NAME : SURYA D

EMAIL: surya.d.0004@gamil.com

(NOTE : you need to install these before trying this out)
pip install fastapi
pip install "uvicorn[standard]"
pip install fastapi-pagination[fastapi]
Run it in cmd with command
uvicorn file_name:app_name –reload
to view the API docs just go to the
URL/docs

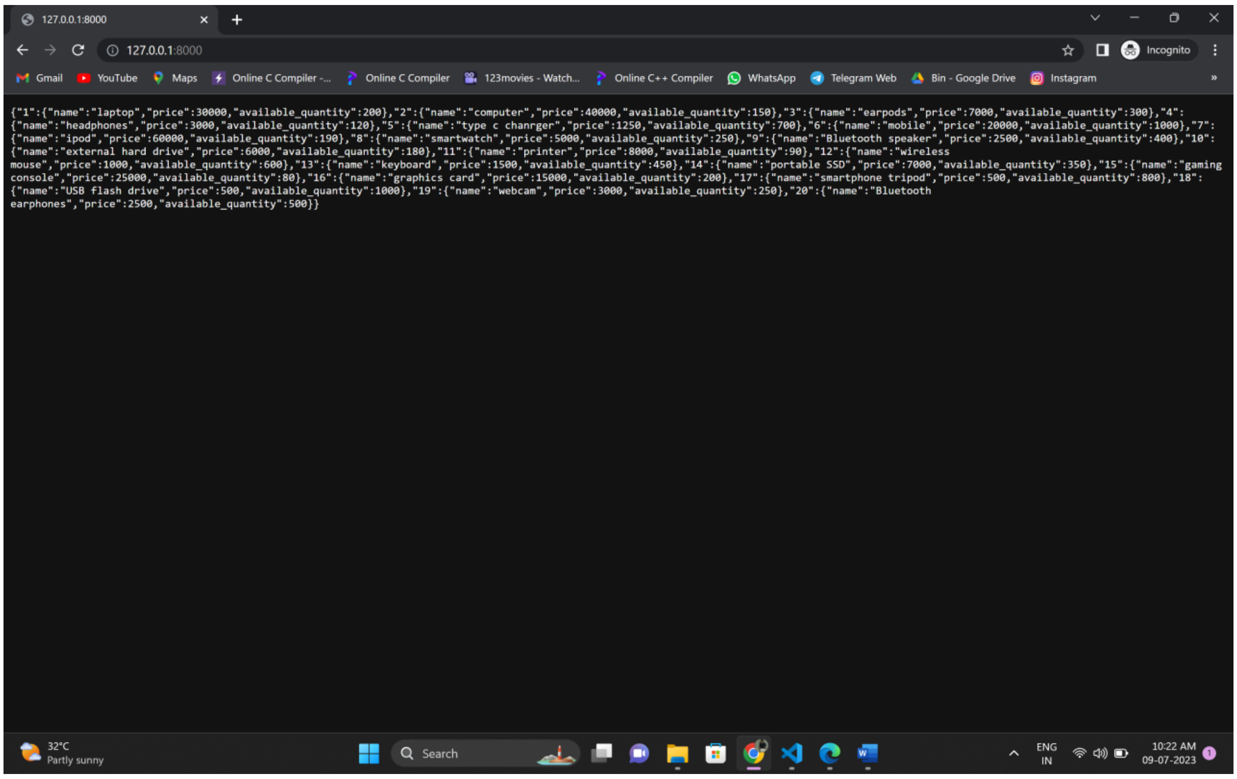
1. API to List all available products in the system. You can create some 10-20 dummy products like TV, laptop, etc for reference. Each product should have these attributes -
    a. Product name
    b. Product price
    c. Product available quantity

CODE:

```
from fastapi import FastAPI

app1 = FastAPI()


product = {
    1:{"name":"laptop","price":30000,"available_quantity":200},
    2:{"name":"computer","price":40000,"available_quantity":150},
    3:{"name":"earpods","price":7000,"available_quantity":300},
    4:{"name":"headphones","price":3000,"available_quantity":120},
    5:{"name":"type c chanrger","price":1250,"available_quantity":700},
    6:{"name":"mobile","price":20000,"available_quantity":1000},
    7:{"name":"ipod","price":60000,"available_quantity":190},
    8:{"name":"smartwatch","price":5000,"available_quantity":250},
    9:{"name":"Bluetooth speaker","price":2500,"available_quantity":400},
    10:{"name":"external hard drive","price":6000,"available_quantity":180},
    11:{"name":"printer","price":8000,"available_quantity":90},
    12:{"name":"wireless mouse","price":1000,"available_quantity":600},
    13:{"name":"keyboard","price":1500,"available_quantity":450},
    14:{"name":"portable SSD","price":7000,"available_quantity":350},
    15:{"name":"gaming console","price":25000,"available_quantity":80},
    16:{"name":"graphics card","price":15000,"available_quantity":200},
    17:{"name":"smartphone tripod","price":500,"available_quantity":800},
    18:{"name":"USB flash drive","price":500,"available_quantity":1000},
    19:{"name":"webcam","price":3000,"available_quantity":250},
    20:{"name":"Bluetooth earphones","price":2500,"available_quantity":500}
}
@app1.get("/")
def send():
    return product
```

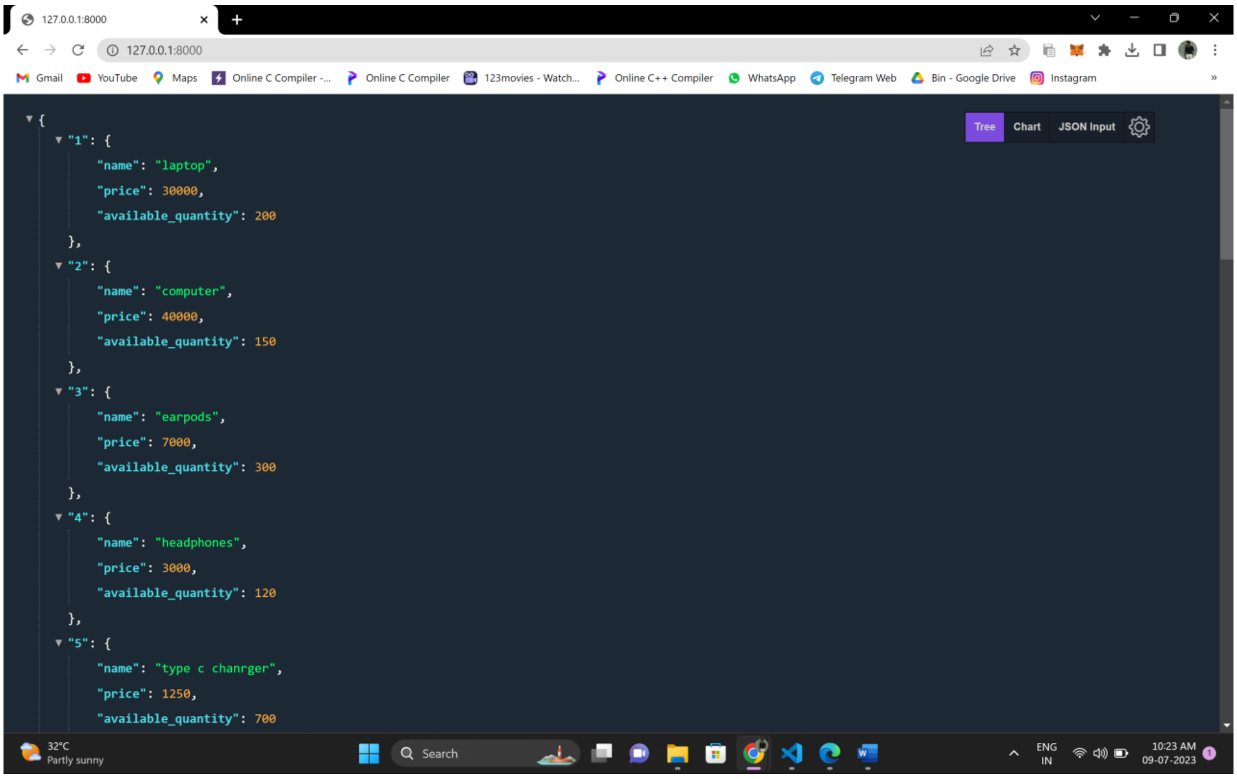
<u>Run it by command :</u>

```
uvicorn  files_name:app1 –reload
```



```
{"1":{"name":"laptop","price":30000,"available_quantity":200},"2":{"name":"computer","price":40000,"available_quantity":150},"3":{"name":"earpods","price":7000,"available_quantity":300},"4":
{"name":"headphones","price":3000,"available_quantity":120},"5":{"name":"type c chanrger","price":1250,"available_quantity":700},"6":{"name":"mobile","price":20000,"available_quantity":1000},"7":
{"name":"ipod","price":60000,"available_quantity":190},"8":{"name":"smartwatch","price":5000,"available_quantity":250},"9":{"name":"Bluetooth speaker","price":2500,"available_quantity":400},"10":
{"name":"external hard drive","price":6000,"available_quantity":180},"11":{"name":"printer","price":8000,"available_quantity":90},"12":{"name":"wireless
mouse","price":1000,"available_quantity":600},"13":{"name":"keyboard","price":1500,"available_quantity":450},"14":{"name":"portable SSD","price":7000,"available_quantity":350},"15":{"name":"gaming
console","price":25000,"available_quantity":80},"16":{"name":"graphics card","price":15000,"available_quantity":200},"17":{"name":"smartphone tripod","price":500,"available_quantity":800},"18":
{"name":"USB flash drive","price":500,"available_quantity":1000},"19":{"name":"webcam","price":3000,"available_quantity":250},"20":{"name":"Bluetooth
earphones","price":2500,"available_quantity":500}}
```

