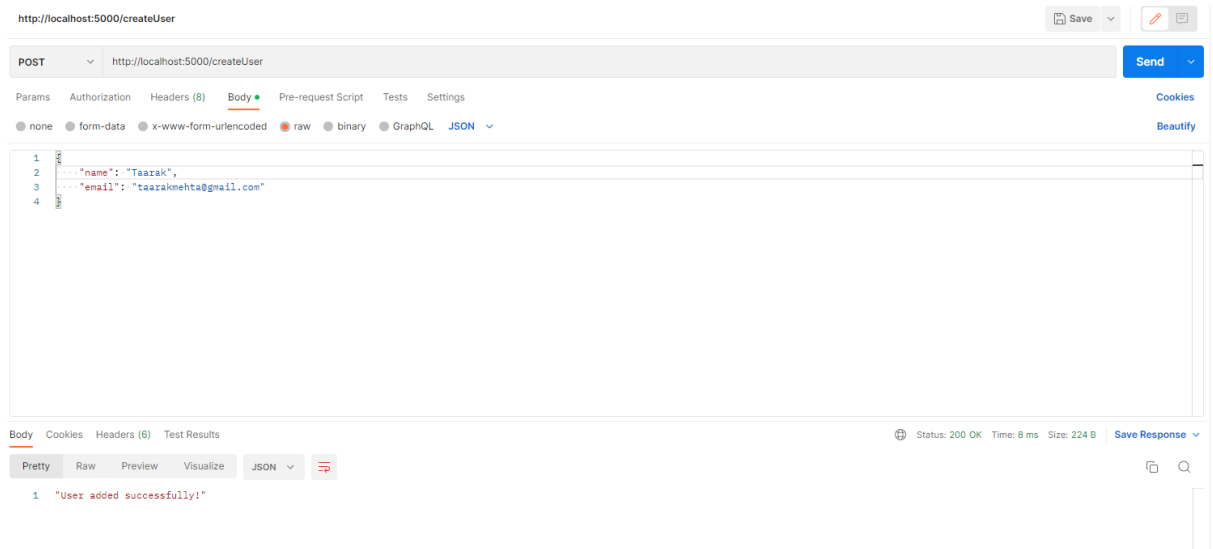


Python Coding Assessment - NavTech

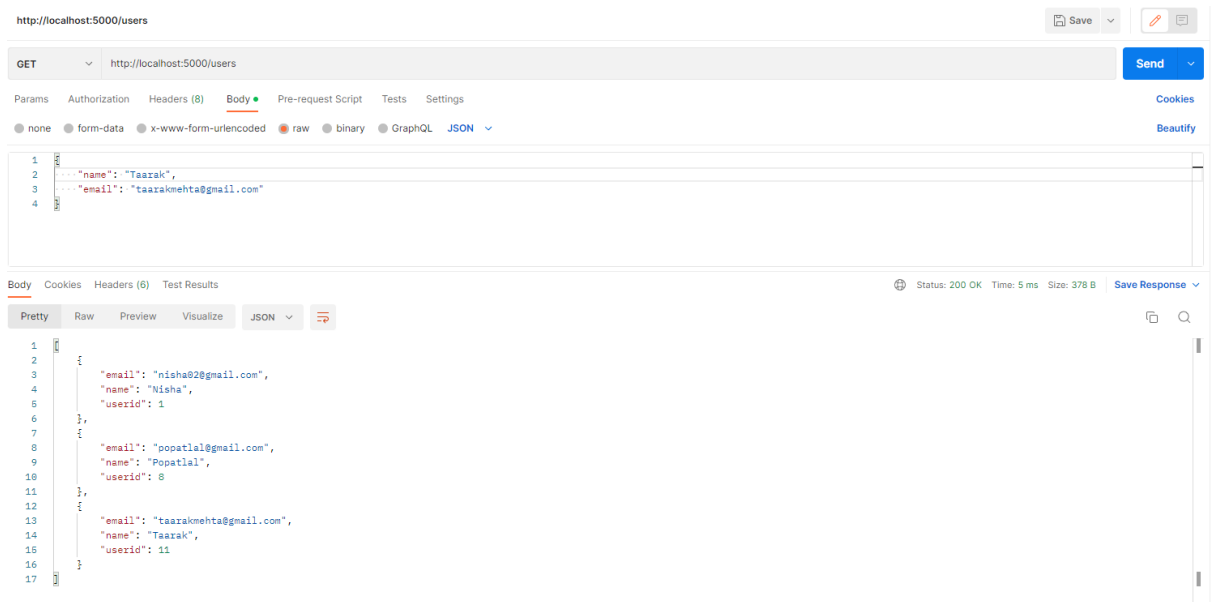
Flask app created, where user can create it's account, view the products available and place orders with multiple products in one order. Admin can handle all the products and view all users and their orders.

