




Portfolio API

 dhyey1706@gmail.com  [+91 9265449293](tel:+919265449293)  [Dhyey Vora](https://www.linkedin.com/in/DhyeyVora)

CONTENTS

Portfolio API.....	1
Schemas.....	2
Portfolio Schema.....	3
Stock Schema.....	3
Trade Schema:.....	3
API Endpoints.....	4
1. /api/portfolio :.....	4
2. /api/portfolio/trade :.....	7

Schemas

Three schemas in particular have been taken into account:

1. Portfolio Schema - For all portfolio related items
2. Stock Schema - For stock related properties
3. Trade Schema - For all types of trades of a particular stock

Portfolio Schema

➤ Properties:

- *_id* : Primary key for the each document of portfolio collection
 - *Username* : Owner of the portfolio
-

Stock Schema

➤ Properties:

- *_id* : Primary key for the each document of Stock collection
 - *Stockname* : Name of the Stock
 - *Quantity* : At any time states quantity of particular stock a owner owns
 - *portfolioId* : This works as reference between portfolio schema and Stock schema as it states particular stock belongs to which portfolio
-

Trade Schema:

➤ Properties:

- *_id* : Primary key for the each document of trade collection
- *Quantity* : Quantity of stocks involved in a particular trade
- *Stockid* : This works as reference between stock schema and trade schema , it states particular trade is related to which stock
- *Action* : States action done on the stock. Ex. Bought a stock or sold a stock
- *Price* : price at which stock is bought or sold

