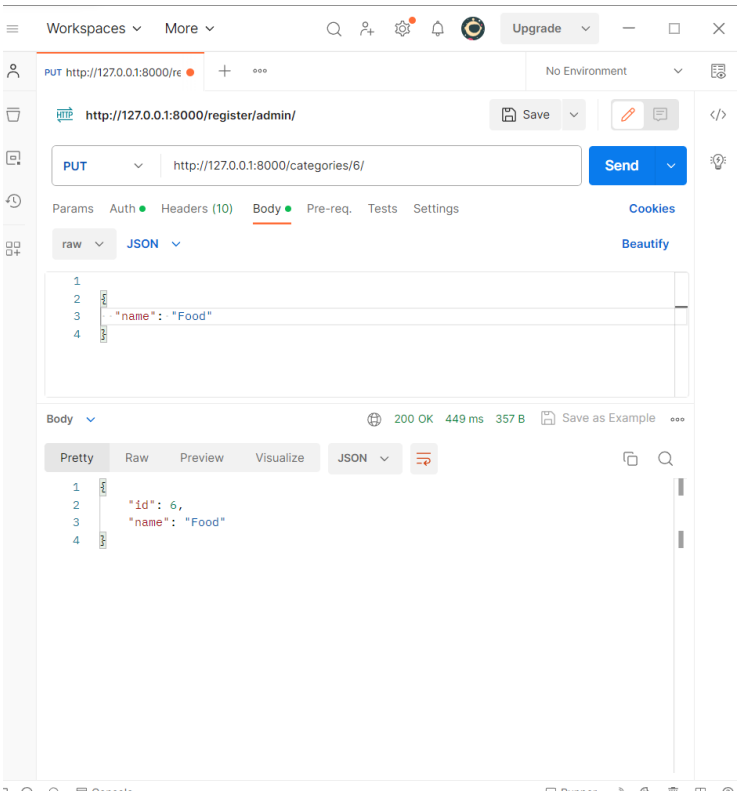
Screenshot:



Endpoint: `PUT /categories/<category_id>/`
Functionality: Update an existing product category.
Example Request:

```
PUT http://localhost:8000/categories/1/

{
  "name": "Updated Electronics"
}
```

Screenshot:



2.4 Admin Product Management

Admins can manage products using CRUD operations and handle product attributes such as name, description, price, and image upload.

Create a New Product

Admins can add new products by sending a POST request to the endpoint below. Provide the necessary attributes, including an image file, in the request body.

Endpoint: `POST /products/`

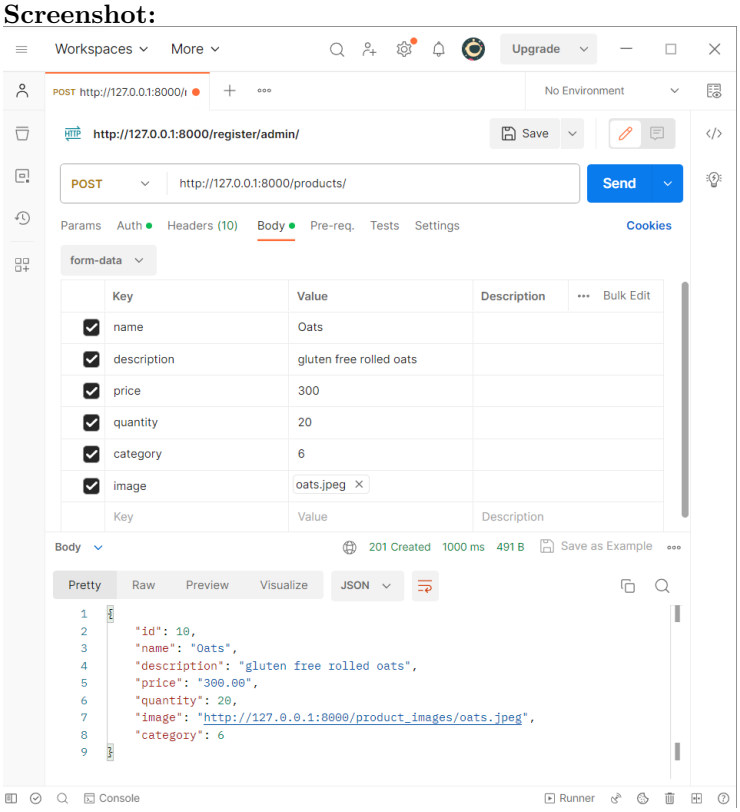
Functionality: Create a new product with attributes.

Example Request:

```
POST http://localhost:8000/products/

{
  "name": "Product Name",
  "description": "Product description.",
  "price": 29.99,
  "image": (image file),
  "category":1,
```

```
  "quantity":20
}
```

Screenshot:



Retrieve Product Details

Admins can retrieve details of a specific product by sending a GET request to the endpoint below. Replace {`product_id`} with the desired product's ID.

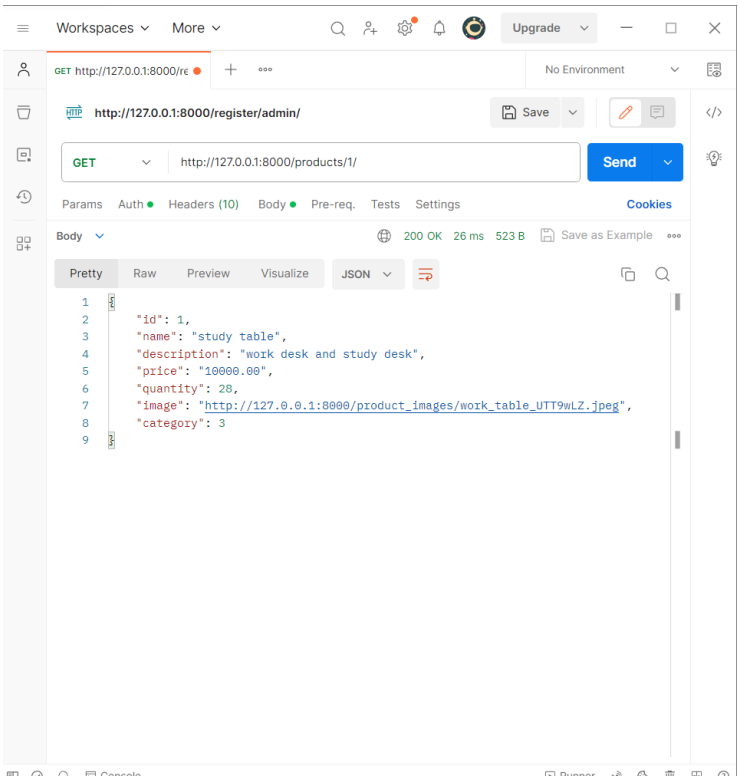
Endpoint: `GET /products/{product_id}/`

Functionality: Get details of a product.

Example Request:

```
GET http://localhost:8000/products/1/
```

Screenshot:



Update Product Details

Admins can update product details by sending a PUT or PATCH request to the endpoint below. Replace {`product_id`} with the ID of the product to update. Provide attributes to modify in the request body.

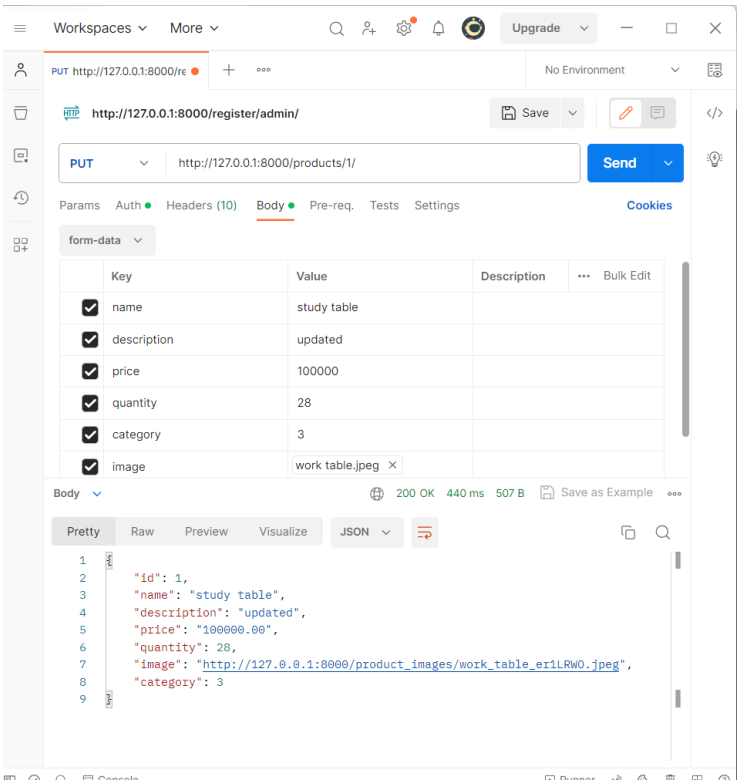
Endpoint: `PUT /products/{product_id}/` or `PATCH /products/{product_id}/`

Functionality: Update attributes of a product.