Functionalities:

1. Create User: User can create their account.



2. Read Users: Admin can view all registered users.



3. Read 1 User: Admin can view 1 particular user.

http://localhost:5000/user/4

GET http://localhost:5000/user/1

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 8 ms Size: 253 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "email": "nisha2@gmail.com",
3   "name": "Nisha",
4   "userid": 1
5 }
```

4. Update User: User can update their account.

http://localhost:5000/user/4

PUT http://localhost:5000/updateUser

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

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

```
1 {
2   "email": "popatlal2@gmail.com",
3   "name": "Popatlal",
4   "userid": 8
5 }
```

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 8 ms Size: 226 B Save Response

Pretty Raw Preview Visualize JSON

```
1 "User updated successfully!"
```

5. Delete User: User can delete their account or Admin can delete a particular user account.

http://localhost:5000/user/4

DELETE http://localhost:5000/deleteUser/11

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 8 ms Size: 226 B Save Response

Pretty Raw Preview Visualize JSON

```
1 "User deleted successfully!"
```

6. Create Product: Admin can create a product.

http://localhost:5000/user/4

POST http://localhost:5000/createProduct

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

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

```
1 {
2   "price": 897,
3   "product_name": "Monopoly Game"
4 }
```

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 8 ms Size: 227 B Save Response

Pretty Raw Preview Visualize JSON

```
1 "Product added successfully!"
```

7. Read Products: Admin and User both can view all products.

http://localhost:5000/user/4

GET http://localhost:5000/products

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 5 ms Size: 376 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "price": 112,
3   "product_id": 1,
4   "product_name": "Python Book"
5 },
6 {
7   "price": 69,
8   "product_id": 2,
9   "product_name": "Rubik's Cube"
10 },
11 {
12   "price": 897,
13   "product_id": 4,
14   "product_name": "Monopoly Game"
15 }
16 }
```

8. Read 1 Product: Admin and User both can view 1 particular product.

http://localhost:5000/user/4

GET http://localhost:5000/product/4

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 7 ms Size: 257 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "price": 897,
3   "product_id": 4,
4   "product_name": "Monopoly Game"
5 }
```

9. Update Product: Admin can update a particular product.

The screenshot shows a REST client interface for a PUT request to `http://localhost:5000/updateProduct`. The request body is a JSON object: `{ "price": 857, "product_name": "Monopoly Game", "product_id": 4 }`. The response status is 200 OK, and the response body is `"Product updated successfully!"`.

```
PUT http://localhost:5000/updateProduct

{
  "price": 857,
  "product_name": "Monopoly Game",
  "product_id": 4
}
```

Status: 200 OK Time: 7 ms Size: 229 B Save Response

1 "Product updated successfully!"

10. Delete Product: Admin can delete a particular user account.

The screenshot shows a REST client interface for a DELETE request to `http://localhost:5000/deleteProduct/4`. The request body is a JSON object: `{ "price": 857, "product_name": "Monopoly Game", "product_id": 4 }`. The response status is 200 OK, and the response body is `"Product deleted successfully!"`.

```
DELETE http://localhost:5000/deleteProduct/4

{
  "price": 857,
  "product_name": "Monopoly Game",
  "product_id": 4
}
```

Status: 200 OK Time: 7 ms Size: 229 B Save Response

1 "Product deleted successfully!"

11. Create Order: User can create their orders.

The screenshot shows a REST client interface for a POST request to `http://localhost:5000/createOrder`. The request body is a JSON object: `{ "amount": 345, "order_details": [{"Rubik's Cube": 6}], "userid": 8 }`. The response status is 200 OK, and the response body is `"Order placed successfully!"`.

```
POST http://localhost:5000/createOrder

{
  "amount": 345,
  "order_details": [{"Rubik's Cube": 6}],
  "userid": 8
}
```

Status: 200 OK Time: 7 ms Size: 226 B Save Response

1 "Order placed successfully!"

12. Read Orders: Admin can view all orders.

http://localhost:5000/user/4

GET http://localhost:5000/orders

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 7 ms Size: 479 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "amount": 1826,
3   "order_details": [{"Python Book": 3, "Rubik's Cube": 18}],
4   "order_id": 1,
5   "userid": 1
6 },
7 {
8   "amount": 181,
9   "order_details": [{"Python Book": 1, "Rubik's Cube": 1}],
10  "order_id": 2,
11  "userid": 1
12 },
13 {
14   "amount": 345,
15   "order_details": [{"Rubik's Cube": 5}],
16   "order_id": 4,
17   "userid": 8
18 }
19 ]
20 }
```

13. Read 1 Order: Admin can view 1 particular order.

http://localhost:5000/user/4

GET http://localhost:5000/order/4

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 4 ms Size: 276 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "amount": 345,
3   "order_details": [{"Rubik's Cube": 5}],
4   "order_id": 4,
5   "userid": 8
6 }
```

Files Significance:

The following files are present inside the project folder for:

1. app.py: This is the main flask app file which contains all the code with functionalities.
2. Order_system.sql: This is the MySQL database used in the app.