Using beautify JSON viewer



```
▼ {
    ▼ "1": {
        "name": "laptop",
        "price": 30000,
        "available_quantity": 200
    },
    ▼ "2": {
        "name": "computer",
        "price": 40000,
        "available_quantity": 150
    },
    ▼ "3": {
        "name": "earpods",
        "price": 7000,
        "available_quantity": 300
    },
    ▼ "4": {
        "name": "headphones",
        "price": 3000,
        "available_quantity": 120
    },
    ▼ "5": {
        "name": "type c chanrger",
        "price": 1250,
        "available_quantity": 700
```

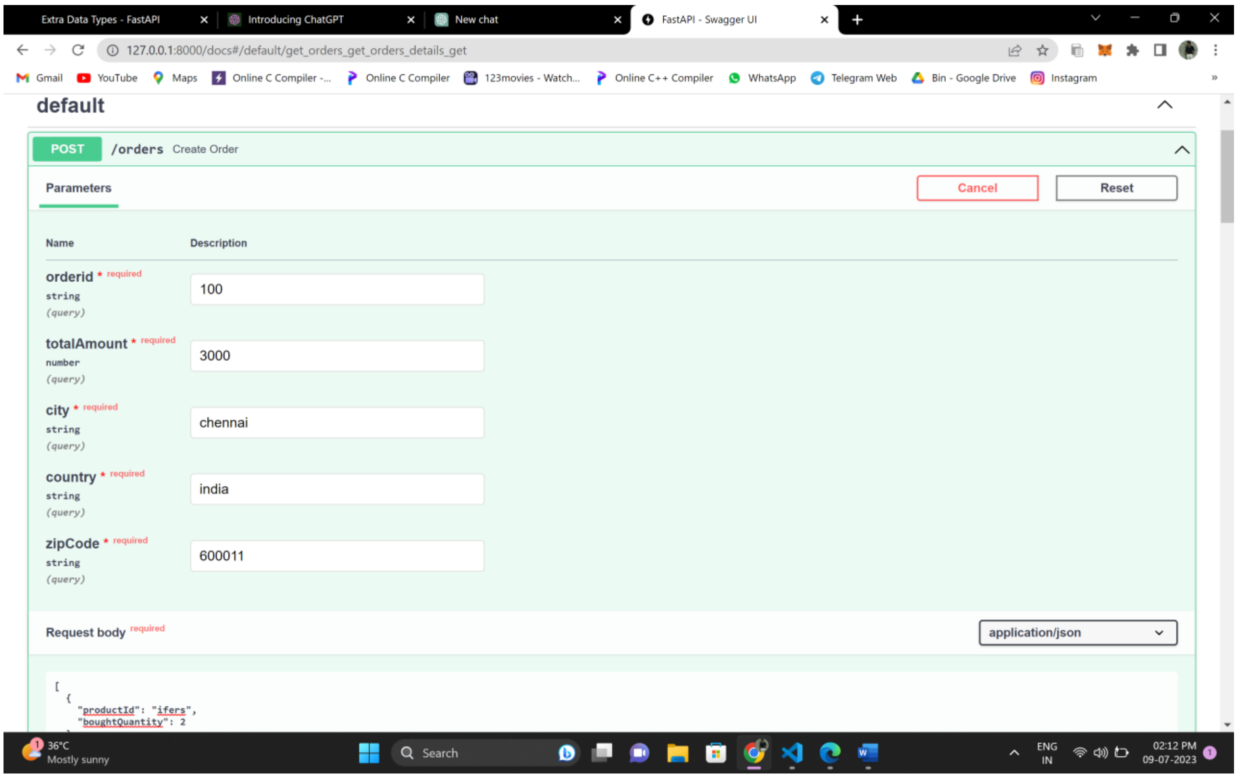
2. API to Create a new order. Each order should have these properties -
   a. Timestamp
   b. Items – list of items bought in the Order. Each record in this array would have these properties -
      i. productId
      ii. boughtQuantity
   c. Total amount
   d. User Address – nested object having these properties -
      i. City
      ii. Country
      iii. Zip Code

CODE:

```
from fastapi import FastAPI, Body
from pydantic import BaseModel
from datetime import datetime

app2 = FastAPI()

orders = []

class Item(BaseModel):
    productId: str
    boughtQuantity: int

class UserAddress(BaseModel):
    city: str
    country: str
    zipCode: str

class Order(BaseModel):
    orderID : str
    timestamp: datetime
    items: list[Item]
    totalAmount: float
    userAddress: UserAddress

@app2.post("/orders")
def create_order(
    orderid : str,
    items: list[Item] ,
    totalAmount: float ,
    city: str ,
    country: str ,
    zipCode: str
):
    order = Order(
        orderID=orderid,
        timestamp=datetime.now(),
```

```python
            items=items,
            totalAmount=totalAmount,
            userAddress=UserAddress(city=city, country=country, zipCode=zipCode)
        )

        orders.append(order)
        return {"message": "Order created successfully"}

@app2.get("/get-orders-details")
def get_orders(order_id : str):
    for temp_order in orders:
        if temp_order.orderID == order_id:
            return temp_order
```

(NOTE : uvicorn file_name:app2 –reload)



127.0.0.1:8000/docs#/default/get_orders_get_orders_details_get

default

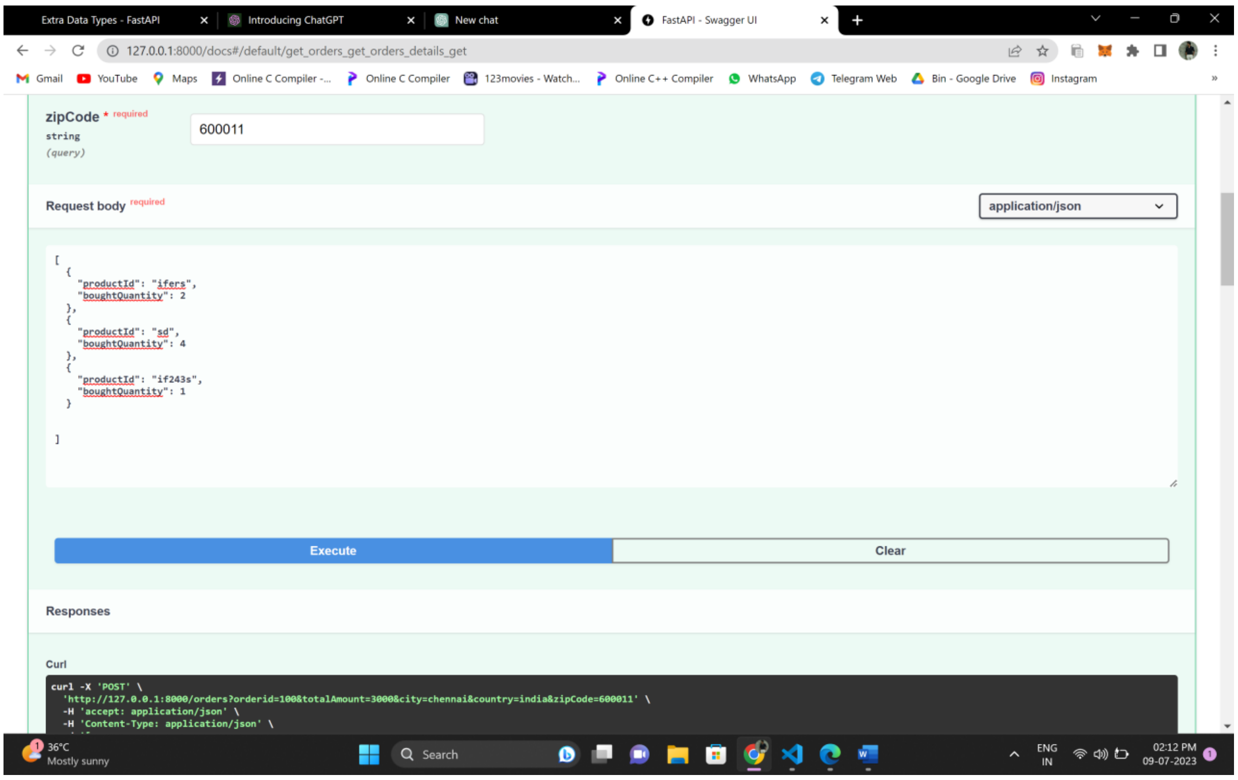
POST /orders Create Order

Parameters

Name	Description
orderid * required string (query)	100
totalAmount * required number (query)	3000
city * required string (query)	chennai
country * required string (query)	india
zipCode * required string (query)	600011