Example Request:

```
PUT http://localhost:8000/products/1/

{
  "name": "Updated Product Name",
  "description": "Updated description.",
  "price": 39.99
}
```

Screenshot:



Delete a Product

Admins can delete a product by sending a DELETE request to the endpoint below. Replace {`product_id`} with the ID of the product to delete.

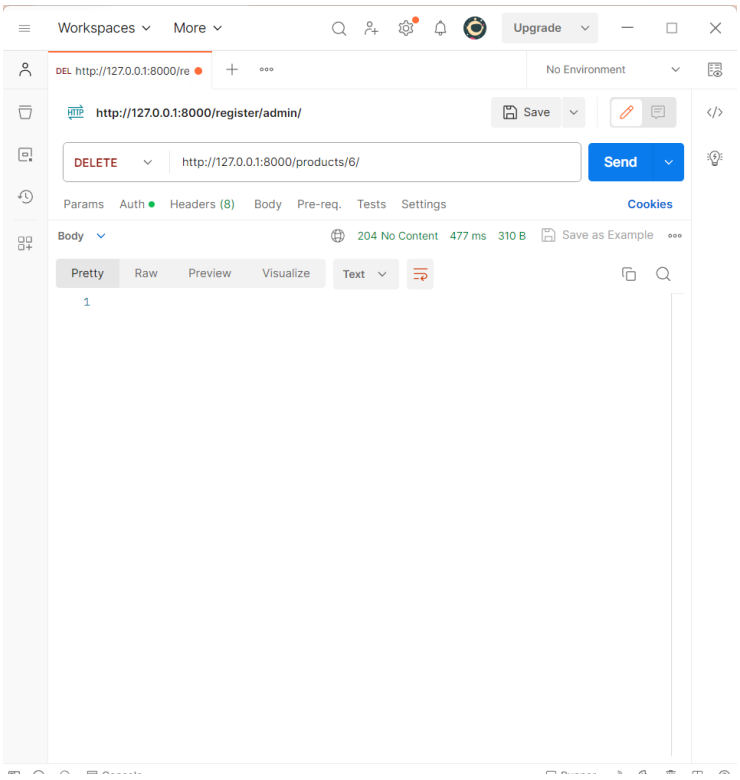
Endpoint: `DELETE /products/{product_id}/`

Functionality: Delete a product.

Example Request:

```
DELETE http://localhost:8000/products/6/
```

Screenshot:



2.5 Order Management

Admins have the capability to manage orders, view order details, and update the status of orders.

View All Orders

Admins can retrieve a list of all orders placed on the platform by sending a GET request to the following endpoint.

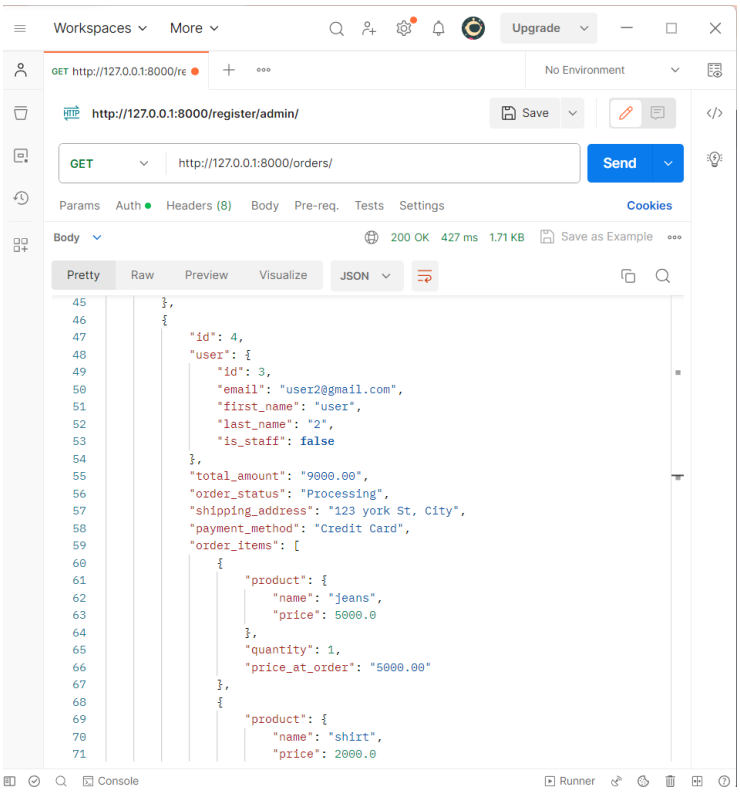
Endpoint: `GET /orders/`

Functionality: Get a list of all orders.

Example Request:

```
GET http://localhost:8000/orders/
```

Screenshot:



Update Order Status

Admins have the authority to update the status of an order by sending a PATCH request to the following endpoint. Replace {`order_id`} with the ID of the order to be updated. Include the `order_status` field in the request body to indicate the new status.

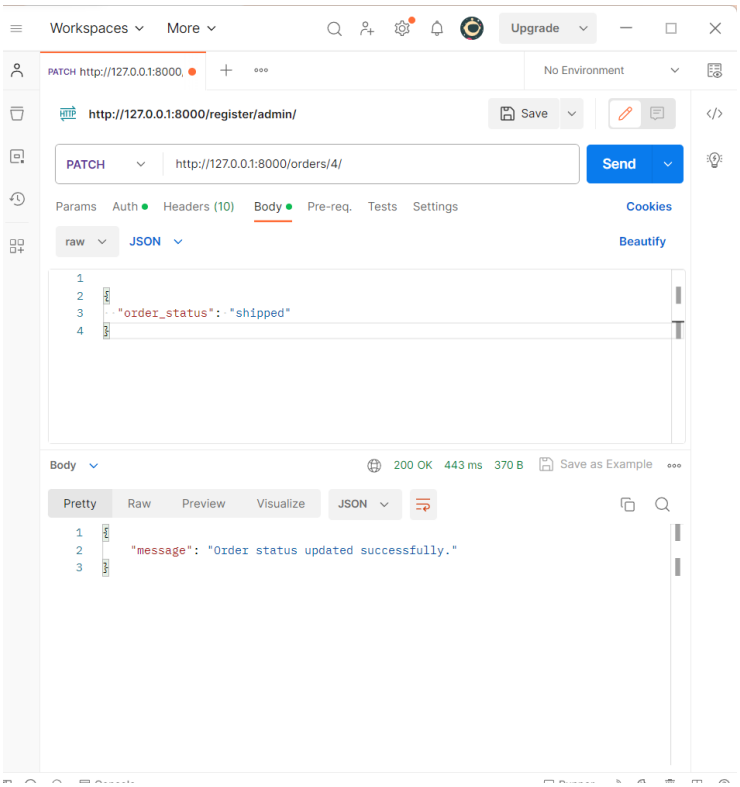
Endpoint: `PATCH /orders/{order_id}/`

Functionality: Update the status of an order.

Example Request:

```
PATCH http://localhost:8000/orders/1/

{
  "order_status": "shipped"
}
```

Screenshot:



Retrieve Order Details

Admins can retrieve detailed information about a specific order by sending a GET request to the following endpoint. Replace {`order_id`} with the ID of the desired order.

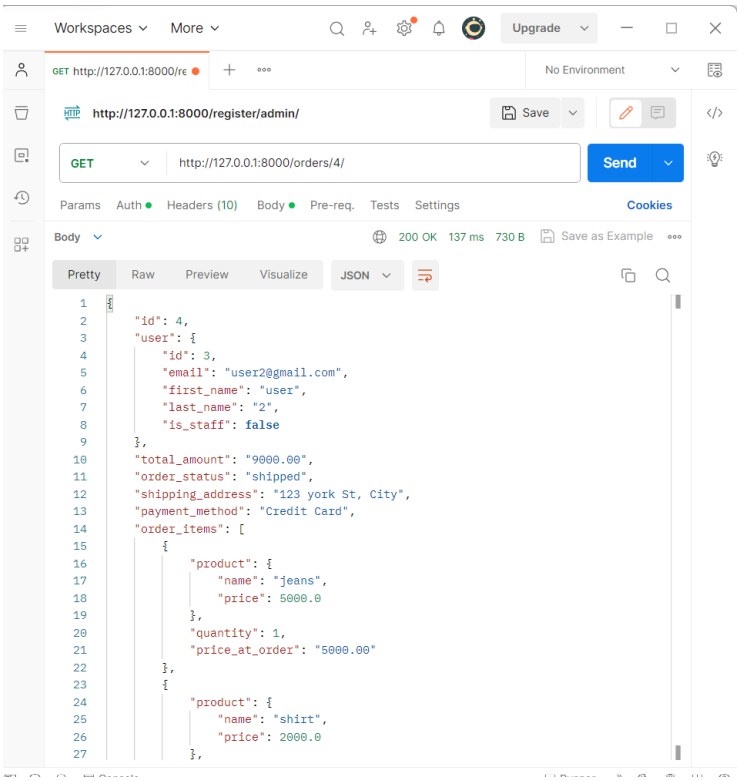
Endpoint: `GET /orders/{order_id}/`

Functionality: Get details of a specific order.

