

**Name : Surajkumar Yadav**

**Email : [sy820860@gmail.com](mailto:sy820860@gmail.com)**

**Phone : +917756956865**

**Link : [https://github.com/suraj4502/Trading\\_APIs](https://github.com/suraj4502/Trading_APIs)**

Hello Arjun Sir,

I am submitting this Document in fulfillment of the assignment for the data Engineer intern Role at your company.

- **Understanding the Problem.**

I started the assignment by understanding the problem and what is being asked to solve. After thoroughly analyzing the question I understood that the task is to build a REST API using the FastAPI framework in Python. The API should provide endpoints to retrieve a list of trades, retrieve a single trade by ID, search for trades using specific fields, and filter trades based on various optional query parameters. A Pydantic model representing a trade is provided, and a mocked database interaction layer should be implemented to generate trade data. If the API supports pagination and sorting on the list of trades then it would be a plus.

- **My approach to solve the Problem.**

My first approach was to use a mongodb atlas database and store mock trades in it and retrieve accordingly, but later I thought that it would be best

and easy to store the mocked trades in a list, as I didn't wanted to waste time on connecting and creating databases on the cloud.

So in a list called **database** and Created 100 random trades datapoints using a for loop. The list is used to store the trade records as dictionaries. Each trade record is represented by a dictionary with various fields such as trade ID, asset class, counterparty, instrument ID, trade date and time, trade details, and trader information.

- **Solution Description.**

The provided code is a FastAPI application that implements an API for managing trade records. It includes endpoints for retrieving all trades, retrieving trades by ID, filtering trades based on specific conditions, performing advanced searches, and implementing pagination for trade records.

The API endpoints are defined using FastAPI decorators (@app.get) and handle HTTP GET requests. The response from the endpoints is returned as JSON data.

### **1. Root Endpoint**

- URL: http://127.0.0.1:8000/
- Method: GET
- Description: Returns a message and instructions about the API's available endpoints.
- Response: JSON data containing a message with instructions for using the API.
- Output: on postman

```

"Message": "Hello Team 🙌 Suraj here and this is the API that you asked for. Please look into the instructions below ",
"👉Instructions": {
  "🟢To get all trades": "http://127.0.0.1:8000/trades",
  "🟢To get trade by id": "http://127.0.0.1:8000/trades/{id [from 1 to 102]} ,eg == 'http://127.0.0.1:8000/trades/45'",
  "🟢Trade by Specific condition": "http://127.0.0.1:8000/specificTrades?{condition}={condition_value}, eg == 'http://127.0.0.1:8000/specificTrades?trader=Suraj%20Yadav'",
  "🟢Advance_Filetering": "http://127.0.0.1:8000/advancedSearch?{condition}={condition_value}, eg == 'http://127.0.0.1:8000/advancedSearch?assetClass=FX'",
  "🟢pagination": " http://127.0.0.1:8000/trades/?page={page_no}&limit={number_of_records}, eg == 'http://127.0.0.1:8000/trades/?page=1&limit=10'"
},
"Thak You": "🙏"