- *Timestamp* : shows timestamp at which document is created or updated

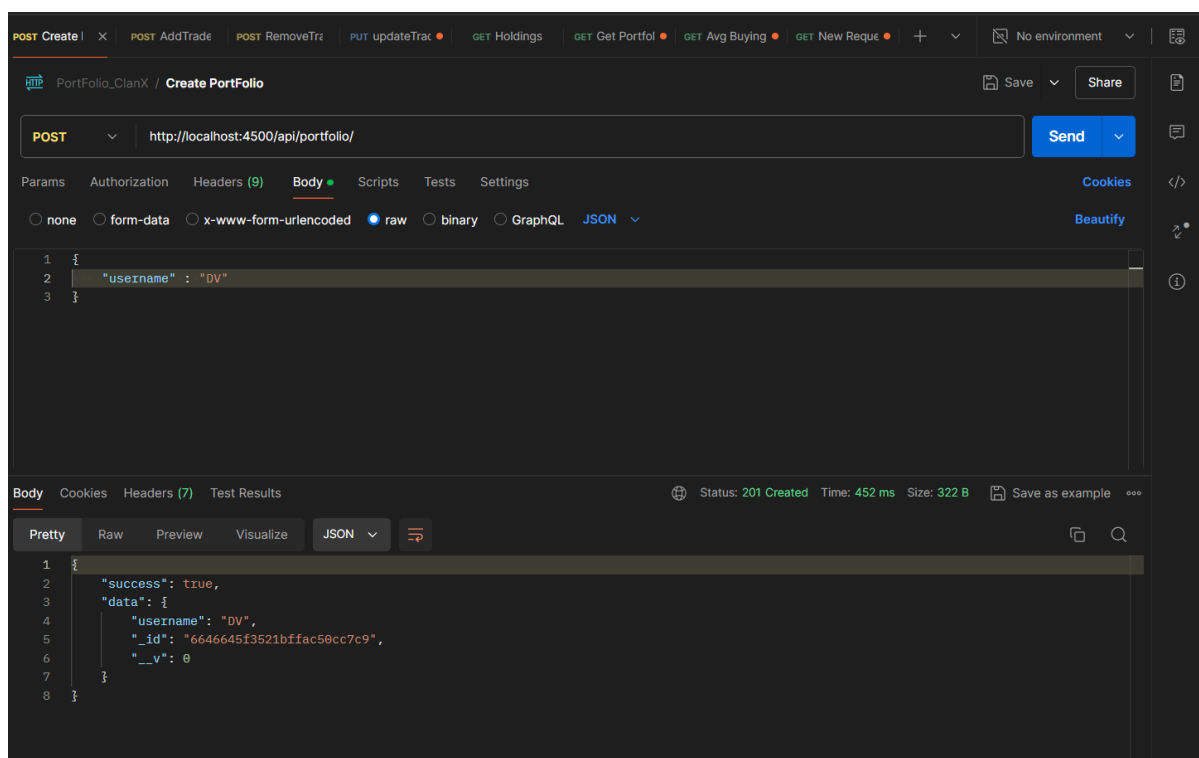
API Endpoints

1. /api/portfolio :

- Endpoint to get or save portfolio related items

➤ **POST** 👉 / : creates a portfolio

- Request parameters
 - Username : String (Request Body)
- Request parameters Format - JSON
- Response Format - JSON
- Response :



➤ **GET** 🖱️/:portfolioId : Provides information about whole portfolio including trades for stocks of portfolio

- Request parameters
 - PortfolioId : ObjectId() (Url Parameter)
- Request format : JSON
- Response format : JSON
- Response :

PortFolio_ClanX / **Get Portfolio** Save Share

GET ⌵ `http://localhost:4500/api/portfolio/664459df249ec2d1aeab43b1` Send ⌵

Params Authorization Headers (7) Body Scripts Tests Settings Cookies </>

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (7) Test Results ⊕ Status: 200 OK Time: 27 ms Size: 2.14 KB Save as example ⋮

Pretty Raw Preview Visualize **JSON** ⌵ 🔍

```
1 {
2   "success": true,
3   "data": [
4     {
5       "_id": "664459df249ec2d1aeab43b1",
6       "username": "Dhyey",
7       "__v": 0,
8       "stocks": [
9         {
10          "_id": "66451286570d7b2be9505e7e",
11          "stockname": "Reliance",
12          "quantity": 120,
13          "portfolioId": "664459df249ec2d1aeab43b1",
14          "__v": 0
15        },
16        {
17          "_id": "6645bbbf3c57478f665d0fdc",
18          "stockname": "Tesla",
19          "quantity": 530,
20          "portfolioId": "664459df249ec2d1aeab43b1",
```

➤ GET 👉 /:portfolioId :

Provides holdings of a particular portfolio

- Request Parameters : PortfolioId : ObjectId() (Url Parameter)
- Request Format : JSON
- Response Format : JSON
- Response :

The screenshot shows a REST client interface with a GET request to `http://localhost:4500/api/portfolio/holdings/664459df249ec2d1aeab43b1`. The response is a JSON object with the following structure:

```
1 {
2   "success": true,
3   "data": {
4     "owner": "Dhyey",
5     "stocks": [
6       {
7         "stockname": "Reliance",
8         "quantity": 120
9       },
10      {
11        "stockname": "Tesla",
12        "quantity": 530
13      }
14    ]
15  }
16 }
```

The response status is 200 OK, with a time of 122 ms and a size of 365 B. The response is displayed in the 'Body' tab, formatted as JSON.

2. [/api/portfolio/trade](#):

- contains routes to all the trade related functions and returns etc.

➤ POST 👉 /addTrade :

- Adds a trade(Buys a stock) for the given stock and given price and also modifies current stock number owned by the owner in stock collection
- Request Parameters
 - Stockname : String(Request Body)
 - portfolioId : ObjectId (Request Body)
 - Quantity : Number (Request Body)
 - Price : Number(Request Body)
 - Action : String(“BUY” or “SELL”) (Request Body)
- Request Format : JSON
- Response Format : JSON
- Response :