Example Request:

```
GET http://localhost:8000/orders/1/
```

Screenshot:



2.6 User Management

Admins possess the ability to manage registered users and their accounts, including viewing user details and making changes to user accounts.

View User List

Admins can retrieve a list of registered users (excluding other administrators) by sending a GET request to the following endpoint.

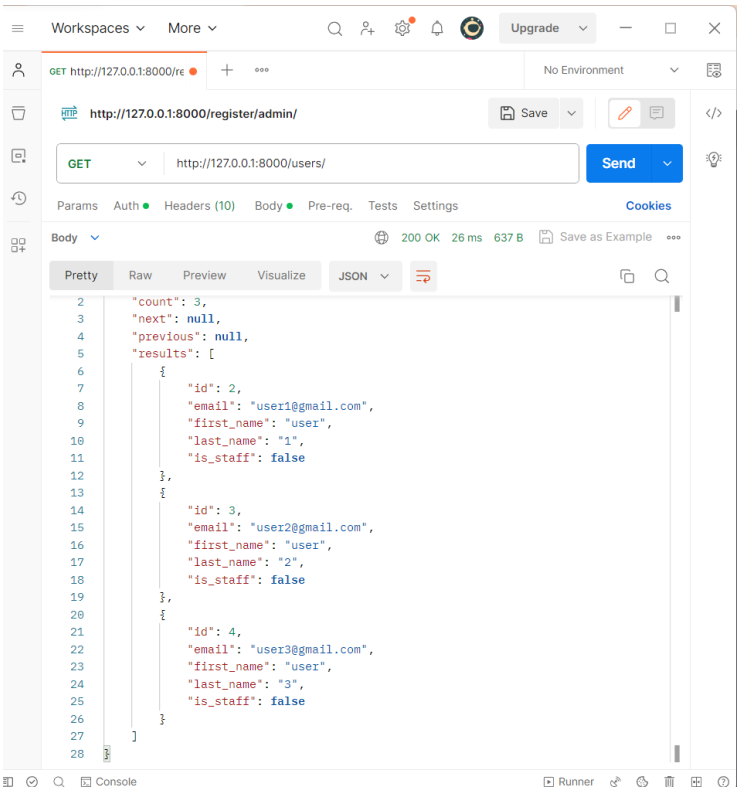
Endpoint: `GET /users/`

Functionality: Get a list of registered users.

Example Request:

```
GET http://localhost:8000/users/
```

Screenshot:



View User Details

Admins can retrieve detailed information about a specific user by sending a GET request to the following endpoint. Replace {`user_id`} with the ID of the desired user.

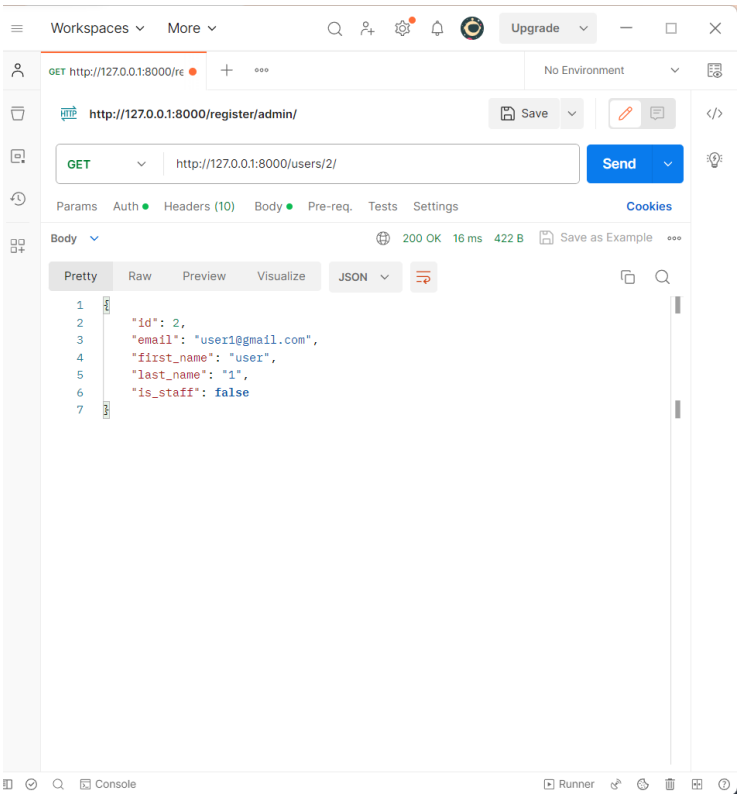
Endpoint: `GET /users/{user_id}/`

Functionality: Get details of a specific user.

Example Request:

```
GET http://localhost:8000/users/2/
```

Screenshot:



Deactivate or Delete User Account

Admins can deactivate or delete a user's account by sending a DELETE request to the following endpoint. Replace {`user_id`} with the ID of the user to be deactivated or deleted.

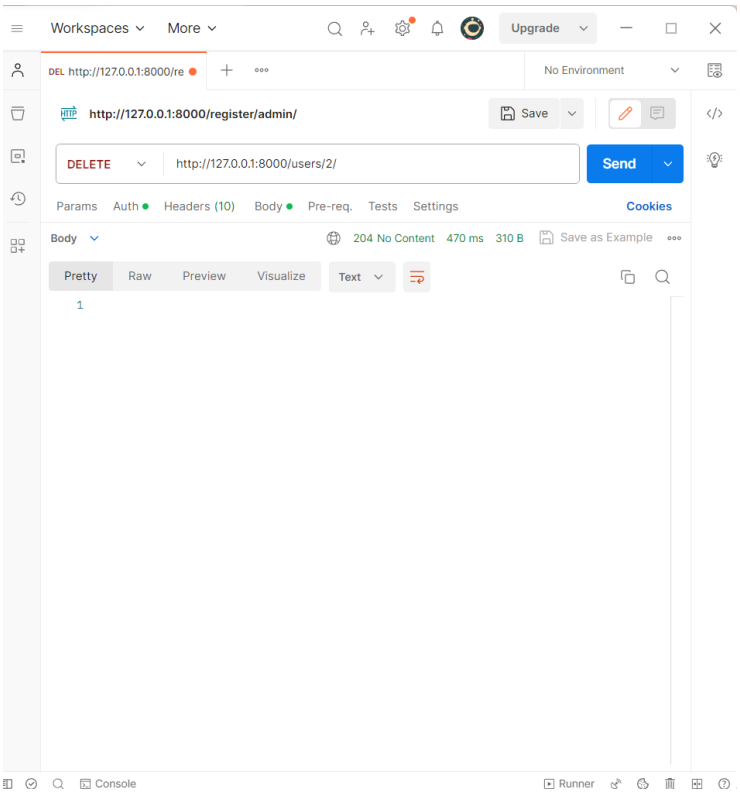
Endpoint: `DELETE /users/{user_id}/`

Functionality: Deactivate or delete a user's account.

Example Request:

```
DELETE http://localhost:8000/users/3/
```

Screenshot:



2.7 Email Notifications

Admins have the capability to send custom email notifications to users for various purposes, such as order updates and promotional emails.

Send Custom Email

Admins can send a custom email notification to users by sending a POST request to the following endpoint.

