**API Documentation**

**Base URL**

http://localhost:5000

**Transactions Controller**

**1. Add Transaction**

**Endpoint:** /transactions/

**Method:** POST

**Description:** Adds a new transaction to the database.

**Request Body:**

{

"ticker": "AAPL",

"trans\_type": "buy",

"quantity": 2,

"price\_per\_unit": 150.0

}

**Responses:**

* **201 Created**

{

"trans\_id": 1,

"ticker": "AAPL",

"trans\_type": "buy",

"quantity": 2,

"price\_per\_unit": 150.0

}

* **400 Bad Request**

{

"message": "Missing data"

}

**2. Get Transaction by ID**

**Endpoint:** /transactions/id/<int:trans\_id>

**Method:** GET

**Description:** Retrieves a transaction by its ID.

**Parameters:**

* trans\_id: The ID of the transaction to retrieve.

**Responses:**

* **200 OK**

{

"trans\_id": 1,

"ticker": "AAPL",

"trans\_type": "buy",

"quantity": 2,

"price\_per\_unit": 150.0

}

* **404 Not Found**

{

"message": "Transaction not found"

}

**3. Get Transactions by ticker**

**Endpoint:** /transactions/ticker/<string:ticker>

**Method:** GET

**Description:** Retrieves a transaction by its ticker.

**Parameters:**

* ticker: The ticker of the transaction to retrieve.

**Responses:**

* **200 OK**

[

{

"trans\_id": 1,

"ticker": "AAPL",

"trans\_type": "buy",

"quantity": 10,

"price\_per\_unit": 150.0

},

{

"trans\_id": 2,

"ticker": "AAPL",

"trans\_type": "sell",

"quantity": 5,

"price\_per\_unit": 160.0

}

]

* **404 Not Found**

{

"message": "Transactions not found for the ticker"

}

**4. Get All Transactions**

**Endpoint:** /transactions/

**Method:** GET

**Description:** Retrieves all transactions.

**Responses:**

* **200 OK**

[

{

"trans\_id": 1,

"ticker": "AAPL",

"trans\_type": "buy",

"quantity": 2,

"price\_per\_unit": 150.0

},

{

"trans\_id": 2,

"ticker": "GOOGL",

"trans\_type": "sell",

"quantity": 1,

"price\_per\_unit": 1000.0

}

]

**5. Update Transaction**

**Endpoint:** /transactions/<int:trans\_id>

**Method:** PUT

**Description:** Updates an existing transaction.

**Parameters:**

* trans\_id: The ID of the transaction to update.

**Request Body:**

{

"ticker": "AAPL",

"trans\_type": "sell",

"quantity": 3,

"price\_per\_unit": 160.0

}

**Responses:**

* **200 OK**

{

"trans\_id": 1,

"ticker": "AAPL",

"trans\_type": "sell",

"quantity": 3,

"price\_per\_unit": 160.0

}

* **404 Not Found**

{

"message": "Transaction not found"

}

**6. Delete Transaction**

**Endpoint:** /transactions/<int:trans\_id>

**Method:** DELETE

**Description:** Deletes a transaction by its ID.

**Parameters:**

* trans\_id: The ID of the transaction to delete.

**Responses:**

* **200 OK**

{

"message": "Transaction deleted"

}

* **404 Not Found**

{

"message": "Transaction not found"

}

**Holdings Controller**

**1. Add Holding**

**Endpoint:** /holdings/

**Method:** POST

**Description:** Adds a new holding to the database.

**Request Body:**

{

"ticker": "AAPL",

"quantity": 5,

"price": 145.0

}

**Responses:**

* **201 Created**

{

"holding\_id": 1,

"ticker": "AAPL",

"quantity": 5,

"price": 145.0

}

* **400 Bad Request**

{

"message": "Missing data"

}

**2. Get Holding by ID**

**Endpoint:** /holdings/<int:holding\_id>

**Method:** GET

**Description:** Retrieves a holding by its ID.

**Parameters:**

* holding\_id: The ID of the holding to retrieve.