The screenshot displays a REST client interface for a POST request to `http://localhost:4500/api/portfolio/trade/addTrade`. The request body is a JSON object with the following fields: `stockname` (Tesla), `portfolioId` (664459df249ec2d1aeab43b1), `quantity` (50), `price` (60), and `action` (BUY). The response is a JSON object indicating success, with fields for `success` (true), `data` (containing quantity, stockid, action, price, _id, created_at, updated_at, and __v), and `__v` (0).

```
1 {
2   ... "stockname": "Tesla",
3   ... "portfolioId": "664459df249ec2d1aeab43b1",
4   ... "quantity": 50,
5   ... "price": 60,
6   ... "action": "BUY"
7 }
```

```
1 {
2   "success": true,
3   "data": {
4     "quantity": 50,
5     "stockid": "6645bbbf3c57478f665d0fdc",
6     "action": "BUY",
7     "price": 60,
8     "_id": "6645bbe03c57478f665d0fe2",
9     "created_at": "2024-05-16T07:55:12.095Z",
10    "updated_at": "2024-05-16T07:55:12.095Z",
11    "__v": 0
12  }
13 }
```

➤ **POST 👉 /removeTrade :**

- Removes a trade(sells a stock) for given stock and given price and also modifies current stock number owned by the owner in stock collection.
- Also checks for errors in trading like selling more stocks then the quantity
 - Request Parameters:
 - Stockname : String(Request Body)
 - portfolioId : ObjectId (Request Body)
 - Quantity : Number (Request Body)
 - Price : Number(Request Body)
 - Action : String(“BUY” or “SELL”) (Request Body)
 - Request Format : JSON
 - Response Format : JSON

- Response :

➤ *When there is an error in removing a trade*

The screenshot shows a REST client interface for a POST request to `http://localhost:4500/api/portfolio/trade/removeTrade`. The request body is a JSON object:

```
1 {
2   "stockname": "Tesla",
3   "portfolioId": "664459df249ec2d1aeab43b1",
4   "quantity": 610,
5   "price": 50,
6   "action": "SELL"
7 }
```

The response status is **400 Bad Request** with a time of 116 ms and size of 387 B. The response body is a JSON object:

```
1 {
2   "success": false,
3   "data": {
4     "error": "You can't sell more quantity of stocks than you have purchase , Please refer your holdings for each stock"
5   }
6 }
```

➤ *If trade is removed successfully without an error*

The screenshot shows a REST client interface for a POST request to `http://localhost:4500/api/portfolio/trade/removeTrade`. The request body is a JSON object:

```
1 {
2   "stockname": "Tesla",
3   "portfolioId": "664459df249ec2d1aeab43b1",
4   "quantity": 10,
5   "price": 50,
6   "action": "SELL"
7 }
```

The response status is **200 OK** with a time of 103 ms and size of 460 B. The response body is a JSON object:

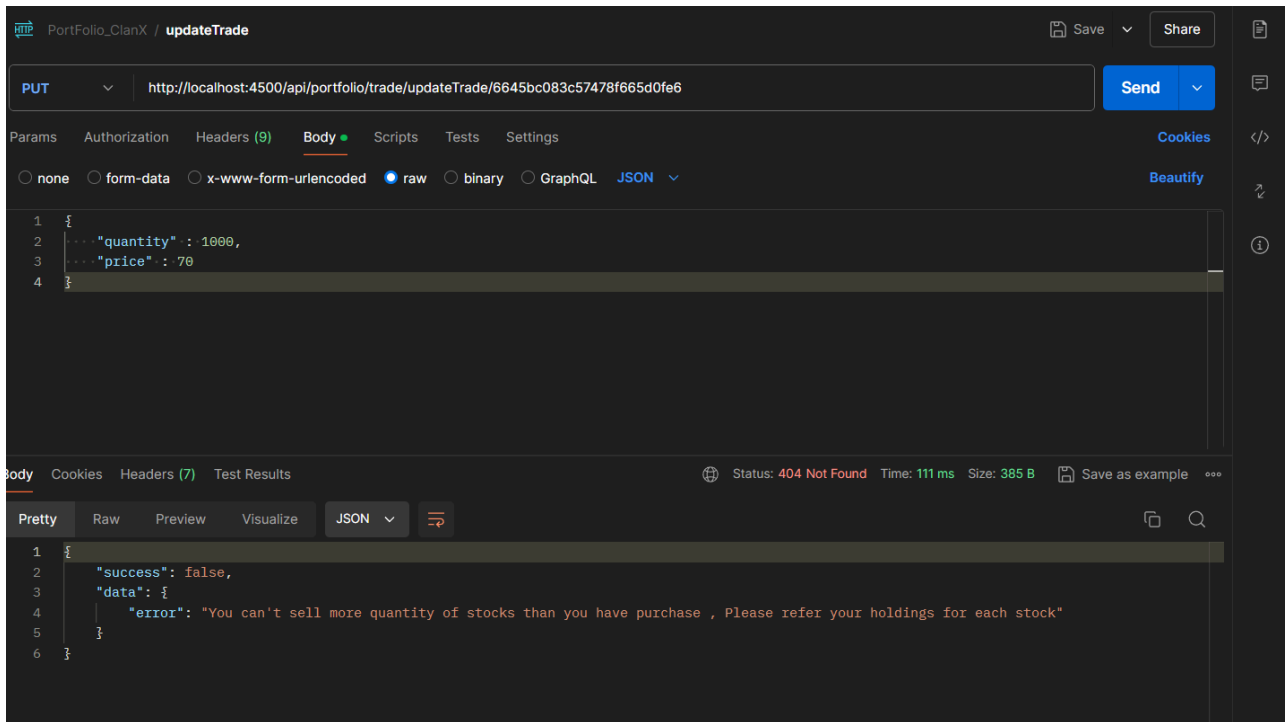
```
1 {
2   "success": true,
3   "data": {
4     "quantity": 10,
5     "stockid": "6645bbb3c57478f665d0fdc",
6     "action": "SELL",
7     "price": 50,
8     "_id": "66466c913521bfffac50cc7cd",
9     "created_at": "2024-05-16T20:29:05.196Z",
10    "updated_at": "2024-05-16T20:29:05.196Z",
11    "__v": 0
12  }
13 }
```