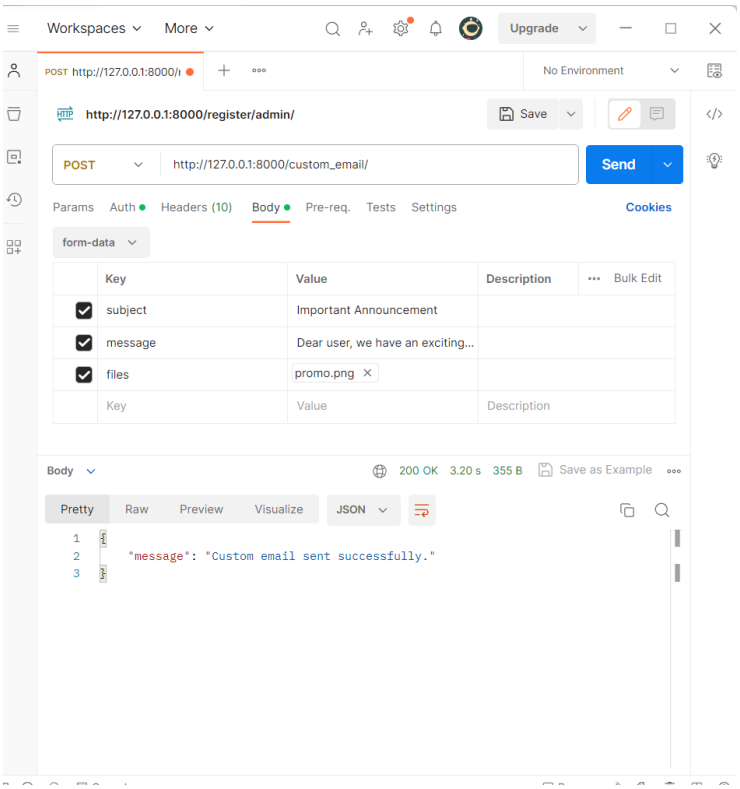
Endpoint: `POST /custom_email/`

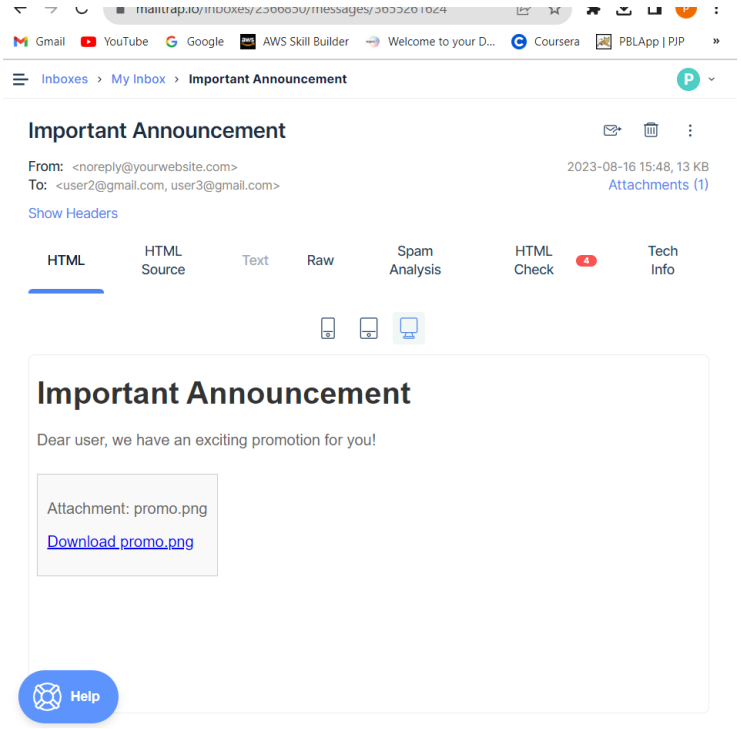
Functionality: Send a custom email notification to users.

Example Request:

```
POST http://localhost:8000/custom_email/

{
  "subject": "Important Announcement",
  "message": "Dear user, we have an exciting promotion for you!",
  "files": attachment.png
}
```

Screenshot:


Workspaces
More
Upgrade
POST http://127.0.0.1:8000/
No Environment
http://127.0.0.1:8000/register/admin/
Save
POST
http://127.0.0.1:8000/custom_email/
Send
Params
Auth
Headers (10)
Body
Pre-req.
Tests
Settings
Cookies
form-data
Key
Value
Description
Bulk Edit
subject
Important Announcement
message
Dear user, we have an exciting...
files
promo.png
Key
Value
Description
Body
200 OK
3.20 s
355 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
1
2
3
"message": "Custom email sent successfully."



3 User Functionalities

3.1 User Registration and Authentication

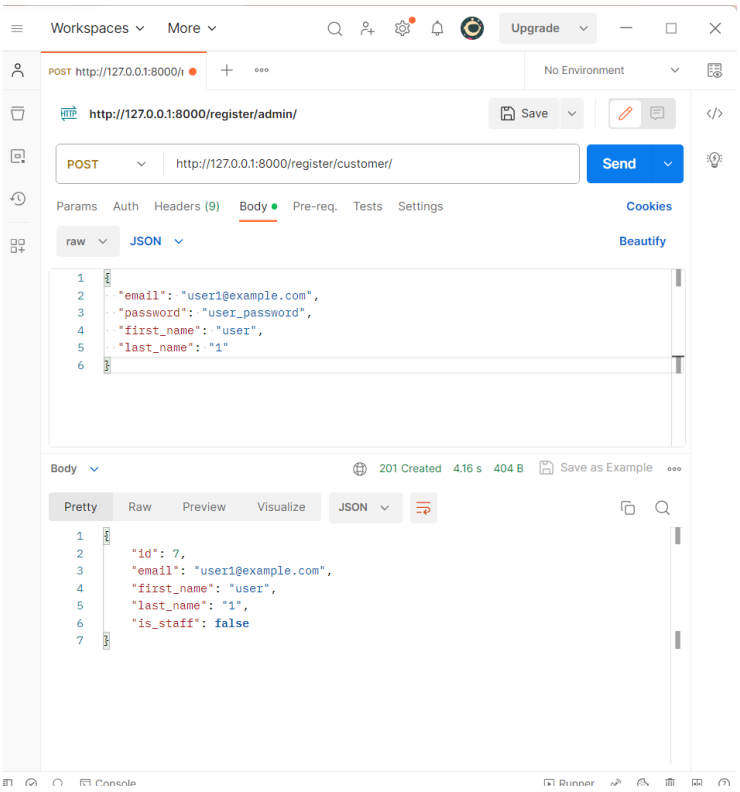
Users can perform registration, login, and reset passwords using the provided endpoints and functionalities.

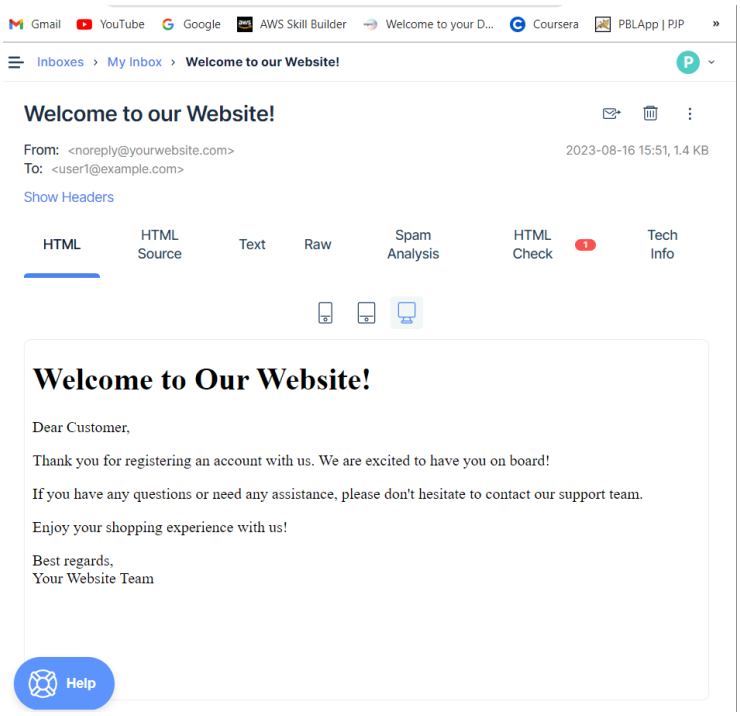
User Registration

Endpoint: `POST /register/customer/`
Functionality: Allow users to register for an account and email is sent.
Example Request:

```
POST http://localhost:8000/register/customer/

{
  "email": "user1@example.com",
  "password": "user_password",
  "first_name": "user",
  "last_name": "1"
}
```

Screenshot:


Workspaces
More
Upgrade
POST http://127.0.0.1:8000/
No Environment
http://127.0.0.1:8000/register/admin/
Save
POST
http://127.0.0.1:8000/register/customer/
Send
Params
Auth
Headers (9)
Body
Pre-req.
Tests
Settings
Cookies
raw
JSON
Beautify
{
"email": "user1@example.com",
"password": "user_password",
"first_name": "user",
"last_name": "1"
}
Body
201 Created
4.16 s
404 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 7,
"email": "user1@example.com",
"first_name": "user",
"last_name": "1",
"is_staff": false
}
Console
Runner