**Responses:**

* **200 OK**

{

"holding\_id": 1,

"ticker": "AAPL",

"quantity": 5,

"price": 145.0

}

* **404 Not Found**

{

"message": "Holding not found"

}

**3. Get All Holdings**

**Endpoint:** /holdings/

**Method:** GET

**Description:** Retrieves all holdings.

**Responses:**

* **200 OK**

[

{

"holding\_id": 1,

"ticker": "AAPL",

"quantity": 5,

"price": 145.0

},

{

"holding\_id": 2,

"ticker": "GOOGL",

"quantity": 3,

"price": 950.0

}

]

**4. Update Holding**

**Endpoint:** /holdings/<int:holding\_id>

**Method:** PUT

**Description:** Updates an existing holding.

**Parameters:**

* holding\_id: The ID of the holding to update.

**Request Body:**

{

"ticker": "AAPL",

"quantity": 7,

"price": 140.0

}

**Responses:**

* **200 OK**

{

"holding\_id": 1,

"ticker": "AAPL",

"quantity": 7,

"price": 140.0

}

* **404 Not Found**

{

"message": "Holding not found"

}

**5. Delete Holding**

**Endpoint:** /holdings/<int:holding\_id>

**Method:** DELETE

**Description:** Deletes a holding by its ID.

**Parameters:**

* holding\_id: The ID of the holding to delete.

**Responses:**

* **200 OK**

{

"message": "Holding deleted"

}

* **404 Not Found**

{

"message": "Holding not found"

}

**Actions Controller**

**1. Add Buy/Sell Transaction**

**Endpoint:** /actions/add\_buy\_sell

**Method:** POST

**Description:** Adds a buy or sell transaction and updates the holdings accordingly.

**Request Body:**

{

"ticker": "AAPL",

"trans\_type": "buy",

"quantity": 2,

"price\_per\_unit": 150.0

}

**Responses:**

* **201 Created**

{

"trans\_id": 1,

"ticker": "AAPL",

"trans\_type": "buy",

"quantity": 2,

"price\_per\_unit": 150.0

}

* **400 Bad Request**

{

"message": "Missing data"

}

* **400 Bad Request**

{

"message": "Insufficient quantity to sell"

}

* **404 Not Found**

{

"message": "Holding not found"

}

* **400 Bad Request**

{

"message": "Invalid transaction type"

}

**2. Get Historical Data**

**Endpoint:** /actions/historical\_data

**Method:** GET

**Description:** Retrieves historical data for a given ticker between specified start and end dates.

**Parameters:**

* ticker: The stock ticker.
* start\_date: The start date for the historical data (YYYY-MM-DD).
* end\_date: The end date for the historical data (YYYY-MM-DD).

**Responses:**

* **200 OK**

[

{

"Date": "2023-01-01",

"Open": 150.0,

"High": 155.0,

"Low": 145.0,

"Close": 152.0,

"Volume": 1000000

},

{

"Date": "2023-01-02",

"Open": 152.0,

"High": 157.0,

"Low": 150.0,

"Close": 155.0,

"Volume": 1200000

}

]

* **400 Bad Request**

{

"error": "Please provide ticker, start\_date, and end\_date"

}

* **500 Internal Server Error**

{

"error": "An error occurred while fetching historical data"

}

**3. Get Ticker Data**

**Endpoint:** /actions/get\_ticker

**Method:** GET

**Description:** Retrieves a given ticker and can be used to validate and retrieve info about the ticker.

**Parameters:**

* ticker: The stock ticker.

