Developed By

C SUMANTH

Email : sumanthchinna047@gmail.com

Phone : 9014991769

GitHub Link: https://github.com/SUMANTHCHINNA/Real_Time_Dynamic_Stock_Portfolio_Tracker.git

Project Functionality Link : https://youtu.be/bvZWZoFKb7k?si=C_vvADGH2Sp6cMsm

---

Stock Portfolio Tracker

The **Stock Portfolio Tracker** is a web application that allows users to manage their stock investments effectively. It provides features like user registration and authentication, adding and updating stocks in a portfolio, viewing real-time stock prices, and calculating key portfolio metrics such as total value, top-performing stocks, and portfolio distribution.

Features

Backend Features:

1. **User Management:**
    - User registration with username, email, and password.
    - Secure login with JWT-based authentication.

2. **Stock Management:**
    - Add, update, and delete stocks in the user's portfolio.
    - View all stocks in the portfolio.

3. **Portfolio Metrics:**
    - Real-time portfolio metrics including total value, top-performing stock, and portfolio distribution.
    - Fetch real-time stock prices from external APIs.

Frontend Features:

1. User-friendly interface for managing portfolio.
2. Forms for adding, updating, and deleting stocks.
3. Interactive dashboard displaying real-time metrics like total value, top-performing stock, and portfolio distribution.
4. Authentication system for secure login and registration.

1. User Authentication (Register & Login)

Register Page

- Allows new users to register by providing a username, email, and password.
- Sends the registration data to the backend to create a new user.
- After successful registration, stores the JWT token and redirects to the **Dashboard**.

Login Page

- Allows users to log in using their credentials(email and password).
- Upon successful login, stores the JWT token in `localStorage` for authentication.
- Redirects to the **Dashboard** after successful login.

2. Dashboard page

- Displays an overview of the user's portfolio, including:
  - Total portfolio value.
  - Top-performing stocks.
  - Portfolio distribution.
- Accessible only for authenticated users.
- Shows a summary of current stocks, investments, and key metrics such as total value,Top-performing stocks and Portfolio distribution.

3. Stock Management (My Stocks)

MyStocks Page

- Displays a list of stocks owned by the user.
- Each stock entry shows:
  - Stock name and symbol.
  - Quantity owned.

    - Buy price of the stock.
- Actions for each stock:
    - **Edit**: Opens a modal to edit the stock details (e.g., quantity, buy price).
    - **Delete**: Deletes the stock from the portfolio.

Modal for Editing Stock

- Allows users to edit stock information (name, quantity, buy price).
- Sends the updated data to the backend via the `PATCH` request to update the stock details.

4. Add New Stock

AddStock Page

- Allows users to add a new stock to their portfolio by entering:
    - Stock name.
    - symbol.
    - Quantity.
    - Buy price.
- Sends the data to the backend via the `POST` request to add the stock to the database.

5. Stock Prices

StockPrices Page

- Displays real-time stock prices for a list of stocks fetched from an external API (e.g., Alpha Vantage, Yahoo Finance).
- Each stock displays:
    - Symbol.
    - Date and time of the price update.
    - Open price, high price, low price, and close price.
    - Volume traded.
- Allows users to check the latest prices for stocks in their portfolio.

6. Sidebar Navigation

Sidebar Component

- Provides navigation links to the various pages in the application:

    - Dashboard
    - MyStocks
    - StockPrices
    - LogOut
- Allows users to log out by removing the token from `localStorage` and redirecting them to the login page.

API Endpoints

User Management APIs

1. **POST** `/api/user/register`
    - new user to register by providing username, email and password.
      Request Body:

```
{
      "username": "user",
      "email": "user@example.com",
      "password": "password123"
}
```

Response:

- **Success (200 OK)**

```
{
      "status": "true",
      "message": "Registered successfully",
      "token": "jwt_token"
}
```

- **Error (400 Bad Request)**

```
{
      "status": "false",
      "message": "User already exist please login"
}
```

2. **POST** `/api/user/login`
    - Logs in a user and provides a JWT token for authentication in future requests.

```
{
    "email": "user@example.com",
    "password": "password123"
}
```

Response:

- **Success (200 OK)**

```
{
     "status": "true",
     "message": "Login successful",
     "token": "jwt_token"
}
```