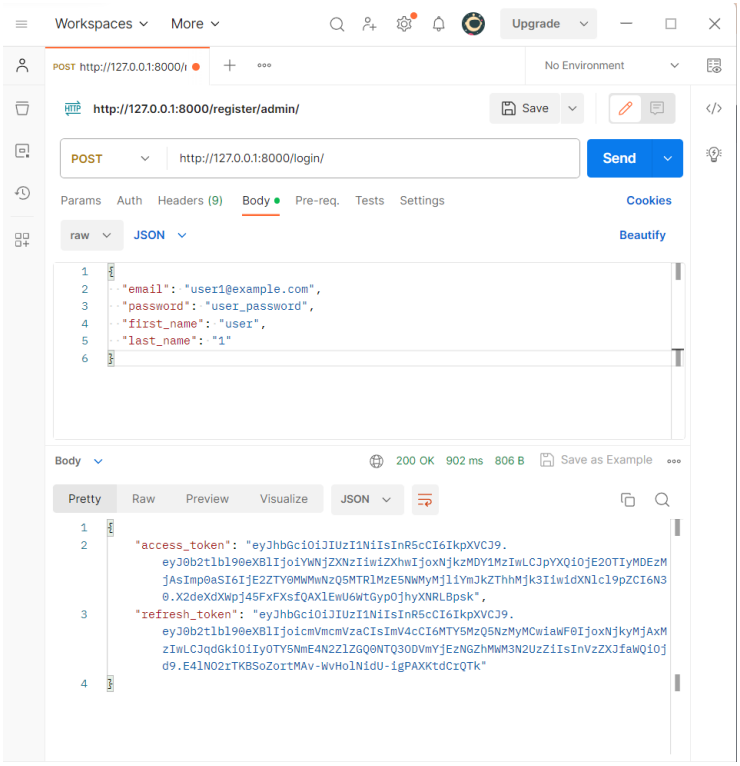


User Login

Endpoint: `POST /login/`
Functionality: Authenticate users and generate JWT tokens for access.
Example Request:

```
POST http://localhost:8000/login/

{
  "email": "user1@example.com",
  "password": "user_password"
}
```

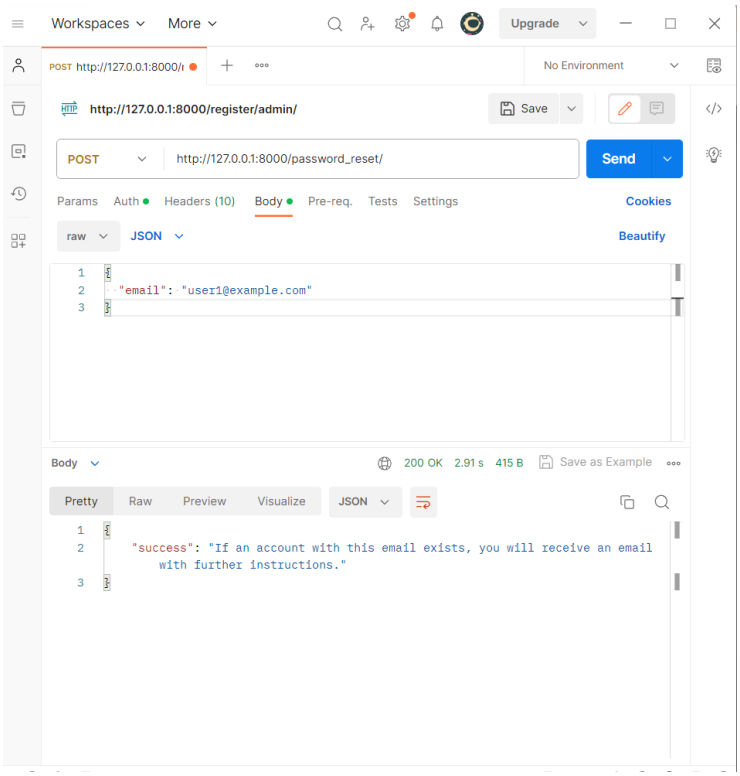
Screenshot:



Password Reset Request
Endpoint: `POST /password_reset/`
Functionality: Allow users to request a password reset email.
Example Request:

```
POST http://localhost:8000/password_reset/

{
  "email": "user1@example.com"
}
```

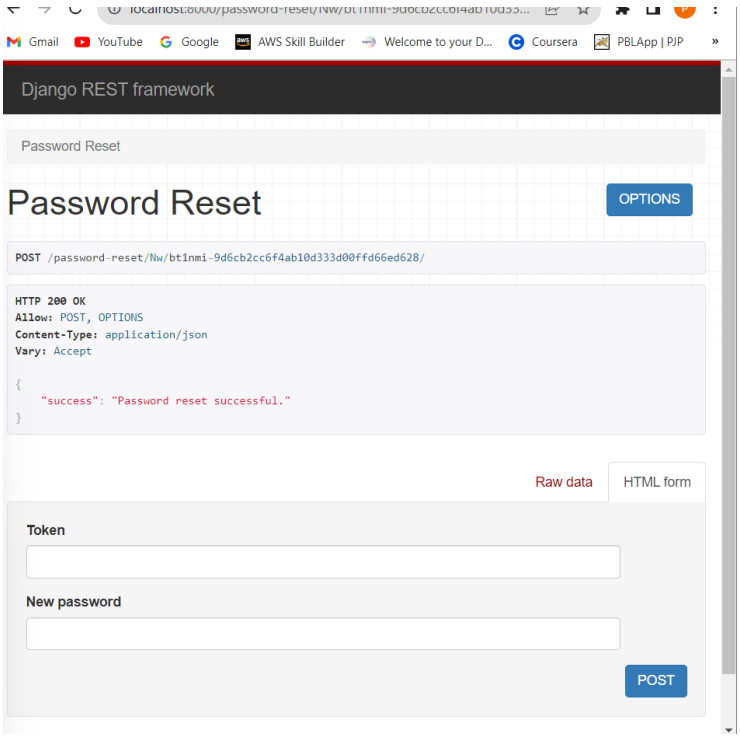
Screenshot:



Password Reset
Endpoint: `POST /password_reset/{uidb64}/{token}/`
Functionality: Reset the password for users using a valid token.
Example Request:

```
POST http://localhost:8000/password_reset/uidb64/token/

{
  "new_password": "new_user_password"
}
```

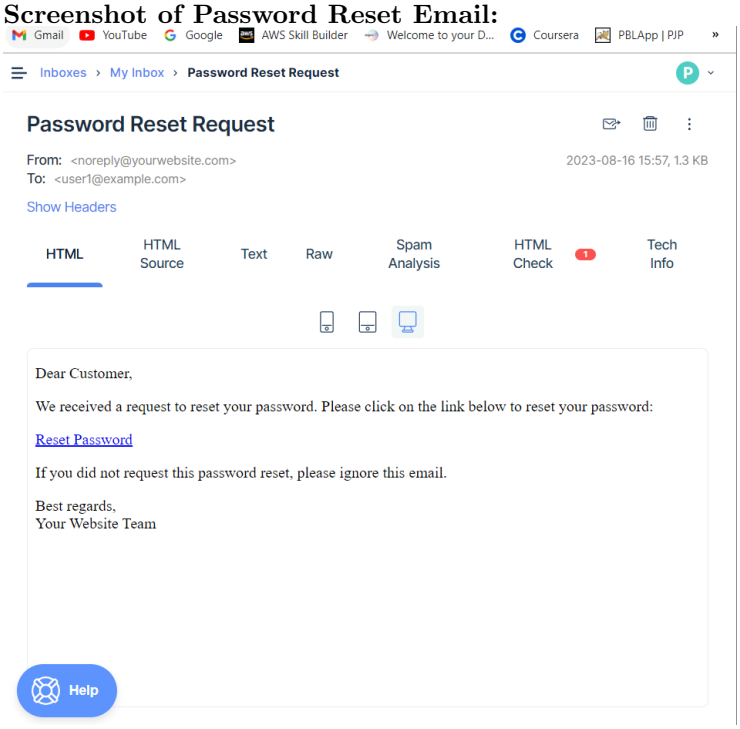
Screenshot:



3.1.1 Password Reset

To reset the password, users will receive an email containing a link with a token. This token is generated based on the user's ID and is used to verify the user's identity. Upon clicking the link, users are directed to a page where they can input their new password.

Screenshot of Password Reset Email:



3.2 Product Listing and Filtering

Users can view and filter the list of available products on the platform using the provided endpoints and filtering options.

Product Listing

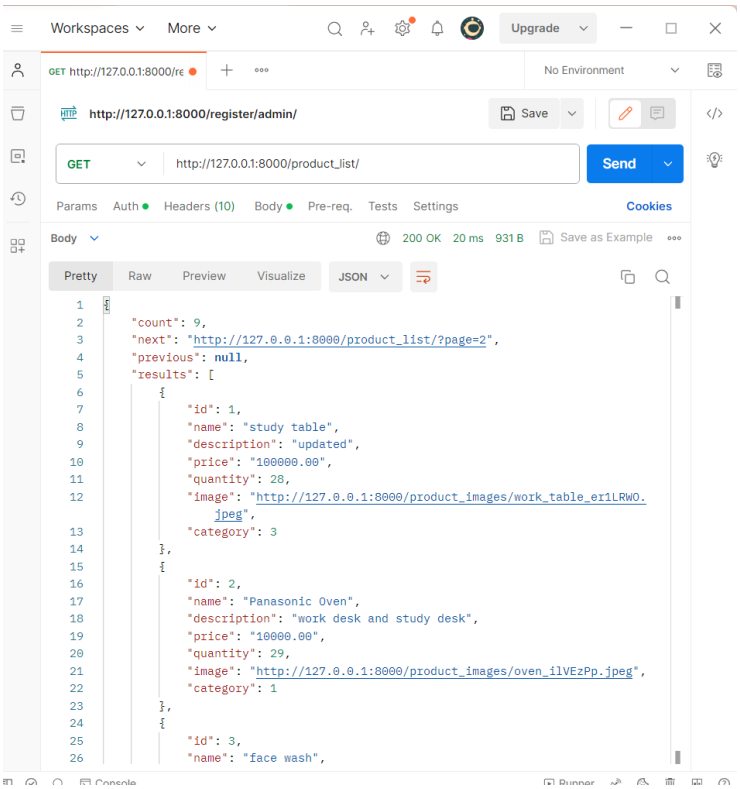
Endpoint: `GET /product_list/`

Functionality: Allow users to retrieve a paginated list of all available products.