➤ **PUT** 🖱️ **/updateTrade/:tradeId**

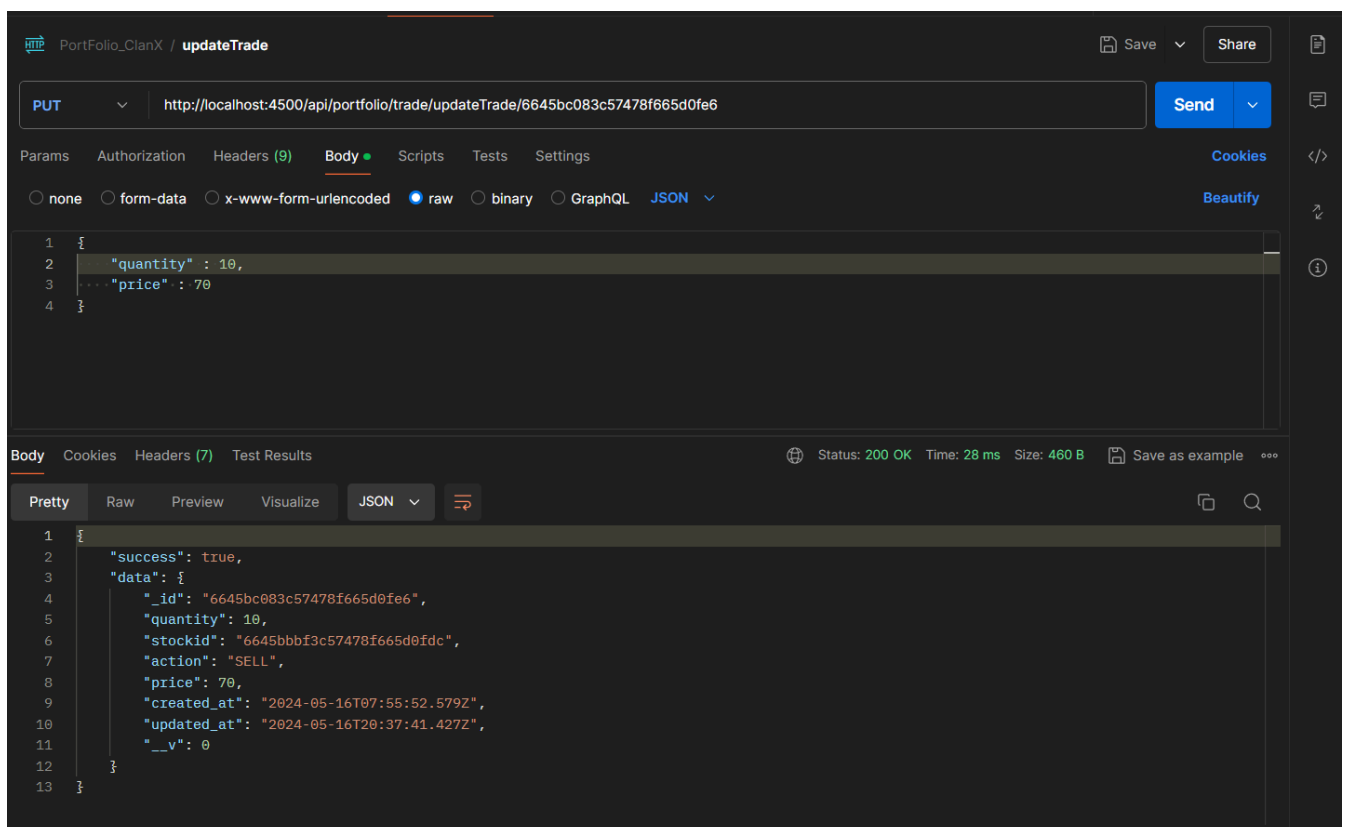
- Updates price or quantity for a particular trade and changes holding stock amount accordingly. Practically we can only edit two given properties in a trade after it has taken place.
- Throws an error in case of any invalid operation.
 - Request Parameters:
 - Quantity : Number(Request body)
 - Price : Number (Request Body)
 - tradeId : ObjectId (URL Parameter)
 - Request Format : JSON
 - Response Format : JSON

- Response :

1. *When there is an invalid edit performed in the trade , it will throw an error*



2. *When trade edit is successful, it'll respond with updated Trade information*



➤ GET 👉 /averageBuyingPrice

- Provides average buying price of each stock in a portfolio
- Request Parameters : NA
- Request Format : JSON
- Response Format : JSON
- Response :

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:4500/api/portfolio/trade/averageBuyingPrice`
- Method:** GET
- Response Status:** 200 OK, Time: 80 ms, Size: 346 B
- Response Body (JSON):**

```
1 {
2   "success": true,
3   "data": [
4     {
5       "stockname": "Tesla",
6       "averagePrice": 60
7     },
8     {
9       "stockname": "Reliance",
10      "averagePrice": 47.5
11    }
12  ]
13 }
```

Key	Value	Description
Key	Value	Description

➤ **GET** 👉 /returns :

- Provides cumulative returns of a portfolio considering final price as 100 and initial price as average buying price for that stock (This is logic provided as per company document and same query answered over email)
 - Request Parameters : NA
 - Request Format : JSON
 - Response Format : JSON
-
- Response :

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:4500/api/portfolio/trade/returns`
- Method:** `GET`
- Status:** `200 OK`, **Time:** `33 ms`, **Size:** `264 B`
- Response Body (JSON):**

```
{
  "success": true,
  "data": 30640
}
```

Key	Value	Description
Key	Value	Description