Request body required

```
[
  {
    "productId": "ifers",
    "boughtQuantity": 2
```



zipCode * required
string
(query)

600011

Request body required

application/json

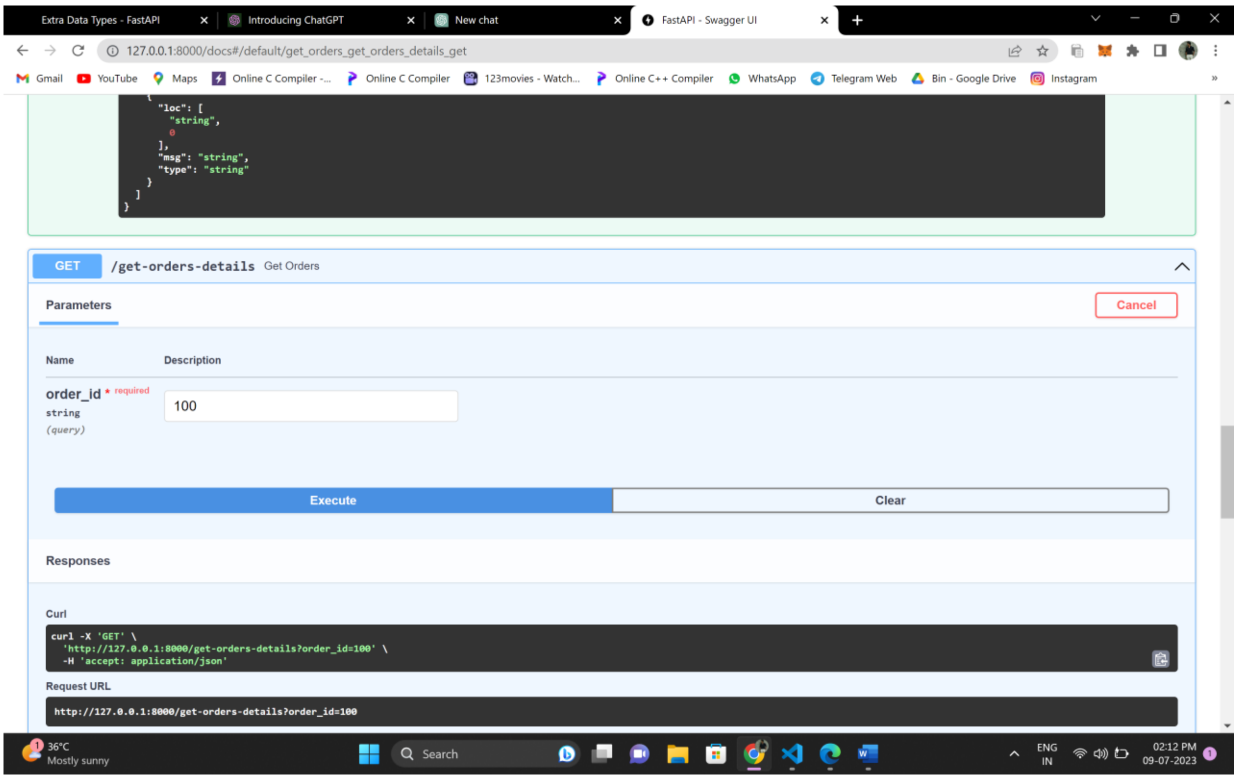
```
[
  {
    "productId": "ifers",
    "boughtQuantity": 2
  },
  {
    "productId": "sd",
    "boughtQuantity": 4
  },
  {
    "productId": "if243s",
    "boughtQuantity": 1
  }

]
```

Execute | Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/orders?orderid=100&totalAmount=3000&city=chennai&country=india&zipCode=600011' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
```



```
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

GET /get-orders-details Get Orders

Parameters | Cancel

Name	Description
order_id * required string (query)	100

Execute | Clear

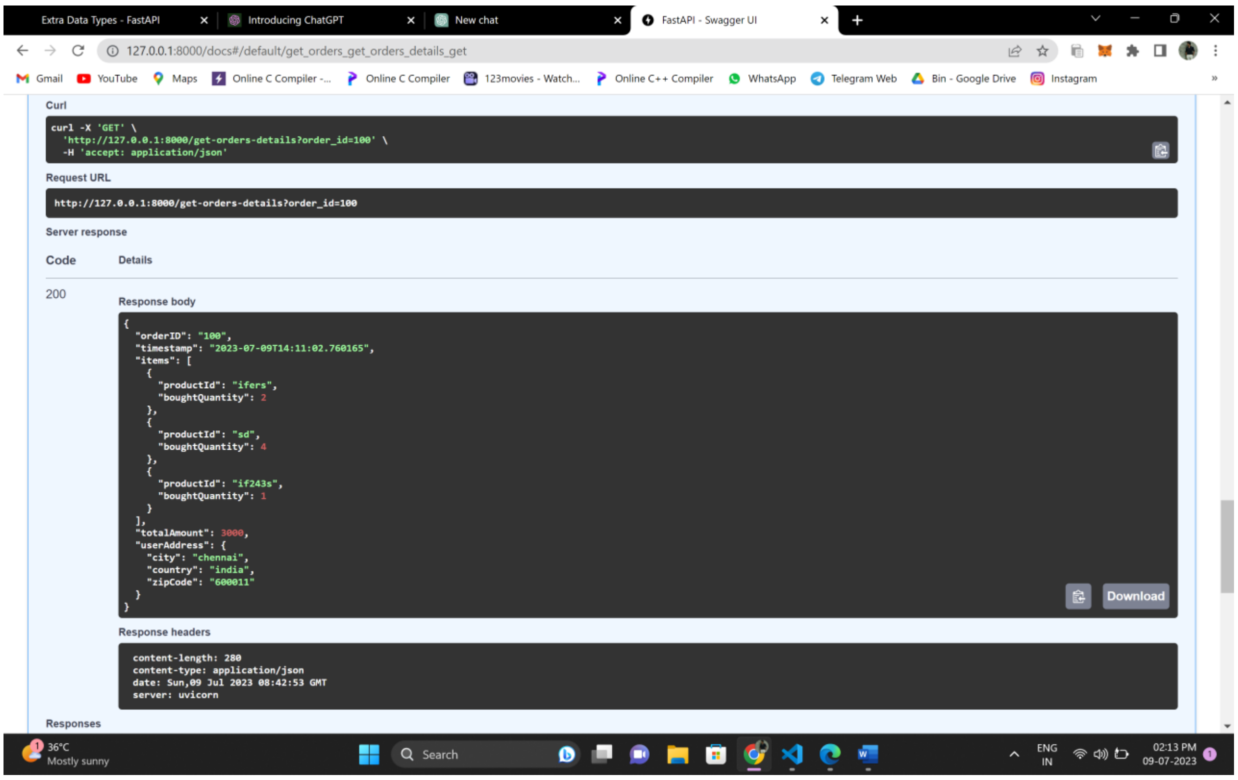
Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/get-orders-details?order_id=100' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/get-orders-details?order_id=100
```



Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/get-orders-details?order_id=100' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/get-orders-details?order_id=100
```

Server response

Code Details

200

Response body

```
{
  "orderID": "100",
  "timestamp": "2023-07-09T14:11:02.760165",
  "items": [
    {
      "productId": "ifers",
      "boughtQuantity": 2
    },
    {
      "productId": "sd",
      "boughtQuantity": 4
    },
    {
      "productId": "if243s",
      "boughtQuantity": 1
    }
  ],
  "totalAmount": 3000,
  "userAddress": {
    "city": "chennai",
    "country": "india",
    "zipCode": "600011"
  }
}
```

Response headers

```
content-length: 280
content-type: application/json
date: Sun,09 Jul 2023 08:42:53 GMT
server: uvicorn
```

Responses

3. API to fetch all orders from the system. Implement pagination using limit and offset.

(note : for this api you need to additional install pagination

pip install fastapi-pagination[fastapi])

CODE:

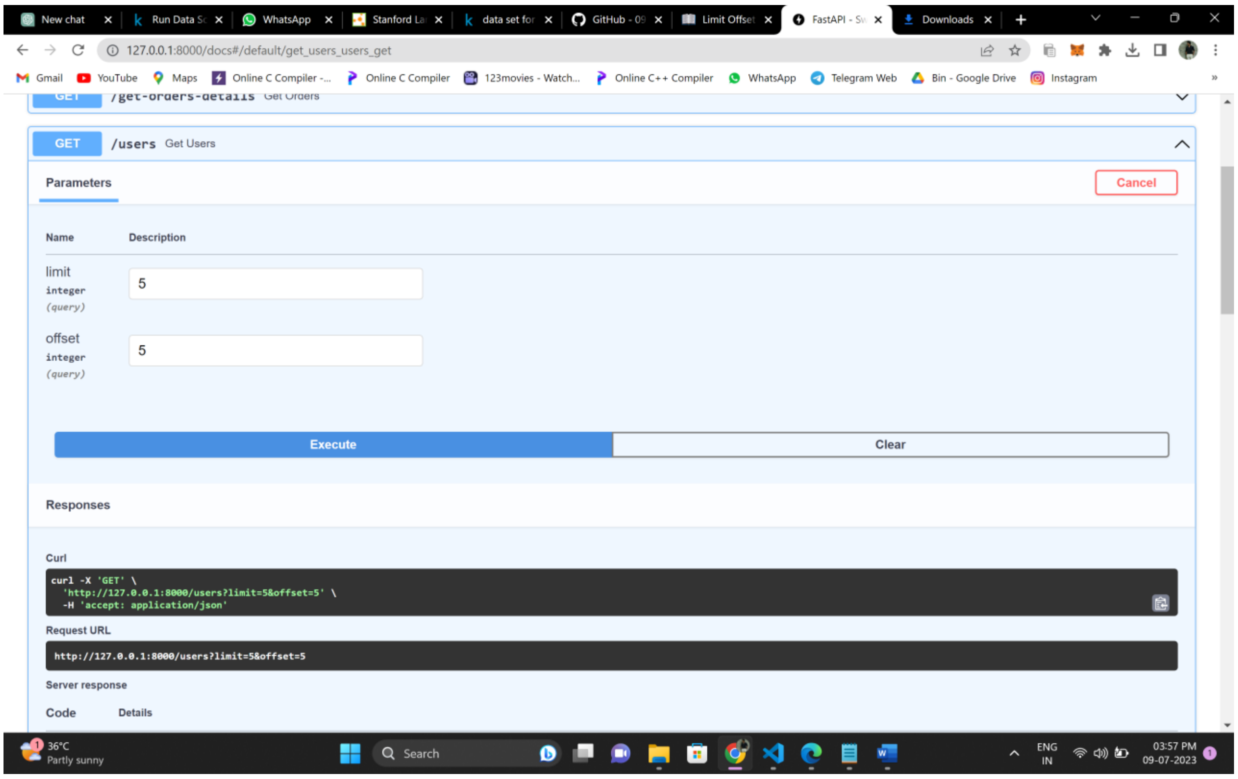
```
from typing import List
from fastapi import FastAPI
from pydantic import BaseModel
from fastapi_pagination import LimitOffsetPage, add_pagination, paginate
from datetime import datetime

app3 = FastAPI()
add_pagination(app3)

class Item(BaseModel):
    productId: str
    boughtQuantity: int

class UserAddress(BaseModel):
    city: str
    country: str
    zipCode: str

class Order(BaseModel):
    orderID: str
    timestamp: datetime = None
    items: List[Item]
    totalAmount: float
    userAddress: UserAddress

    class Config:
        arbitrary_types_allowed = True

orders = [
    Order(
        orderID="ORD12345",
        timestamp=datetime.now(),
        items=[Item(productId="P12345", boughtQuantity=1)],
        totalAmount=10.0,
        userAddress=UserAddress(city="New York", country="United States",
zipCode="12345")
    ),
    Order(
        orderID="ORD23456",
        timestamp=datetime.now(),
        items=[Item(productId="P23456", boughtQuantity=2)],
        totalAmount=20.0,
        userAddress=UserAddress(city="London", country="United Kingdom",
zipCode="12345")
    ),
    Order(
        orderID="ORD34567",
        timestamp=datetime.now(),
        items=[Item(productId="P34567", boughtQuantity=3)],
        totalAmount=30.0,
        userAddress=UserAddress(city="Paris", country="France",
zipCode="12345")
    ),
    Order(
        orderID="ORD45678",
        timestamp=datetime.now(),
        items=[Item(productId="P45678", boughtQuantity=4)],
        totalAmount=40.0,
        userAddress=UserAddress(city="Tokyo", country="Japan", zipCode="12345")
    ),
    Order(
        orderID="ORD56789",
        timestamp=datetime.now(),
        items=[Item(productId="P56789", boughtQuantity=5)],
        totalAmount=50.0,
        userAddress=UserAddress(city="Sydney", country="Australia",
zipCode="12345")
    ),
    Order(
        orderID="ORD67890",
        timestamp=datetime.now(),
        items=[Item(productId="P67890", boughtQuantity=6)],
        totalAmount=60.0,
        userAddress=UserAddress(city="Berlin", country="Germany",
zipCode="12345")
```

```
    ),
    Order(
        orderID="ORD78901",
        timestamp=datetime.now(),
        items=[Item(productId="P78901", boughtQuantity=7)],
        totalAmount=70.0,
        userAddress=UserAddress(city="Toronto", country="Canada",
zipCode="12345")
    ),
    Order(
        orderID="ORD89012",
        timestamp=datetime.now(),
        items=[Item(productId="P89012", boughtQuantity=8)],
        totalAmount=80.0,
        userAddress=UserAddress(city="Rome", country="Italy", zipCode="12345")
    ),
    Order(
        orderID="ORD90123",
        timestamp=datetime.now(),
        items=[Item(productId="P90123", boughtQuantity=9)],
        totalAmount=90.0,
        userAddress=UserAddress(city="Barcelona", country="Spain",
zipCode="12345")
    ),
    Order(
        orderID="ORD01234",
        timestamp=datetime.now(),
        items=[Item(productId="P01234", boughtQuantity=10)],
        totalAmount=100.0,
        userAddress=UserAddress(city="New Delhi", country="India",
zipCode="12345")
    ),
    Order(
        orderID="ORD12345",
        timestamp=datetime.now(),
        items=[Item(productId="P12345", boughtQuantity=1)],
        totalAmount=10.0,
        userAddress=UserAddress(city="New York", country="United States",
zipCode="12345")
    ),
    Order(
        orderID="ORD23456",
        timestamp=datetime.now(),
        items=[Item(productId="P23456", boughtQuantity=2)],
        totalAmount=20.0,
        userAddress=UserAddress(city="London", country="United Kingdom",
zipCode="12345")
    ),
    Order(
        orderID="ORD34567",
        timestamp=datetime.now(),
        items=[Item(productId="P34567", boughtQuantity=3)],
        totalAmount=30.0,
        userAddress=UserAddress(city="Paris", country="France",
zipCode="12345")
    ),
    Order(
        orderID="ORD45678",
        timestamp=datetime.now(),
        items=[Item(productId="P45678", boughtQuantity=4)],
        totalAmount=40.0,
        userAddress=UserAddress(city="Tokyo", country="Japan", zipCode="12345")
    ),
    Order(
        orderID="ORD56789",
        timestamp=datetime.now(),
        items=[Item(productId="P56789", boughtQuantity=5)],
        totalAmount=50.0,
        userAddress=UserAddress(city="Sydney", country="Australia",
zipCode="12345")
    ),
    Order(
        orderID="ORD67890",
        timestamp=datetime.now(),
        items=[Item(productId="P67890", boughtQuantity=6)],
        totalAmount=60.0,
        userAddress=UserAddress(city="Berlin", country="Germany",
zipCode="12345")
    )
]

@app3.post("/orders")
def create_order(
    orderid: str,
    items: List[Item],
    totalAmount: float,
    city: str,
    country: str,
    zipCode: str
):
    order = Order(
        orderID=orderid,
        timestamp=datetime.now(),
        items=items,
        totalAmount=totalAmount,
        userAddress=UserAddress(city=city, country=country, zipCode=zipCode)
    )

    orders.append(order)
    return {"message": "Order created successfully"}

@app3.get("/get-orders-details")
def get_orders(order_id: str):
    for temp_order in orders:
        if temp_order.orderID == order_id:
            return temp_order
```

```python
@app3.get("/users")
def get_users() -> LimitOffsetPage[Order]:
    return paginate(orders)
```

```
uvicorn file_name:app3 -reload
```

SAMPLE 1:


127.0.0.1:8000/docs#/default/get_users_users_get

GET /users Get Users

Parameters

Name	Description
limit integer *(query)*	5
offset integer *(query)*	5

Responses

Curl

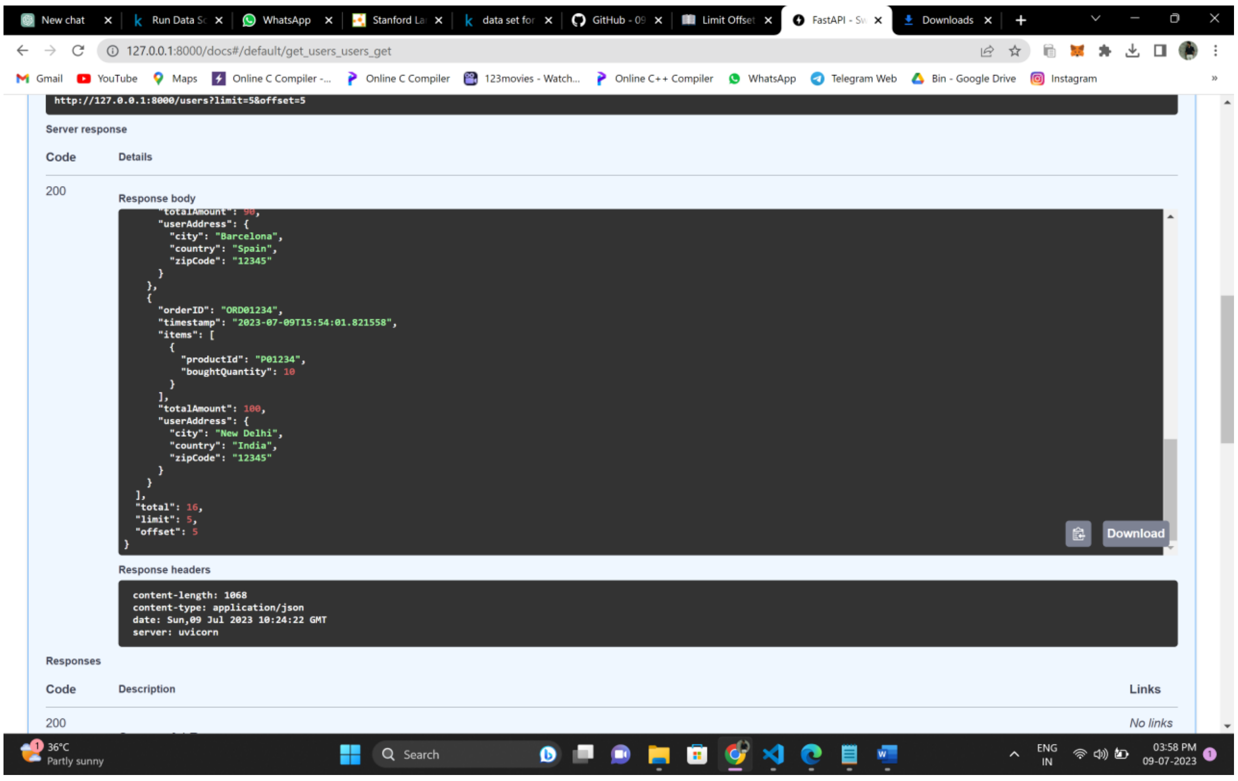
```
curl -X 'GET' \
  'http://127.0.0.1:8000/users?limit=5&offset=5' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/users?limit=5&offset=5
```

Server response

Code Details


```
http://127.0.0.1:8000/users?limit=5&offset=5
```

Server response

Code Details

200

Response body

```
      "totalAmount": 90,
      "userAddress": {
        "city": "Barcelona",
        "country": "Spain",
        "zipCode": "12345"
      }
    },
    {
      "orderID": "ORD01234",
      "timestamp": "2023-07-09T15:54:01.821558",
      "items": [
        {
          "productId": "P01234",
          "boughtQuantity": 10
        }
      ],
      "totalAmount": 100,
      "userAddress": {
        "city": "New Delhi",
        "country": "India",
        "zipCode": "12345"
      }
    }
  ],
  "total": 16,
  "limit": 5,
  "offset": 5
}
```

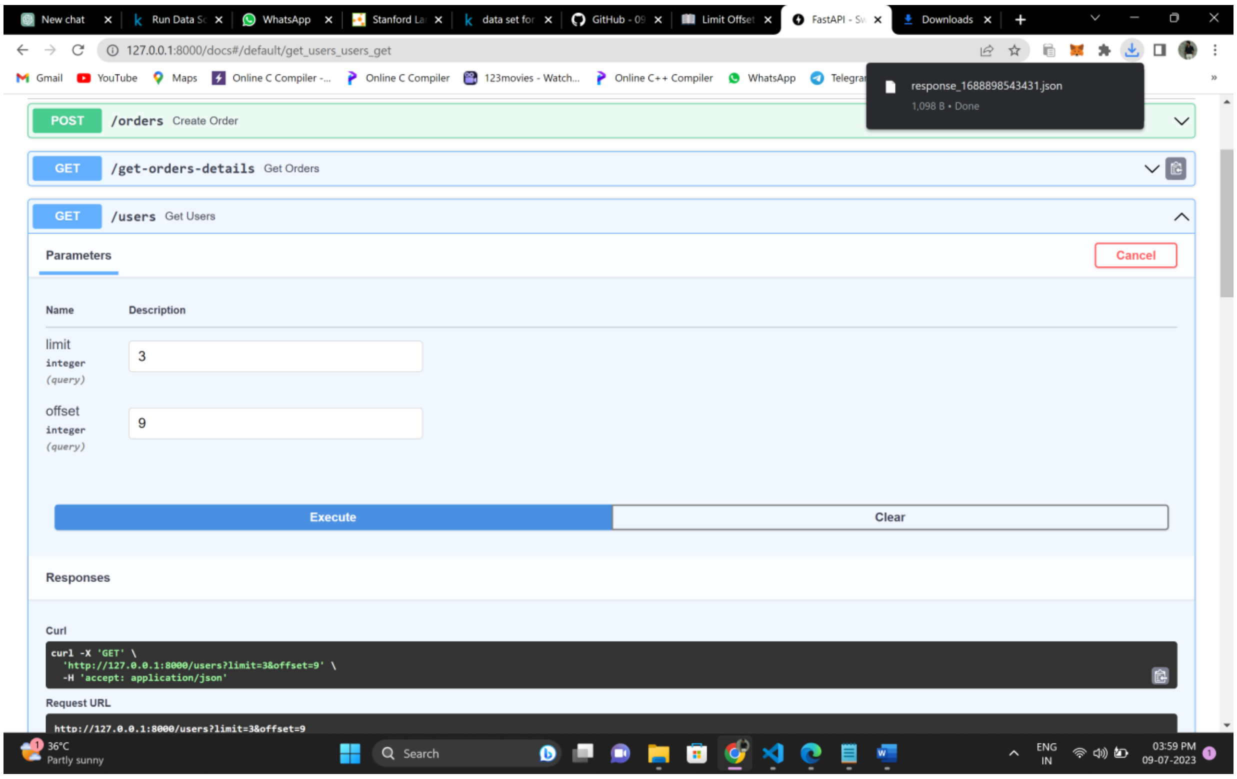
Response headers

```
content-length: 1068
content-type: application/json
date: Sun,09 Jul 2023 10:24:22 GMT
server: uvicorn
```

Responses

Code	Description	Links
200		*No links*

SAMPLE 2:



127.0.0.1:8000/docs#/default/get_users_users_get

response_1688898543431.json
1,098 B • Done

POST /orders Create Order

GET /get-orders-details Get Orders

GET /users Get Users

Parameters

Name	Description
limit integer (query)	3
offset integer (query)	9

Execute Clear

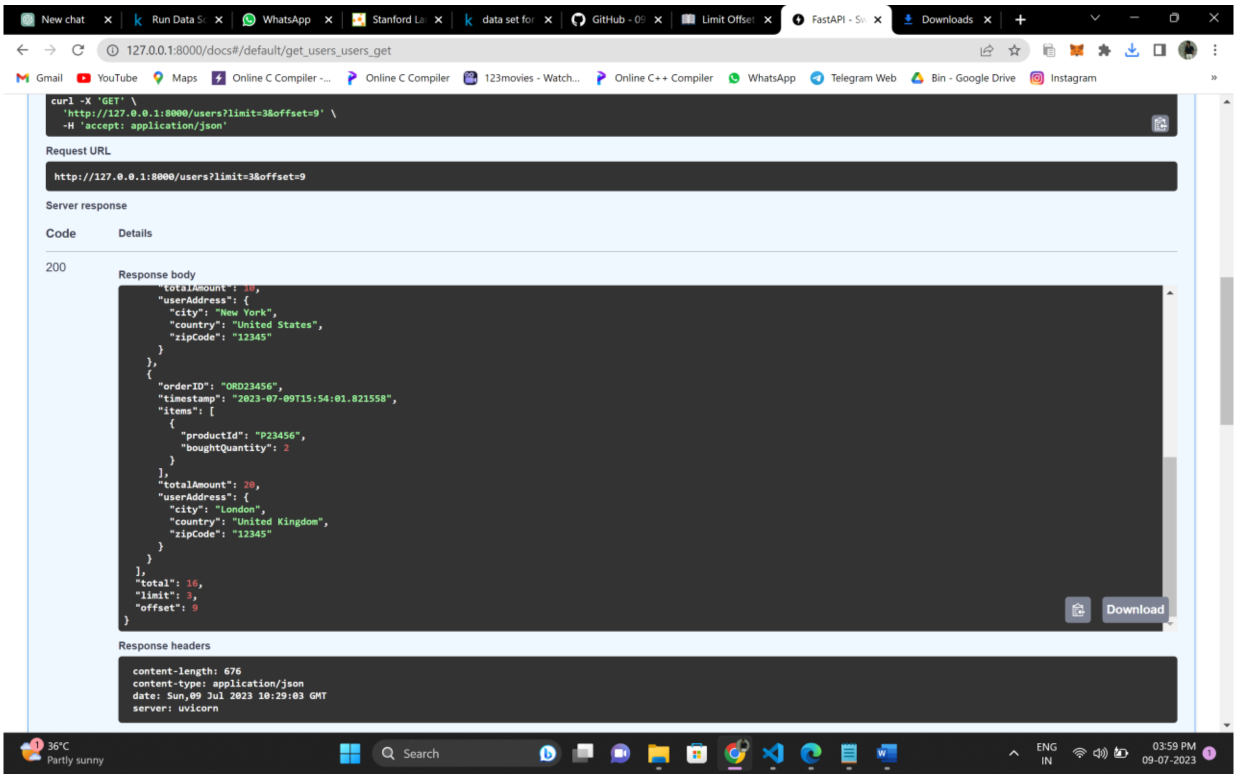
Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/users?limit=3&offset=9' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/users?limit=3&offset=9
```



```
curl -X 'GET' \
  'http://127.0.0.1:8000/users?limit=3&offset=9' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/users?limit=3&offset=9
```

Server response

Code Details

200

Response body

```
        "totalAmount": 10,
        "userAddress": {
          "city": "New York",
          "country": "United States",
          "zipCode": "12345"
        }
      },
      {
        "orderID": "ORD23456",
        "timestamp": "2023-07-09T15:54:01.821558",
        "items": [
          {
            "productId": "P23456",
            "boughtQuantity": 2
          }
        ],
        "totalAmount": 20,
        "userAddress": {
          "city": "London",
          "country": "United Kingdom",
          "zipCode": "12345"
        }
      }
    ],
    "total": 16,
    "limit": 3,
    "offset": 9
}
```

Download

Response headers

```
content-length: 676
content-type: application/json
date: Sun,09 Jul 2023 10:29:03 GMT
server: uvicorn
```

The downaloded json body after setting the limit and the offset

```
{
  "items": [
    {
      "orderID": "ORD01234",
      "timestamp": "2023-07-09T15:54:01.821558",
      "items": [
        {
          "productId": "P01234",
          "boughtQuantity": 10
        }
      ],
      "totalAmount": 100,
      "userAddress": {
        "city": "New Delhi",
        "country": "India",
        "zipCode": "12345"
      }
    },
    {
      "orderID": "ORD12345",
      "timestamp": "2023-07-09T15:54:01.821558",
      "items": [
        {
          "productId": "P12345",
          "boughtQuantity": 1
        }
      ],
      "totalAmount": 10,
      "userAddress": {
        "city": "New York",
        "country": "United States",
        "zipCode": "12345"
      }
    },
    {
      "orderID": "ORD23456",
      "timestamp": "2023-07-09T15:54:01.821558",
      "items": [
        {
          "productId": "P23456",
          "boughtQuantity": 2
        }
      ],
      "totalAmount": 20,
      "userAddress": {
        "city": "London",
        "country": "United Kingdom",
        "zipCode": "12345"
      }
    }
  ],
  "total": 16,
  "limit": 3,
  "offset": 9
}
```

4. API to fetch a single order from the system using Order ID

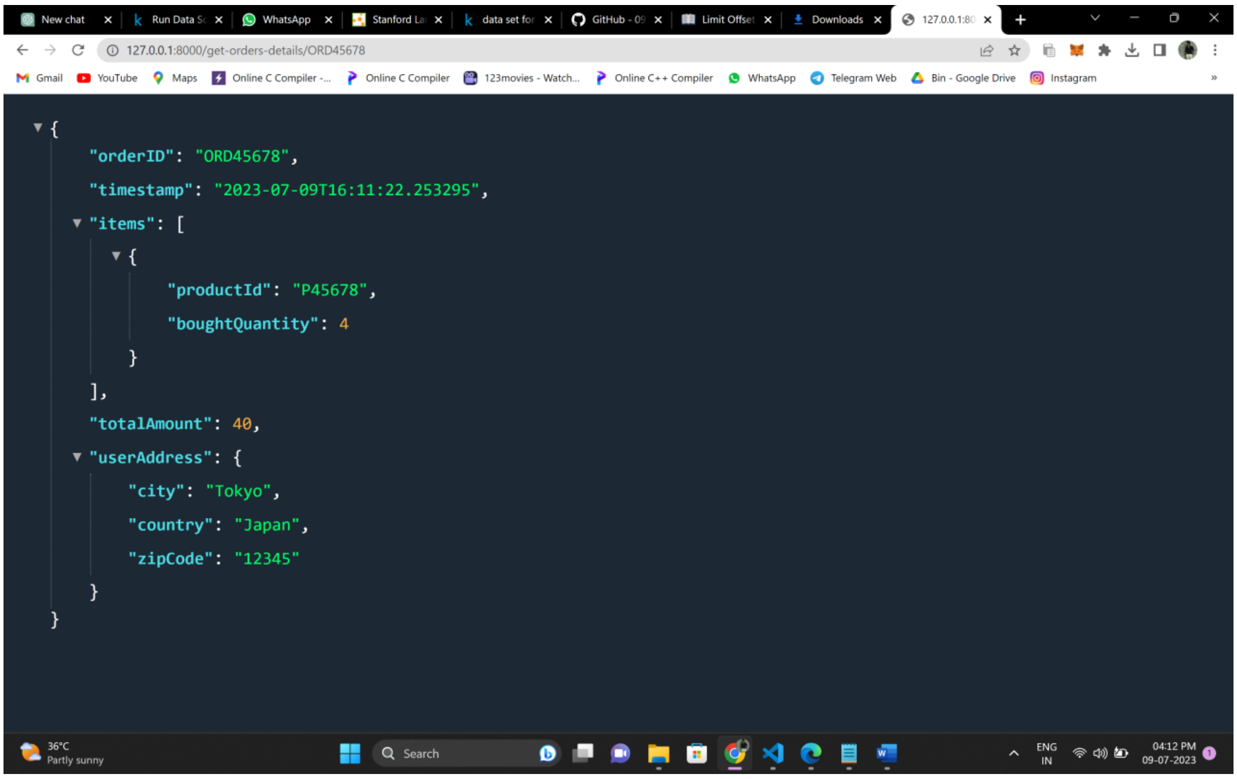
CODE:

```
from fastapi import FastAPI
from pydantic import BaseModel
from datetime import datetime

app4 = FastAPI()

class Item(BaseModel):
    productId: str
    boughtQuantity: int

class UserAddress(BaseModel):
    city: str
    country: str
    zipCode: str



class Order(BaseModel):
    orderID: str
    timestamp: datetime = None
    items: list[Item]
    totalAmount: float
    userAddress: UserAddress






orders = [
    Order(
        orderID="ORD12345",
        timestamp=datetime.now(),
        items=[Item(productId="P12345", boughtQuantity=1)],
        totalAmount=10.0,
        userAddress=UserAddress(city="New York", country="United States",
zipCode="12345")
    ),
    Order(
        orderID="ORD23456",
        timestamp=datetime.now(),
        items=[Item(productId="P23456", boughtQuantity=2)],
        totalAmount=20.0,
```

```python
        userAddress=UserAddress(city="London", country="United Kingdom",
zipCode="12345")
    ),
    Order(
        orderID="ORD34567",
        timestamp=datetime.now(),
        items=[Item(productId="P34567", boughtQuantity=3)],
        totalAmount=30.0,
        userAddress=UserAddress(city="Paris", country="France",
zipCode="12345")
    ),
    Order(
        orderID="ORD45678",
        timestamp=datetime.now(),
        items=[Item(productId="P45678", boughtQuantity=4)],
        totalAmount=40.0,
        userAddress=UserAddress(city="Tokyo", country="Japan", zipCode="12345")
    )
]