Example Request:

```
GET http://localhost:8000/product_list/
```

Screenshot:



Product Filtering

Users can apply filters to the product list based on different attributes such as category, price ranges, and product name.

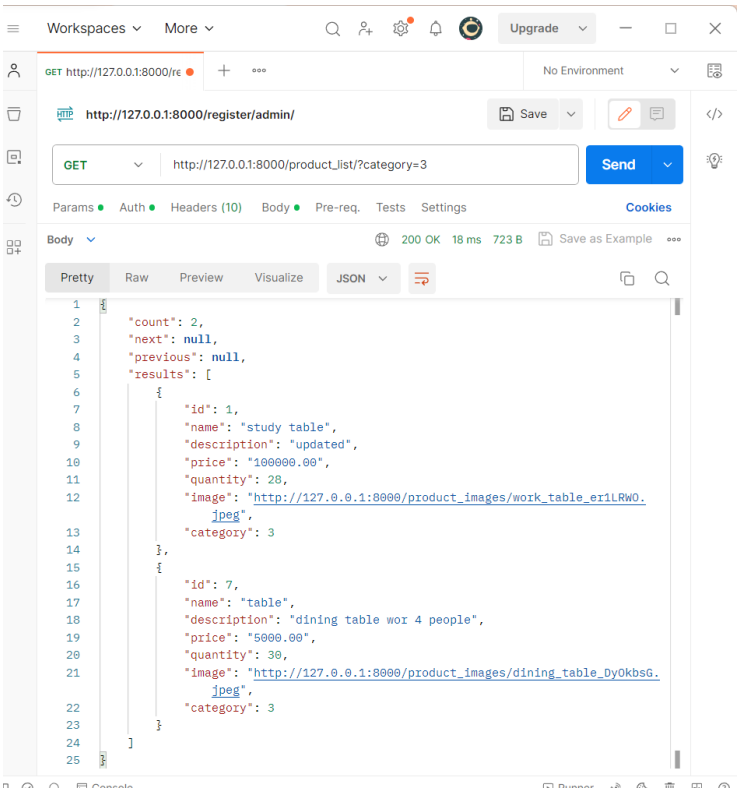
Filtering Options:

1. Filter by Category:

Endpoint: `GET /product_list/?category=<category_id>`

Retrieve a list of products belonging to a specific category.

Screenshot:

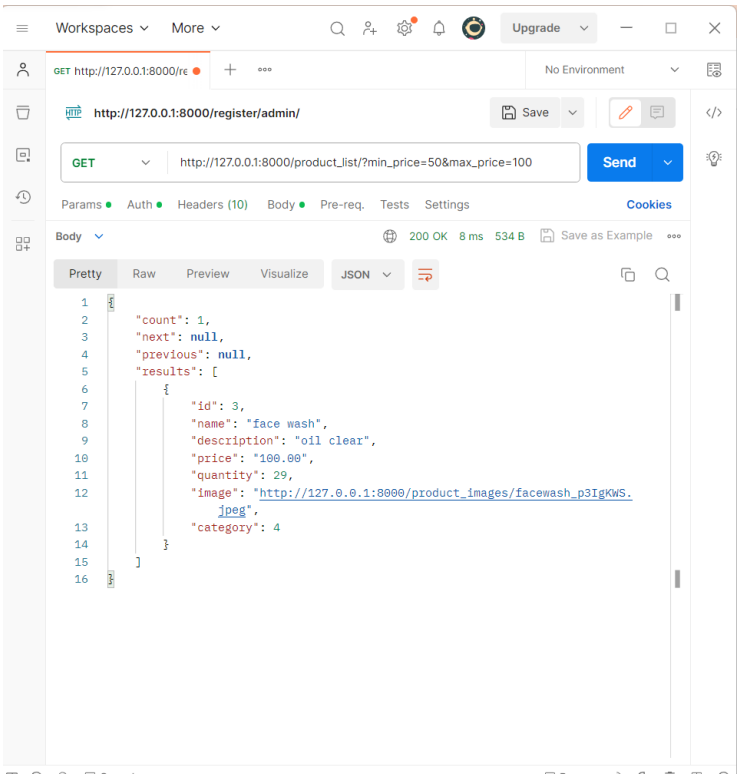


2. Filter by Price Range:

Endpoint: `GET /product_list/?min_price=<min_price>&max_price=<max_price>`

Retrieve a list of products within a specified price range.

Screenshot:

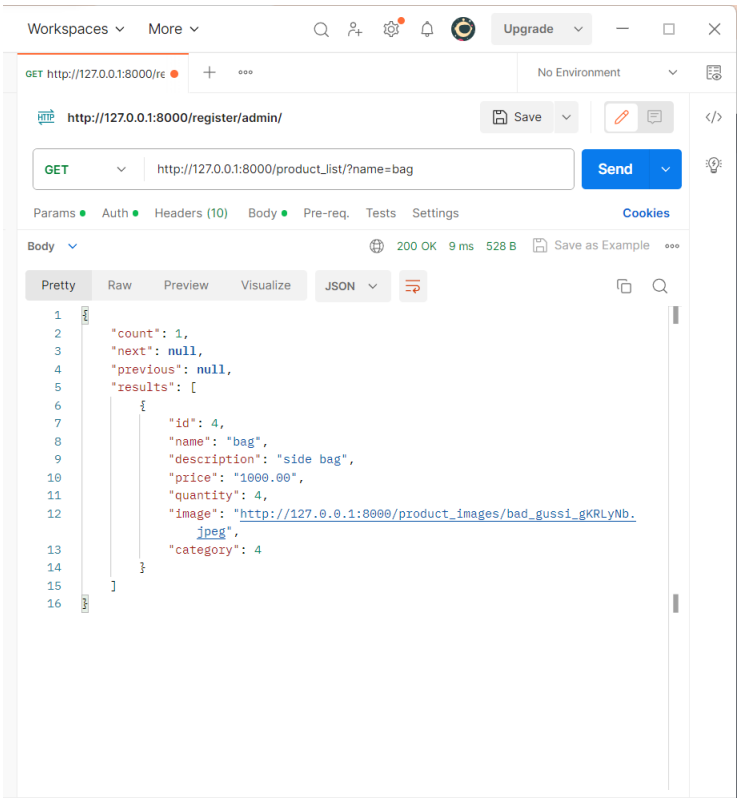


3. Filter by Product Name:

Endpoint: `GET /product_list/?name=<product_name>`

Retrieve list of products containing the specified product name in their names.

Screenshot:



3.3 Product Details

Allow users to view comprehensive information about a specific product, including its description, price, and other relevant details.

Endpoint: `GET /product_detail/<int:pk>/`

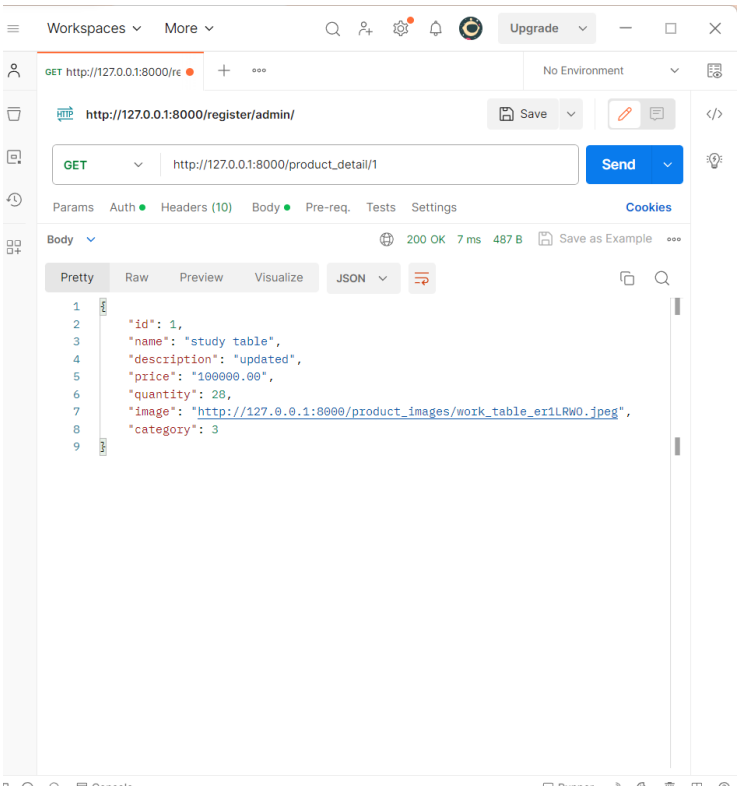
Functionality: Retrieve detailed information about a specific product using its unique ID (`pk`).