```

## On Browser :

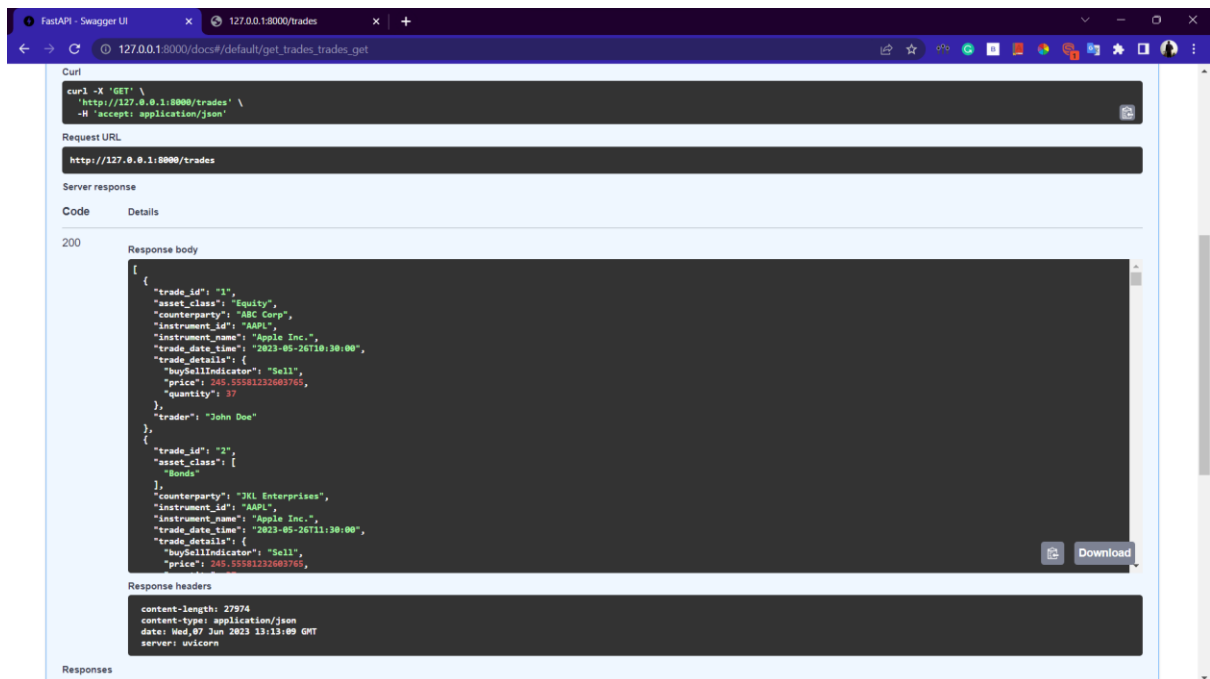
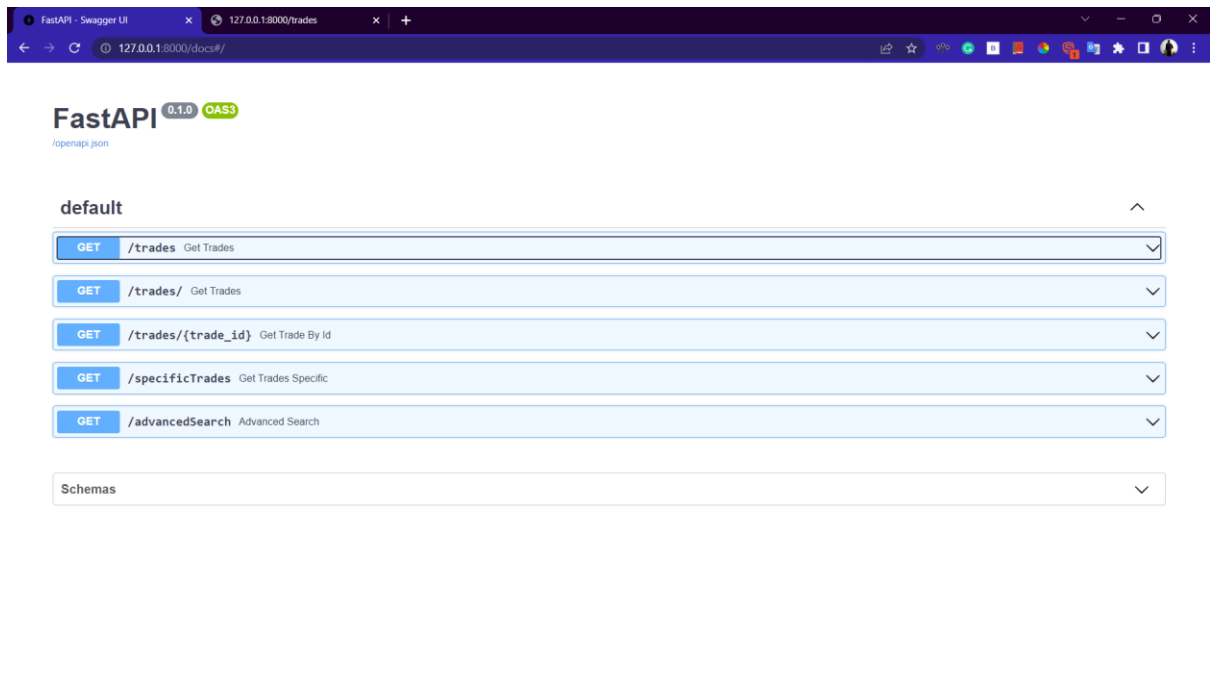
```

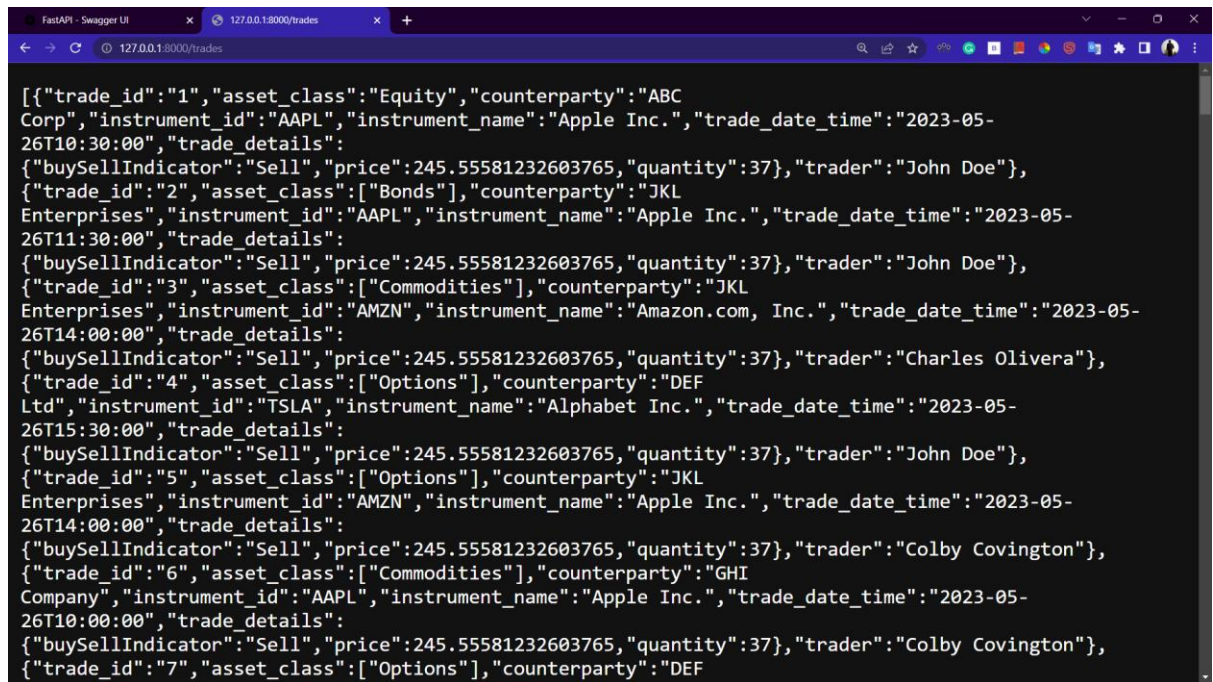
{"Message":"Hello Team 🙌 Suraj here and this is the API that you asked for. Please look into the instructions below ","👉Instructions":{"🟢To get all trades":"http://127.0.0.1:8000/trades","🟢To get trade by id":"http://127.0.0.1:8000/trades/{id [from 1 to 102]} ,eg == 'http://127.0.0.1:8000/trades/45'", "🟢Trade by Specific condition":"http://127.0.0.1:8000/specificTrades?{condition}={condition_value}, eg == 'http://127.0.0.1:8000/specificTrades?trader=Suraj%20Yadav'", "🟢Advance_Filetering":"http://127.0.0.1:8000/advancedSearch?{condition}={condition_value}, eg == 'http://127.0.0.1:8000/advancedSearch?assetClass=FX'", "🟢pagination":" http://127.0.0.1:8000/trades/?page={page_no}&limit={number_of_records}, eg == 'http://127.0.0.1:8000/trades/?page=1&limit=10'"},"Thak You":"🙏"}