@app4.post("/create-orders")
def create_order(
    orderid: str,
    items: list[Item],
    totalAmount: float,
    city: str,
    country: str,
    zipCode: str
):
    order = Order(
        orderID=orderid,
        timestamp=datetime.now(),
        items=items,
        totalAmount=totalAmount,
        userAddress=UserAddress(city=city, country=country, zipCode=zipCode)
    )

    orders.append(order)
    return {"message": "Order created successfully"}

@app4.get("/get-orders-details/{order_id}")
def get_orders(order_id : str):
    for temp_order in orders:
        if temp_order.orderID == order_id:
            return temp_order
```

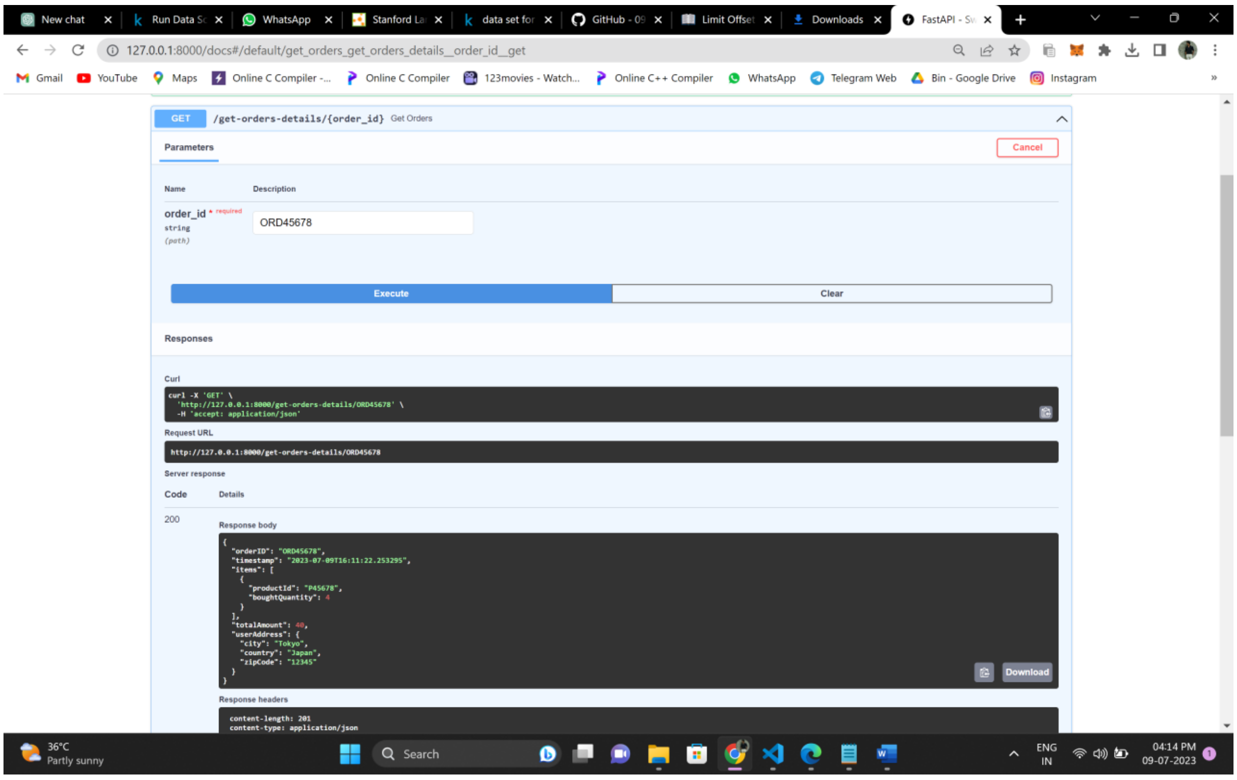
SAMPLE 1:

Through the URL (JSON beatify viewer used)


127.0.0.1:8000/get-orders-details/ORD45678

```
{
    "orderID": "ORD45678",
    "timestamp": "2023-07-09T16:11:22.253295",
    "items": [
        {
            "productId": "P45678",
            "boughtQuantity": 4
        }
    ],
    "totalAmount": 40,
    "userAddress": {
        "city": "Tokyo",
        "country": "Japan",
        "zipCode": "12345"
    }
}
```

<u>Through the docs</u>


127.0.0.1:8000/docs#/default/get_orders_get_orders_details__order_id__get

GET /get-orders-details/{order_id} Get Orders

Parameters

Name	Description
order_id * required string (path)	ORD45678

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/get-orders-details/ORD45678' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/get-orders-details/ORD45678
```

Server response

Code: 200

Response body

```
{
  "orderID": "ORD45678",
  "timestamp": "2023-07-09T16:11:22.253295",
  "items": [
    {
      "productId": "P45678",
      "boughtQuantity": 4
    }
  ],
  "totalAmount": 40,
  "userAddress": {
    "city": "Tokyo",
    "country": "Japan",
    "zipCode": "12345"
  }
}
```

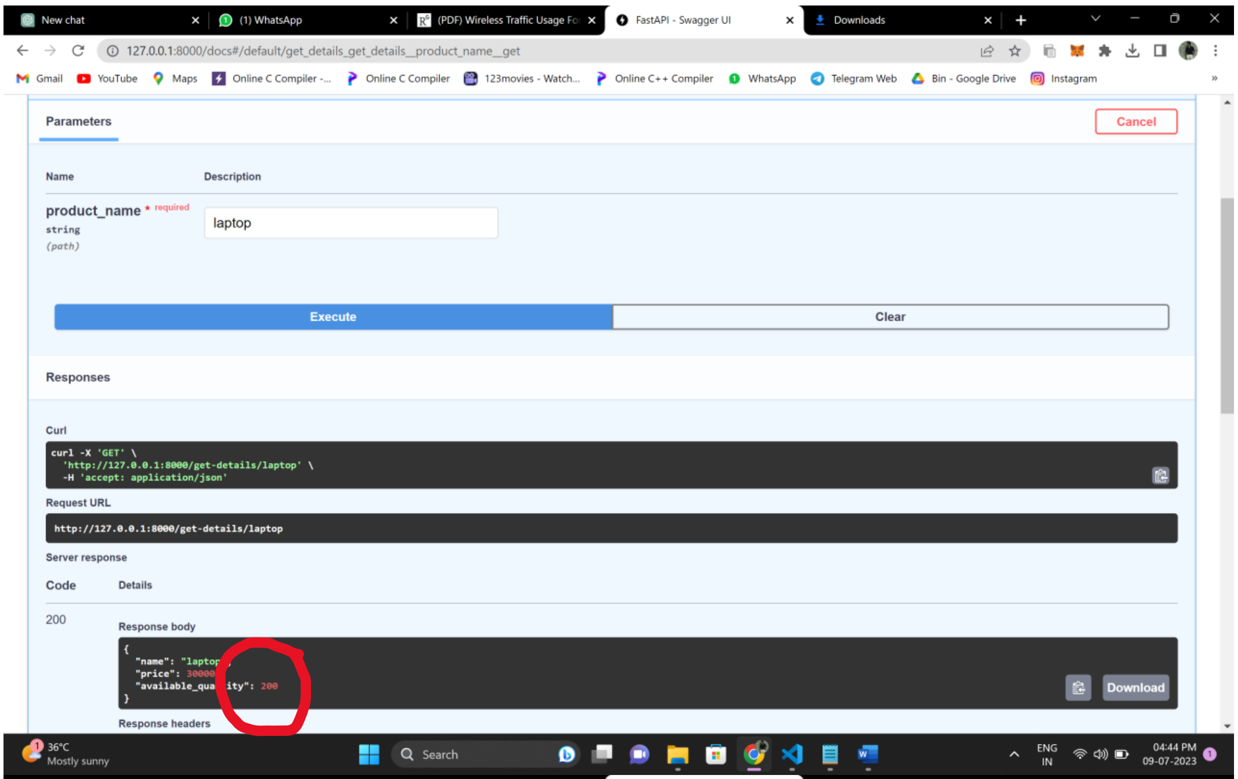
Response headers

```
content-length: 201
content-type: application/json
```