Example Request:

```
GET http://localhost:8000/product_detail/1/
```

Example Response:

Screenshot:



This endpoint allows users to access detailed information about a product, helping them make informed purchasing decisions.

3.4 Shopping Cart

Enable users to manage their shopping cart by adding products, viewing the cart's contents, and updating item quantities.

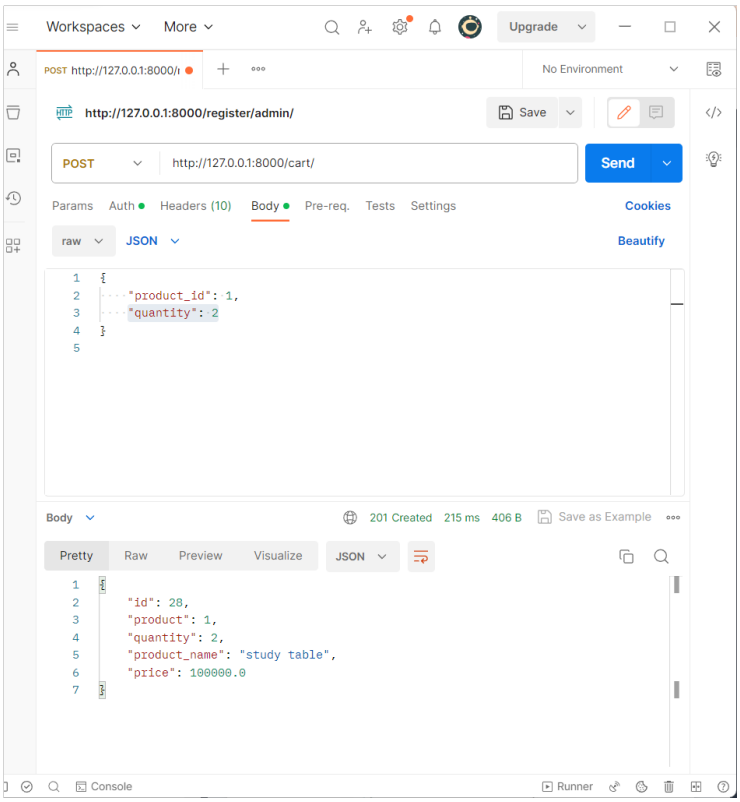
Endpoint: `POST /cart/`

Functionality: Add a product to the user's shopping cart, we can add it multiple times which will increase the quantity in the cart.

Example Request:

```
POST http://localhost:8000/cart/

{
    "product_id": 1,
    "quantity": 2
}
```

Screenshot:



Endpoint: `GET /cart/`
Functionality: Retrieve a list of all items in the user's shopping cart.
Example Request:

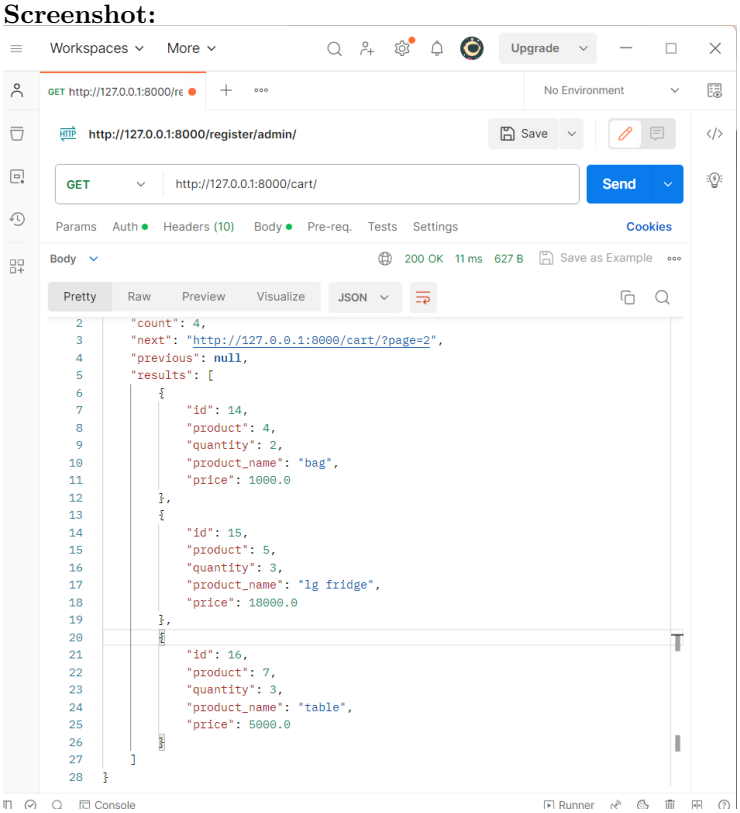
```
GET http://localhost:8000/cart/
```

Example Response:

```
[
        {
           "id": 14,
            "product": 4,
            "quantity": 2,
            "product_name": "bag",
            "price": 1000.0
        },
        {
            "id": 15,
            "product": 5,
            "quantity": 3,
            "product_name": "lg fridge",
            "price": 18000.0
```

```
        },
        {
            "id": 16,
            "product": 7,
            "quantity": 3,
            "product_name": "table",
            "price": 5000.0
        }
]
```

Screenshot:



Endpoint: `GET /cart_items/<int:pk>/`
Functionality: View the details of a specific item in the user's cart.
Example Request:

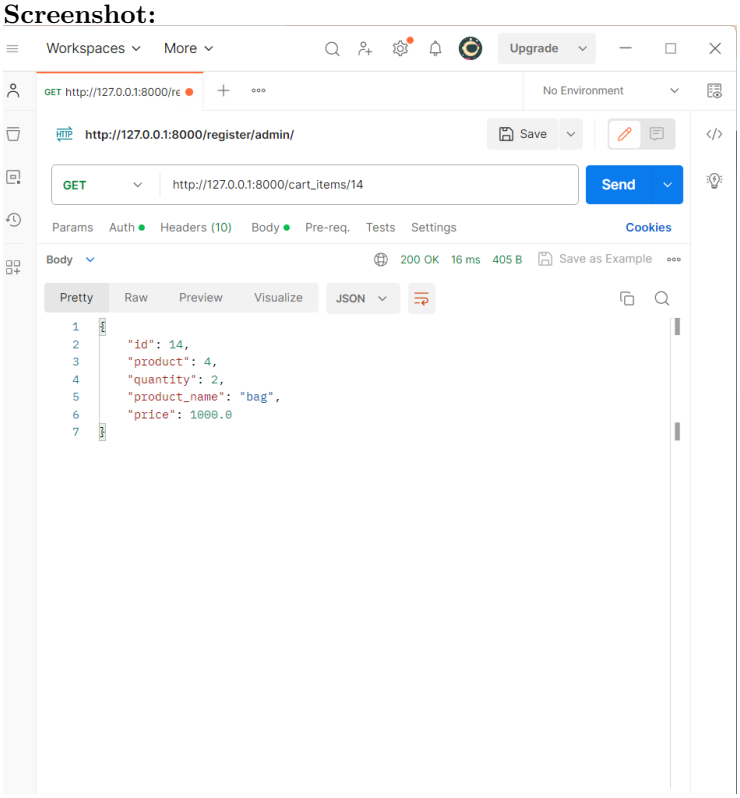
```
GET http://localhost:8000/cart_items/1/
```

Example Response:

```
{
    "id": 10,
    "product": 1,
    "quantity": 2,
```

```
    "product_name": "study table",
    "price": 10000.0
}
```

Screenshot:



3.5 Order Placement

Users can easily place orders for the items in their shopping cart. Upon successful order placement, both users and admins will receive email notifications.

Endpoint: `POST /place_order/`

Functionality: Allows users to place an order for the items in their shopping cart.