```

## 2. Get All Trades

- URL: `/trades`
- Method: GET
- Description: Returns all trade records stored in the database.
- Response: JSON data containing a list of all trade records.
- Output on the browser :

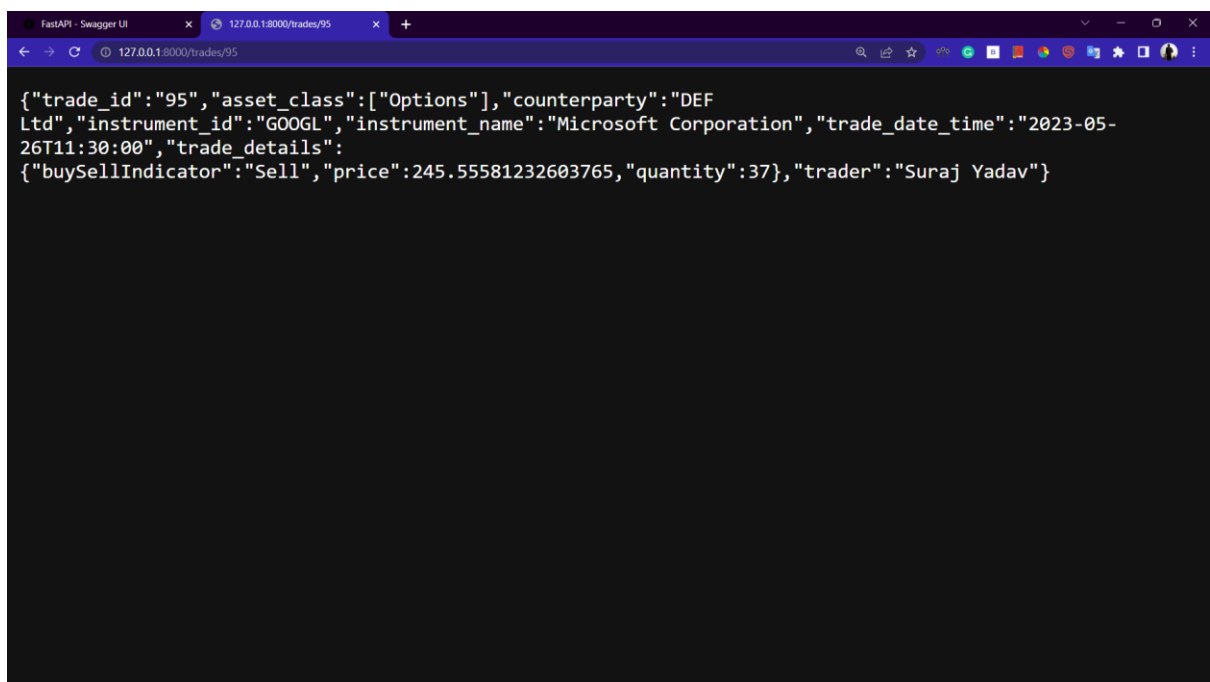
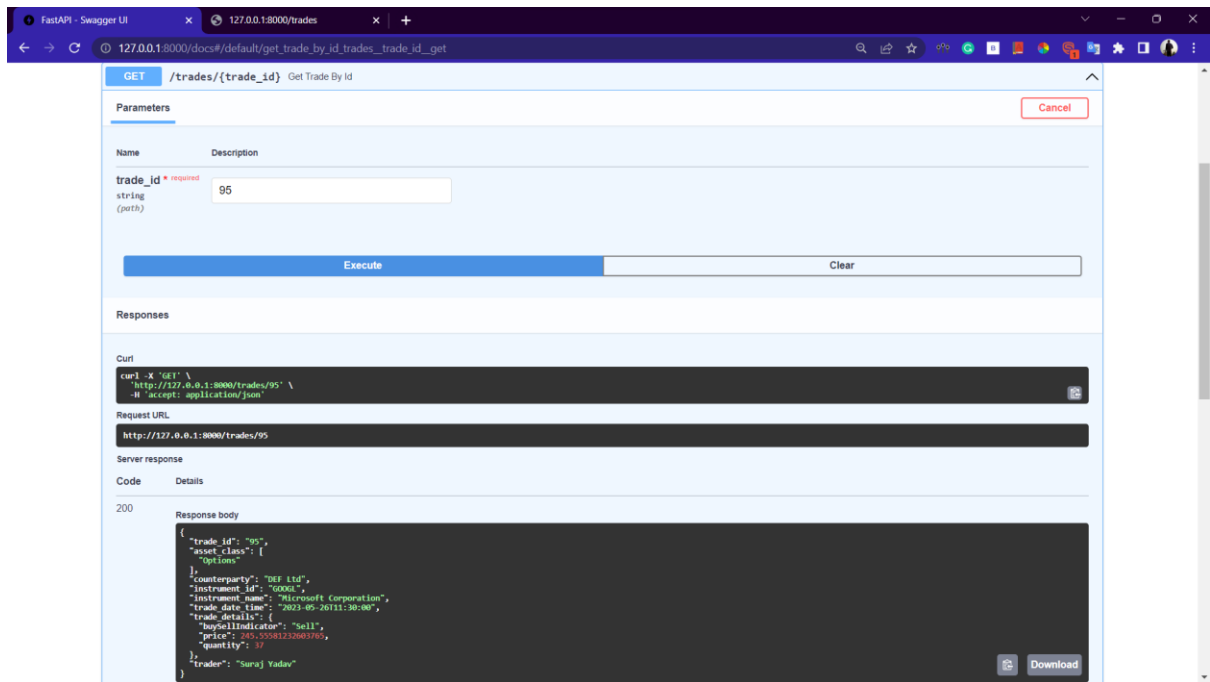




```
[{"trade_id": "1", "asset_class": "Equity", "counterparty": "ABC Corp", "instrument_id": "AAPL", "instrument_name": "Apple Inc.", "trade_date_time": "2023-05-26T10:30:00", "trade_details": {"buySellIndicator": "Sell", "price": 245.55581232603765, "quantity": 37, "trader": "John Doe"}}, {"trade_id": "2", "asset_class": ["Bonds"], "counterparty": "JKL Enterprises", "instrument_id": "AAPL", "instrument_name": "Apple Inc.", "trade_date_time": "2023-05-26T11:30:00", "trade_details": {"buySellIndicator": "Sell", "price": 245.55581232603765, "quantity": 37, "trader": "John Doe"}}, {"trade_id": "3", "asset_class": ["Commodities"], "counterparty": "JKL Enterprises", "instrument_id": "AMZN", "instrument_name": "Amazon.com, Inc.", "trade_date_time": "2023-05-26T14:00:00", "trade_details": {"buySellIndicator": "Sell", "price": 245.55581232603765, "quantity": 37, "trader": "Charles Olivera"}}, {"trade_id": "4", "asset_class": ["Options"], "counterparty": "DEF Ltd", "instrument_id": "TSLA", "instrument_name": "Alphabet Inc.", "trade_date_time": "2023-05-26T15:30:00", "trade_details": {"buySellIndicator": "Sell", "price": 245.55581232603765, "quantity": 37, "trader": "John Doe"}}, {"trade_id": "5", "asset_class": ["Options"], "counterparty": "JKL Enterprises", "instrument_id": "AMZN", "instrument_name": "Apple Inc.", "trade_date_time": "2023-05-26T14:00:00", "trade_details": {"buySellIndicator": "Sell", "price": 245.55581232603765, "quantity": 37, "trader": "Colby Covington"}}, {"trade_id": "6", "asset_class": ["Commodities"], "counterparty": "GHI Company", "instrument_id": "AAPL", "instrument_name": "Apple Inc.", "trade_date_time": "2023-05-26T10:00:00", "trade_details": {"buySellIndicator": "Sell", "price": 245.55581232603765, "quantity": 37, "trader": "Colby Covington"}}, {"trade_id": "7", "asset_class": ["Options"], "counterparty": "DEF
```