**Responses:**

* **200 OK**

{

"ticker": "AAPL",

"valid": true,

"info": {

"symbol": "AAPL",

"longName": "Apple Inc.",

"regularMarketPrice": 175.96,

"marketCap": 2786840000000

// other relevant stock information

}

}

* **400 Bad Request**

{

"ticker": "INVALID",

"valid": false

}

* **Error response body**

{

"error": "Please provide a ticker symbol"

}

**4. Calculate Net Worth**

* **Endpoint**: actions/networth
* **Method**: GET
* **Description**: Calculate the net worth of a user by summing up the value of all their holdings. The value of each holding is determined by multiplying the quantity with the current price of the stock.
* **Response**:
  + **200 OK**: Successfully calculated net worth.

{

"networth": 123456.78

}

* + **200 OK (No holdings)**: No holdings found.

{

"networth": 0,

"message": "No holdings found"

}

* + **500 Internal Server Error**: An error occurred while processing the request.

{

"error": "Error message"

}

**5. Calculate Profit and Loss**

* **Endpoint**: actions/profit\_loss
* **Method**: GET
* **Description**: Calculate the total profit or loss of a user by comparing the current price of stocks with their purchase price. The profit/loss for each holding is determined by multiplying the difference between the current price and the purchase price with the quantity.
* **Response**:
  + **200 OK**: Successfully calculated profit and loss.

{

"profit\_loss": 1234.56

}

* + **200 OK (No holdings)**: No holdings found.

{

"profit\_loss": 0,

"message": "No holdings found"

}

* + **500 Internal Server Error**: An error occurred while processing the request.

{

"error": "Error message"

}

**6. Search Ticker**

* **Endpoint**: actions/search\_ticker
* **Method**: GET
* **Description**: Search for ticker symbols and their corresponding names based on a query string. The search results are ordered by relevance, with matches at the start of the name given higher priority.
* **Parameters**:
  + **query** (string, required): The search query string.
* **Response**:
  + **200 OK**: Successfully retrieved search results.

[

{

"name": "Company Name 1",

"ticker": "Ticker1"

},

{

"name": "Company Name 2",

"ticker": "Ticker2"

}

//... up to 10 results

]

* + **400 Bad Request**: Missing search query.

{

"error": "Please provide a search query"

}

* + **500 Internal Server Error**: An error occurred while processing the request.

{

"error": "Error message"

}

**Watchlist Controller**

**1. Add to Watchlist**

* **Endpoint**: watchlist/
* **Method**: POST
* **Description**: Add a ticker symbol to the user's watchlist.
* **Request Body**:
  + **ticker** (string, required): The ticker symbol to be added to the watchlist.
* **Response**:
  + **201 Created**: Successfully added the ticker to the watchlist.

{

"ticker": "AAPL"

}

* + **400 Bad Request**: Missing ticker symbol in the request body.

{

"error": "Please provide a ticker symbol"

}

* + **500 Internal Server Error**: An error occurred while processing the request.

{

"error": "Error message"

}

**2. Get All Watchlist**

* **Endpoint**: watchlist/
* **Method**: GET
* **Description**: Retrieve all ticker symbols in the user's watchlist.
* **Response**:
  + **200 OK**: Successfully retrieved all watchlist entries.

[

{

"ticker": "AAPL"

},

{

"ticker": "GOOGL"

}

//... other entries

]

* + **500 Internal Server Error**: An error occurred while processing the request.

{

"error": "Error message"

}

**3. Remove from Watchlist**

* **Endpoint**: watchlist/<string:ticker>
* **Method**: DELETE
* **Description**: Remove a ticker symbol from the user's watchlist.
* **Parameters**:
  + **ticker** (string, required): The ticker symbol to be removed from the watchlist.
* **Response**:
  + **200 OK**: Successfully removed the ticker from the watchlist.

{

"ticker": "AAPL"

}

* + **404 Not Found**: The ticker symbol was not found in the watchlist.

{

"message": "Watchlist entry not found"

}

* + **500 Internal Server Error**: An error occurred while processing the request.

{

"error": "Error message"

}