Example Request:

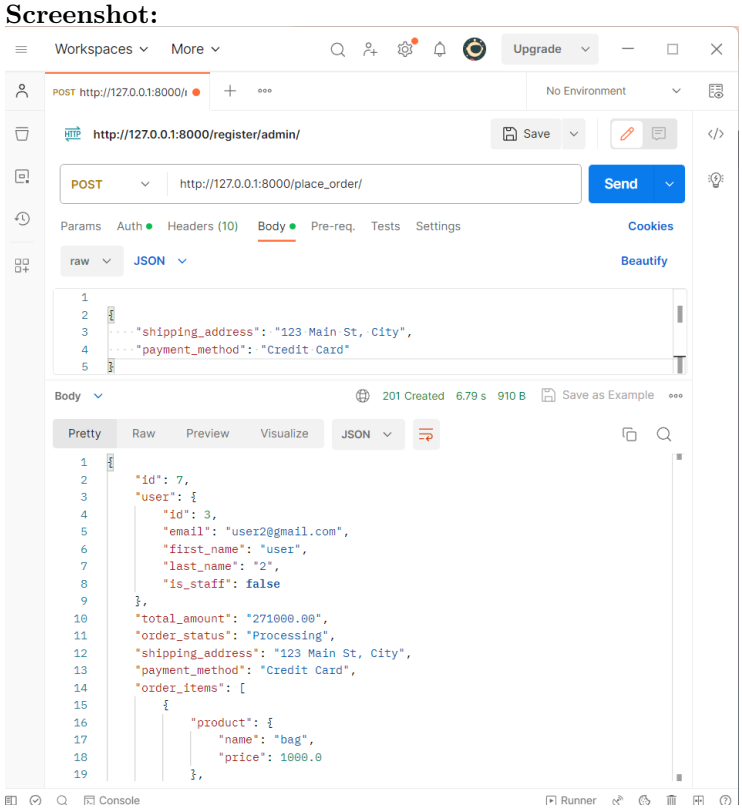
```
POST http://localhost:8000/place_order/

{
    "shipping_address": "123 Main St, City",
    "payment_method": "Credit Card"
}
```

Example Response:

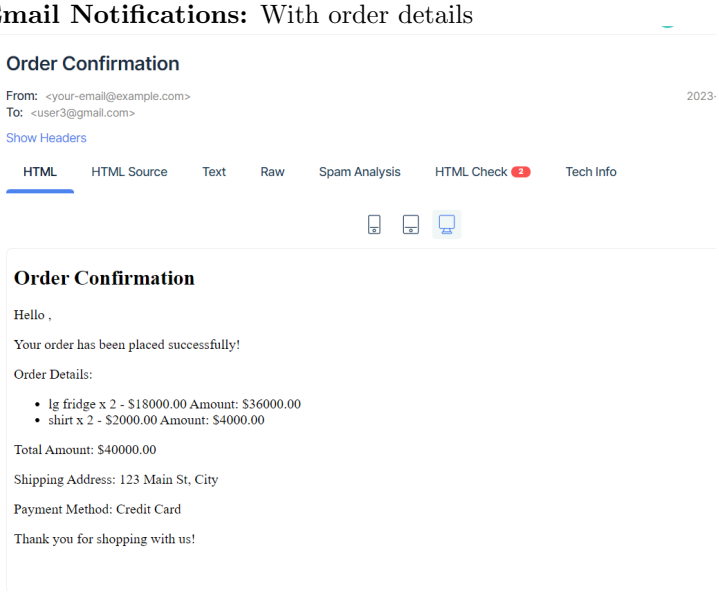
```
{
    "id": 7,
    "user": {
        "id": 3,
        "email": "user2@gmail.com",
        "first_name": "user",
        "last_name": "2",
        "is_staff": false
    },
    "total_amount": "271000.00",
    "order_status": "Processing",
    "shipping_address": "123 Main St, City",
    "payment_method": "Credit Card",
    "order_items": [
        {
            "product": {
                "name": "bag",
                "price": 1000.0
            },
            "quantity": 2,
            "price_at_order": "1000.00"
        },
        {
            "product": {
                "name": "lg fridge",
                "price": 18000.0
            },
            "quantity": 3,
            "price_at_order": "18000.00"
        },
        {
            "product": {
                "name": "table",
                "price": 5000.0
            },
            "quantity": 3,
            "price_at_order": "5000.00"
        },
        {
            "product": {
                "name": "study table",
                "price": 100000.0
            },
            "quantity": 2,
            "price_at_order": "100000.00"
        }
```

```
    ]
}
```

Screenshot:


Workspaces
More
Upgrade
POST http://127.0.0.1:8000/
No Environment
http://127.0.0.1:8000/register/admin/
Save
POST
http://127.0.0.1:8000/place_order/
Send
Params
Auth
Headers (10)
Body
Pre-req.
Tests
Settings
Cookies
raw
JSON
Beautify
{
"shipping_address": "123 Main St, City",
"payment_method": "Credit Card"
}
Body
201 Created
6.79 s
910 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 7,
"user": {
"id": 3,
"email": "user2@gmail.com",
"first_name": "user",
"last_name": "2",
"is_staff": false
},
"total_amount": "271000.00",
"order_status": "Processing",
"shipping_address": "123 Main St, City",
"payment_method": "Credit Card",
"order_items": [
{
"product": {
"name": "bag",
"price": 1000.0
},
Console
Runner

Email Notifications: With order details



3.6 Order History

Users can conveniently view their order history, which includes details of all the past orders they have placed on the platform.

Endpoint: `GET /order_history/`

Functionality: Allows users to retrieve their order history.

Example Request:

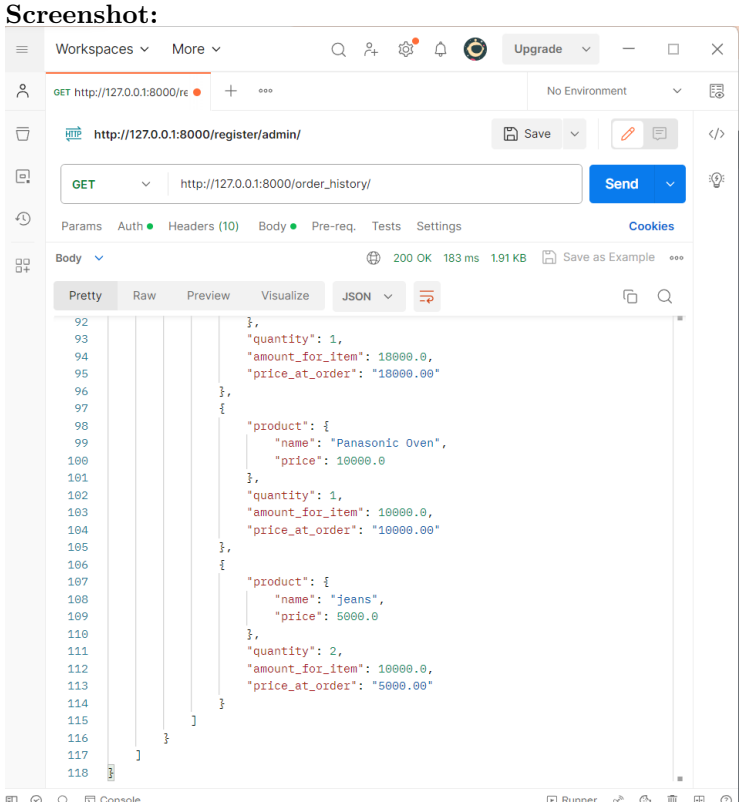
```
GET http://localhost:8000/order_history/
```

Example Response:

```
[
    {
        "id": 8,
            "user": {
                "id": 4,
                "email": "user3@gmail.com",
                "first_name": "user",
                "last_name": "3",
                "is_staff": false
        },
            "total_amount": "200000.00",
            "order_status": "Processing",
            "shipping_address": "123 Main St, City",
            "payment_method": "Credit Card",
        "order_items": [
```

```
            {
               "product": {
                        "name": "lg fridge",
                        "price": 18000.0
                    },
                    "quantity": 1,
                    "amount_for_item": 18000.0,
                    "price_at_order": "18000.00"
                },
                {
                    "product": {
                        "name": "Panasonic Oven",
                        "price": 10000.0
                    },
                    "quantity": 1,
                    "amount_for_item": 10000.0,
                    "price_at_order": "10000.00"
                },
                {
                    "product": {
                        "name": "jeans",
                        "price": 5000.0
                    },
                    "quantity": 2,
                    "amount_for_item": 10000.0,
                    "price_at_order": "5000.00"
            }
        ]
    },
]
```

Screenshot:


Workspaces
More
Upgrade
GET http://127.0.0.1:8000/re
No Environment
http://127.0.0.1:8000/register/admin/
Save
GET
http://127.0.0.1:8000/order_history/
Send
Params
Auth
Headers (10)
Body
Pre-req.
Tests
Settings
Cookies
Body
200 OK
183 ms
1.91 KB
Save as Example
Pretty
Raw
Preview
Visualize
JSON
},
"quantity": 1,
"amount_for_item": 18000.0,
"price_at_order": "18000.00"
},
{
"product": {
"name": "Panasonic Oven",
"price": 10000.0
},
"quantity": 1,
"amount_for_item": 10000.0,
"price_at_order": "10000.00"
},
{
"product": {
"name": "jeans",
"price": 5000.0
},
"quantity": 2,
"amount_for_item": 10000.0,
"price_at_order": "5000.00"
}
]
}
]
}