5. API to update a product when updating the available quantity for the product.

CODE:

```
from fastapi import FastAPI

app5 = FastAPI()


products = [
        {"name":"laptop","price":30000,"available_quantity":200},
        {"name":"computer","price":40000,"available_quantity":150},
        {"name":"earpods","price":7000,"available_quantity":300},
        {"name":"headphones","price":3000,"available_quantity":120},
        {"name":"type c chanrger","price":1250,"available_quantity":700},
        {"name":"mobile","price":20000,"available_quantity":1000},
        {"name":"ipod","price":60000,"available_quantity":190},
        {"name":"smartwatch","price":5000,"available_quantity":250},
        {"name":"Bluetooth speaker","price":2500,"available_quantity":400},
        {"name":"external hard drive","price":6000,"available_quantity":180},
        {"name":"printer","price":8000,"available_quantity":90},
        {"name":"wireless mouse","price":1000,"available_quantity":600},
        {"name":"keyboard","price":1500,"available_quantity":450},
        {"name":"portable SSD","price":7000,"available_quantity":350},
        {"name":"gaming console","price":25000,"available_quantity":80},
        {"name":"graphics card","price":15000,"available_quantity":200},
        {"name":"smartphone tripod","price":500,"available_quantity":800},
        {"name":"USB flash drive","price":500,"available_quantity":1000},
        {"name":"webcam","price":3000,"available_quantity":250},
        {"name":"Bluetooth earphones","price":2500,"available_quantity":500}
]
@app5.get("/update-product-quantity")
def update_quantity(product_name : str,New_quantity_to_be_added : int):
    for product in products:
        if product["name"] == product_name:
            product["available_quantity"] += New_quantity_to_be_added
            return product
    return {"error":"this product is a new product "}

@app5.get("/get-details/{product_name}")
def get_details(product_name : str):
    for product in products:
        if product["name"] == product_name:
            return product
    return {"error":"no such product available"}
```



127.0.0.1:8000/docs#/default/get_details_get_details__product_name__get

Parameters

Name	Description
product_name * required string (path)	laptop

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/get-details/laptop' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/get-details/laptop
```

Server response

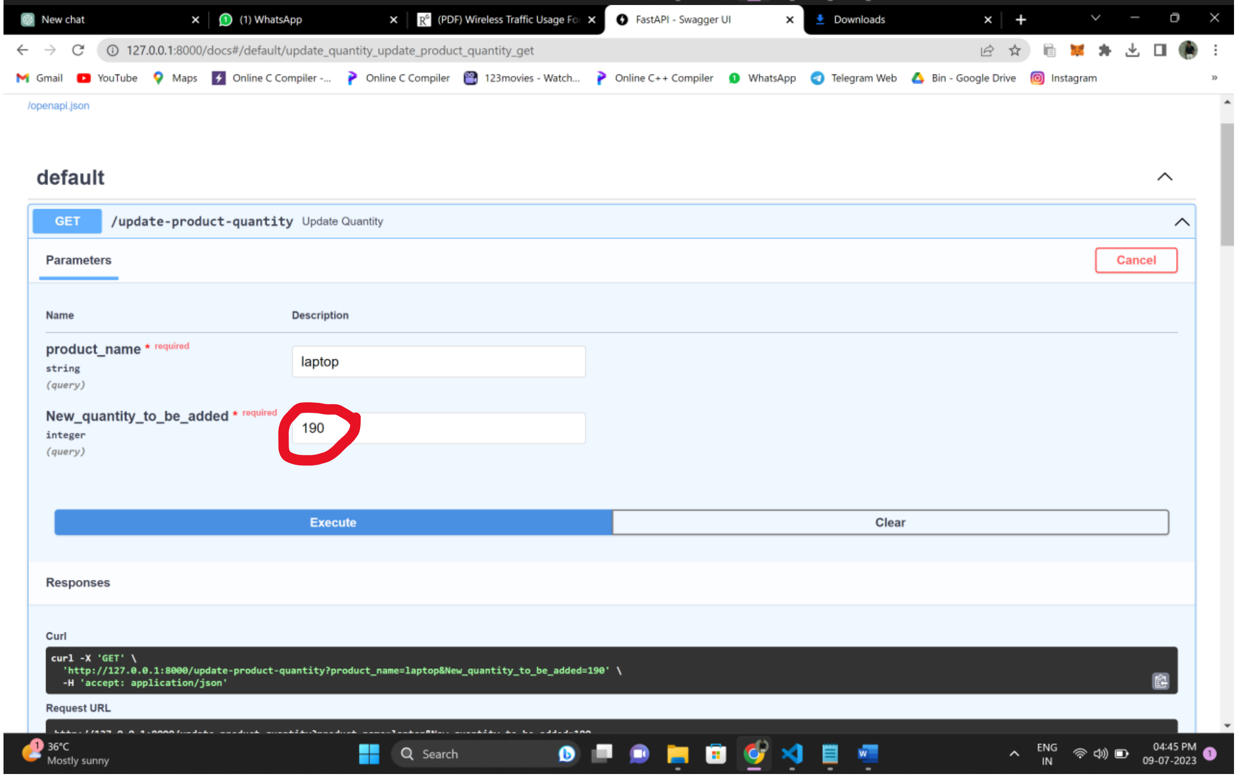
Code Details

200

Response body

```
{
  "name": "laptop",
  "price": 30000
  "available_qua ity": 200
}
```

Response headers



127.0.0.1:8000/docs#/default/update_quantity_update_product_quantity_get

/openapi.json

default

GET /update-product-quantity Update Quantity

Parameters

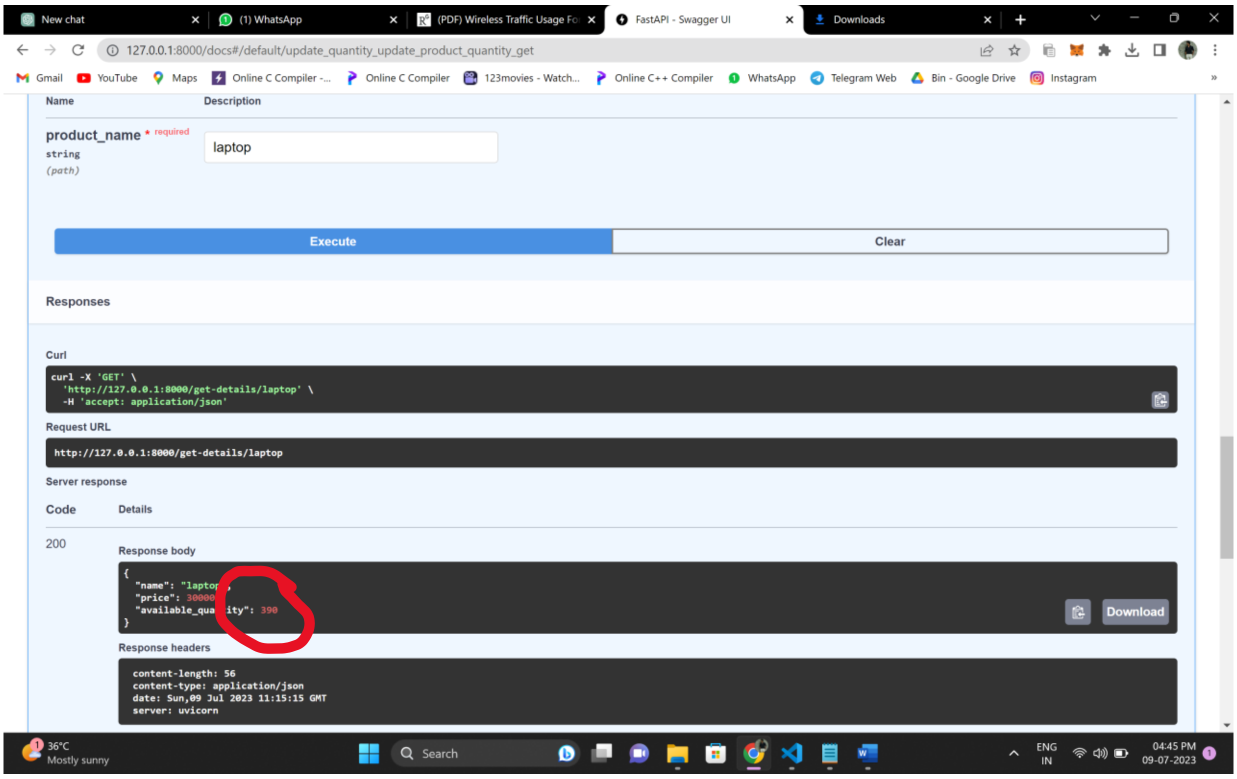
Name	Description
product_name * required string (query)	laptop
New_quantity_to_be_added * required integer (query)	190

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/update-product-quantity?product_name=laptop&New_quantity_to_be_added=190' \
  -H 'accept: application/json'
```

Request URL



127.0.0.1:8000/docs#/default/update_quantity_update_product_quantity_get

Name | Description

product_name * required
string
(path)

laptop

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/get-details/laptop' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/get-details/laptop
```

Server response

Code | Details

200

Response body

```
{
  "name": "laptop",
  "price": 30000,
  "available_quantity": 390
}
```

Response headers

```
 content-length: 56
 content-type: application/json
 date: Sun,09 Jul 2023 11:15:15 GMT
 server: uvicorn
```

- .