### 3. Get Trade by ID

- URL: `/trades/{trade_id}`
- Method: GET
- Description: Retrieves a specific trade record by its ID.
- Path Parameter:
- `trade_id`: The unique ID of the trade.
- Response: JSON data containing the trade record matching the provided ID.
- Output :



#### 4. Filter Trades Based on Specific Conditions.

- URL: */specificTrades*
- Method: GET
- Description: Retrieves trade records filtered based on specific conditions.
- Query Parameters (optional):
  - counterparty: Filter trades by the counterparty.
  - instrumentId: Filter trades by the instrument ID.
  - instrumentName: Filter trades by the instrument name.
  - trader: Filter trades by the trader name.
- Output :

GET /specificTrades Get Trades Specific

Parameters

Name	Description
counterparty string (query)	counterparty
instrumentId string (query)	GOOGL
instrumentName string (query)	instrumentName
trader string (query)	Suraj Yadav

Execute Clear

Responses

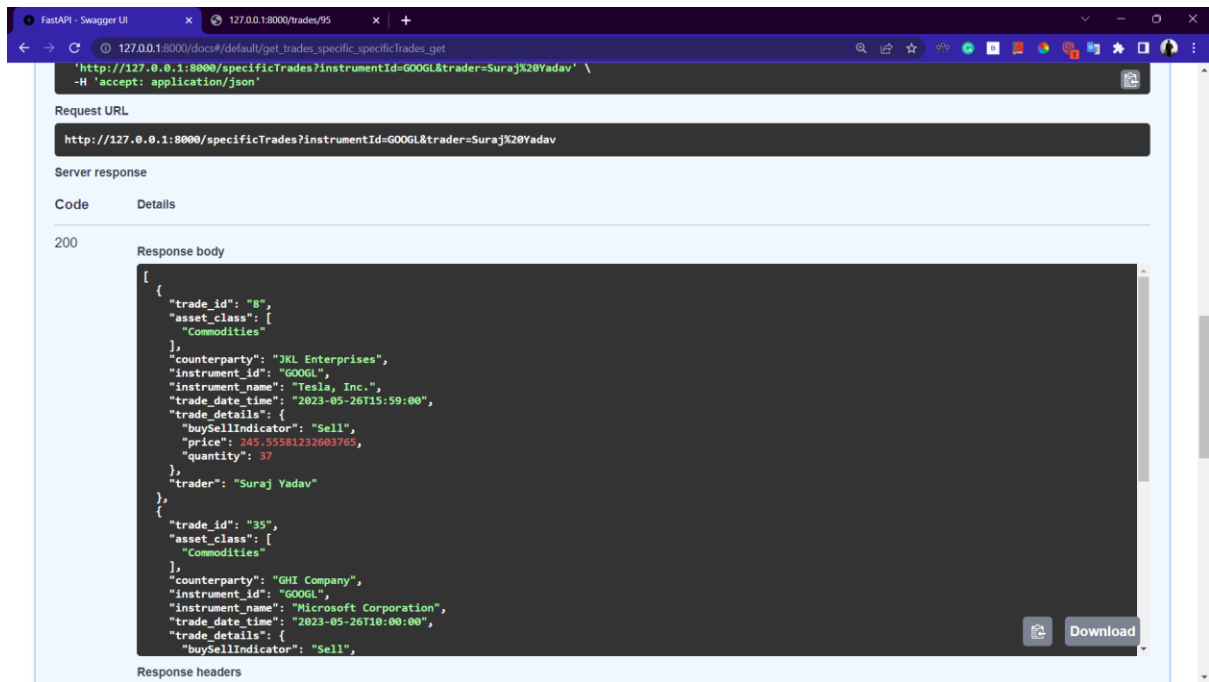
Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/specificTrades?instrumentId=GOOGL&trader=SurajYadav' \
  -H 'accept: application/json'
```

Request URL

http://127.0.0.1:8000/specificTrades?instrumentId=GOOGL&trader=SurajYadav

This takes trader name (Suraj Yadav) as input and instrument\_id(GOOGL) and gives results accordingly.



## 5. Advanced Search

- URL: */advancedSearch*
- Method: GET
- Description: Retrieves trade records based on advanced search criteria.
- Query Parameters (optional):
  - assetClass: Filter trades by the asset class.
  - start: Filter trades with a trade date and time greater than or equal to the provided start date and time.
  - end: Filter trades with a trade date and time less than or equal to the provided end date and time.
  - maxPrice: Filter trades with a price less than or equal to the provided maximum price.
  - minPrice: Filter trades with a price greater than or equal to the provided minimum price.
  - tradeType: Filter trades by the buy/sell indicator.



- Output :

FastAPI - Swagger UI | 127.0.0.1:8000/trades/95 | 127.0.0.1:8000/docs#/default/advanced\_search\_advancedSearch\_get

GET /advancedSearch Advanced Search

Parameters

Name	Description
assetClass	FX
start	start
end	end
maxPrice	maxPrice
minPrice	minPrice
tradeType	Sell

Execute Clear

Responses

FastAPI - Swagger UI | 127.0.0.1:8000/trades/95 | http://127.0.0.1:8000/advancedSearch?assetClass=FX&tradeType=Sell

Server response

Code Details

200

Response body

