Documentation

Problem Statement :

Develop APIs for tracking a portfolio. APIs should include operations such as adding, updating, deleting trade as well as fetching portfolio, holdings and returns.

GitHub Link: https://github.com/agrawaltejas01/portfolio-tracking-api

Swagger UI Link: http://15.207.204.10:8000/api-docs/

Technology Stack :

1. Database: MongoDB
2. Server Side: Node JS
3. Framework: Express
4. ORM: Mongoose
5. Testing Tool: Postman
6. Version Control System: GitHub
7. Deployment Platform: AWS EC2

API Description :

1. **Add Trade :**

   API URL: **http://15.207.204.10:8000/trade/**

   Method: POST

   Body:
   ```
   {
           ticker : “string”,
           action : “number”,
           quantity : “integer”,
           price : “number”
   }.
   ```

   Response **:**

   a. Failed validations :

      Status: 400

      Body: “Invalid Body Received (Reason)”

   b. Operation successful :

      Status : 200

      ```
      Body : {
          ticker: “string",
          tradeId: "string",
          action: “number” ,
          quantity: “integer”,
          price: “integer”,
      }
      ```

2. **Update Trade**

API URL: **http://15.207.204.10:8000/trade/:tradeId**

Method: PATCH

Body: {

action : “number”,

quantity : “integer”,

price : “number”

}.

Response :

a. Failed Validations

Status : 400

Body : “Invalid Body Received (Reason)”

b. Trade No Found

Status : 404

Body :No trade was found (Perhaps wrong tradeId)

c. Operation successful

Status : 200

```
Body : {
    ticker : “string”,
    tradeId: "string" ,
    action : “number” ,
    quantity : “integer”,
   price : “number”
}
```

3. **Delete Trade**

API URL: **http://15.207.204.10:8000/trade/:tradeId**

Method: DELETE

Body: {}

Response :

a. Failed Validations

Status : 400

Body : “Invalid Body Received (Reason)”

b. Trade No Found

Status : 404

Body :No trade was found (Perhaps wrong tradeId)

c. Operation successful

Status : 200

```
Body : {
    ticker : “string” ,
    tradeId: "string"
}
```

4. **Fetch Portfolio**

   API URL : **http://15.207.204.10:8000/portfolio/**

   Method : GET

   Body : {}

   Response :

   a. Failed Validations

      Status : 400

      Body : “Invalid Body Received (Reason)”

   b. Operation successful

      Status : 200

```
Body : {
    ticker : “string”
    trades : [
                {
                    tradeId: "string",
                    timeStamp : “ISO Date”,
                    action : “number” ,
                   quantity : “integer”,
                  price : “number”,
                }
          ]
}
```

5. **Fetch Holding**

   API URL: **http://15.207.204.10:8000/portfolio/holdings**

   Method: GET

   Body: {}

   Response :

   a. Failed Validations

      Status: 400

      Body: “Invalid Body Received (Reason)”

   b. Operation successful

      Status: 200,

```
Body: {
    ticker: “string”,
    averageBuyPrice : “number”,
   Shares : “number”
}
```

6. **Fetch Returns**

API URL : **http://15.207.204.10:8000/portfolio/returns**

Method: GET

Body: {}

Response :

a. Failed Validations

Status: 400

Body: “Invalid Body Received (Reason)”

b. Operation successful

```
Status: 200
Body: {
    totalReturns: “number”
}
```

Validations :

1. Request Body Validations
   a. Number of parameters in the body should match exactly
   b. Name of parameters should match exactly
   c. Data types of all parameters should be correct.
   d. tradeId should be a string of length 24 in hexaDecimal (Contains 0-9 and a-f).
   e. “action” can only be 0(buy) or 1(sell)
   f. “quantity” can only be a whole number. (Integer >= 0).
   g. “price” should always be greater than 0.
2. User cannot sell more shares than he owns.
3. tradeId used in an update or delete trade should exist in the database.
4. If a delete or update operation is making the total number of shares less than 0, do not allow the operation.

Test Scenario :

Sr No	Operation	Body	Expected Result	Actual Result	Verdict
1	Add Trade	ticker: “ABC” action: 1(buy) quantity: 10 price: 100	A trade with given details is generated. Total noOfShares = 10	A trade with given details generated. (Let trade id be 1)	Pass
2	Add Trade	ticker: “ABC” action : 0(sell) quantity: 20 price: 100	Trade should not be added as user owns 10 shares and trying to sell 20	No trade was inserted	Pass
3	Add Trade	ticker: “ABC” action : 0 quantity: 5 price: 100	A trade with given details is generated. Total noOfShars = 5	A trade with given details generated. (Let trade Id be 2)	Pass
4	Fetch Holdings		avgBuyPrice : 100 noOfshares : 5	avgBuyPrice : 100 noOfshares : 5	Pass
5	Update Trade	tradeId : 2 action : 1(sell) quantity : 10 price : 200	Trade with tradeId 2 is updated with given data. Total noOfShares : 20	Trade with tradeId 2 is updated with given data. noOfShares : 20	Pass
6	Fetch Returns		-1000 (current price is assumed to be 100)	-1000	Pass
7	Fetch Holdings		avgBuyPrice : 150 noOfShares: 20	avgBuyPrice: 150 noOfShares: 20	Pass
8	Delete Trade	tradeId : 1	Trade with tradeId 1 is no longer in database noOfShares : 10	Trade with tradeId 1 is no longer in database noOfShares : 10	Pass
9	Fetch Portfolio		ticker: ABC trade : tradeId 1 and details	ticker: ABC trade : tradeId 1 and details	Pass
10	Fetch Holdings		avgBuyPrice : 200 noOfShares : 10	avgBuyPrice: 200 noOfShares : 10	Pass
11	Update Trade	tradId : 2 action : 0 quantity: 10 price: 200	Should not be allowed, as noOfShares will be -10	Trade was not updated	Pass

12	Delete Trade	tradeId : 2	Security for “ABC” should be blank with noOfShares = 0	Trade was deleted	Pass
13	Fetch Returns		totalReturns = 0	totalReturns = 0	Pass

Deployment :

1. Hosted on Amazon EC2 Ubuntu instance
2. IP: http://15.207.204.10/
3. Port: 8000
4. Database hosted on port 27017

Future Works :

1. Using a framework for writing test cases
2. Design system for multiple users.
3. Implement a log generation library.