```
{
  "trade_id": "18",
  "asset_class": [
    "FX"
  ],
  "counterparty": "GHI Company",
  "instrument_id": "AMZN",
  "instrument_name": "Microsoft Corporation",
  "trade_date_time": "2023-05-26T15:30:00",
  "trade_details": {
    "buySellIndicator": "Sell",
    "price": 245.55581232603765,
    "quantity": 37
  },
  "trader": "Suraj Yadav"
},
{
  "trade_id": "26",
  "asset_class": [
    "FX"
  ],
  "counterparty": "JKL Enterprises",
  "instrument_id": "MSFI",
  "instrument_name": "Alphabet Inc.",
  "trade_date_time": "2023-05-26T14:00:00",
  "trade_details": {
    "buySellIndicator": "Sell"
  },
  "trader": "Suraj Yadav"
}
```

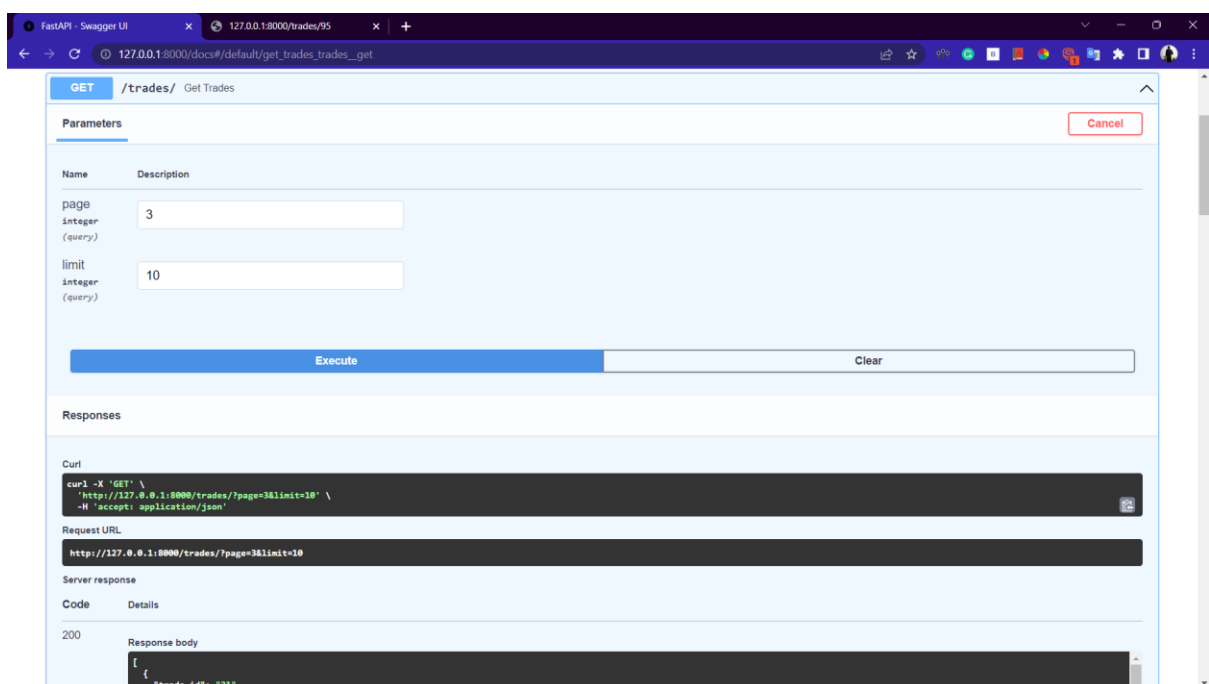
Response headers

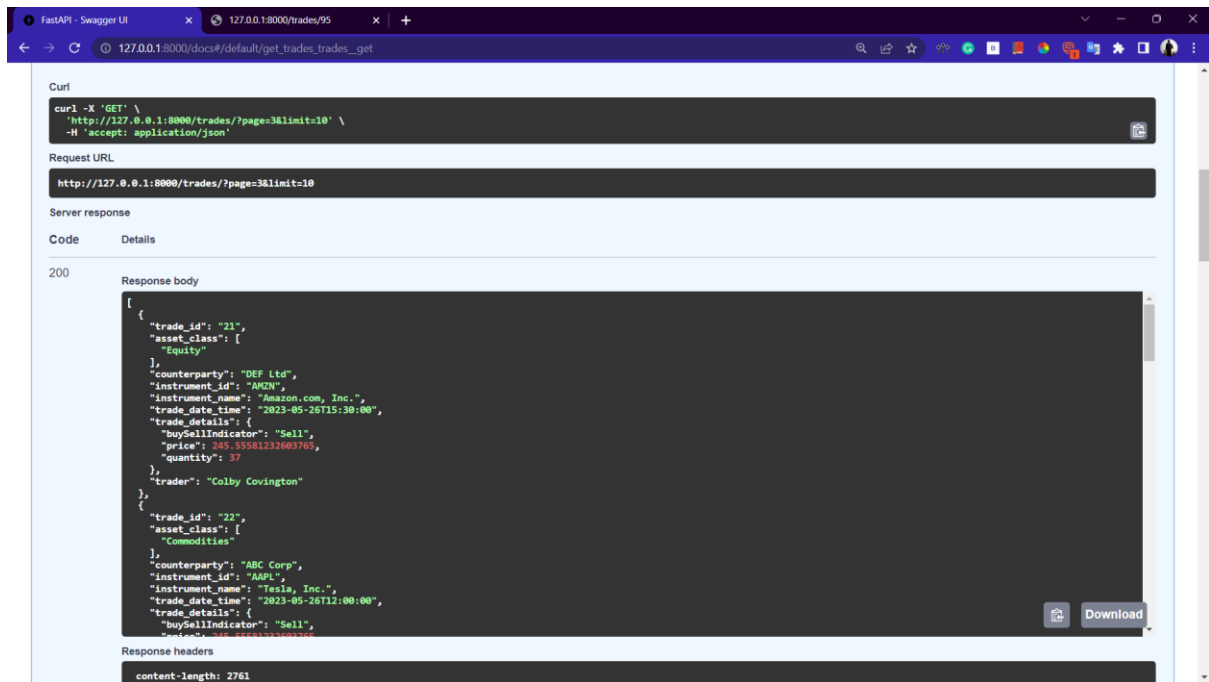
```
content-length: 5478
content-type: application/json
date: Wed, 07 Jun 2023 13:28:33 GMT
```

Download

## 6. Pagination.

- URL: */trades/*
- Method: GET
- Description: Retrieves a paginated list of trade records.
- Query Parameters (optional):
  - page: The page number to retrieve (default: 1).
  - limit: The maximum number of trade records to retrieve per page (default: 10).
- Response: JSON data containing a paginated list of trade records.
- Output :





*I hope this Solution meets all the requirements and I look forward for your reply.*

*Thank You!!*