- **Error (404 Unauthorized)**

```
{
     "status": "false",
     "message": "Unauthorized access"
}
```

- **Error (400 Bad Request)**

```
{
     "status": "false",
     "message": "All fields must be filled"
}
```

Stock Management APIs

3. **POST** `/api/stocks/add`
    - Adds a new stock to the user's portfolio.

```
{
    "stockName": "Apple",
    "symbol": "AAPL",
    "quantity": 5,
    "buyPrice": 150
}
```

Response:

- **Success (200 OK)**

```
{
     "status": "true",
     "message": "Stock added successfully"
}
```

- **Error (400 Bad Request)**

```
{
     "status": "false",
     "message": "Fill all details"
}
```

- **Error (404 Unauthorized)**

```
{
     "status": "false",
     "message": "Unauthorized access"
}
```

4. **GET** `/api/stocks/myStocks`
    - Retrieves all stocks in the user's portfolio.

Response:

- **Success (200 OK)**

```
{
  "status": true,
  "message": [
    {
      "_id": "stock_id_1",
      "stockName": "APPLE",
      "symbol": "AAPL",
      "quantity": 10,
      "buyPrice": 150.00
    }
  ]
}
```

- **Error (404 Unauthorized)**

```
{
     "status": "false",
     "message": "Unauthorized access"
}
```

5. **PATCH** `/api/stocks/update/:stockId`
    - Updates the details (quantity or buy price) of an existing stock.

Response:

- **Success (200 OK)**

```
{
      "status": "true",
      "message": "Stock updated successfully"
}
```

- **Error (404 Unauthorized)**

```
{
      "status": "false",
      "message": "Unauthorized access"
}
```

6. **DELETE** `/api/stocks/delete/:stockId`
    - Deletes a stock from the user's portfolio.

Response:

- **Success (200 OK)**

```
{
      "status": "true",
      "message": "Stock deleted successfully"
}
```

- **Error (404 Unauthorized)**

```
{
      "status": "false",
      "message": "Unauthorized access"
}
```

Portfolio Metrics APIs

7. **GET** `/api/stock/metrics`
    - Calculates and returns key metrics of the user's portfolio such as total portfolio value, top-performing stock, and portfolio distribution based on real-time stock prices.

Response:

- **Success (200 OK)**

```
{
  "status": true,
  "message": [{
    "totalPortfolioValue": 9060,
    "topPerformingStock": {
        "stockName": "Amazon",
        "symbol": "AMZN",
        "value": 6100
    },
    "portfolioDistribution": [
        { "stockName": "Apple", "symbol": "AAPL", "percentage": 8.83 },
        { "stockName": "Tesla", "symbol": "TSLA", "percentage": 23.84 },
        { "stockName": "Amazon", "symbol": "AMZN", "percentage": 67.33 }
    ]
  }]
}
```

- **Error (404 - Unauthorized)**

```
{
     "status": "false",
     "message": "User not found"
}
```

8. **GET** `/api/stock/showPrices`
    - Fetches real-time stock prices for a list of symbols in bulk.

Response:

- **Success (200 OK)**

```
{
  "status": true,
  "message": [
    {
      "symbol": "AAPL",
      "open": 150.00,
      "high": 155.00,
      "low": 148.00,
      "close": 152.00,
      "volume": 100000
    }
  ]
}
```

- **Error (500)**

```
{
      "status": "false",
      "message": "Failed to fetch stock data"
}
```

Calculation Metrics

```
### Total Portfolio Value:
# Formula:
1. Total Value = ∑ ( Quantity × Real-Time Price )

### Top-Performing Stock:
# Formula:
2. Top Stock = max ( Quantity × Real-Time Price )

### Portfolio Distribution:

# Formula :
3. Distribution (%) = ( Stock Value / Total Portfolio Value ) × 100
```

Sample Calculation

Sample Stock Data:

```
[
  {
    "stockName": "Apple",
    "symbol": "AAPL",
    "quantity": 5,
    "buyPrice": 150
  },
  {
    "stockName": "Tesla",
    "symbol": "TSLA",
    "quantity": 3,
    "buyPrice": 700
  },
  {
    "stockName": "Amazon",
    "symbol": "AMZN",
    "quantity": 2,
    "buyPrice": 3000
  }
]
```

Sample Real-Time Stock Prices

```
{
  "AAPL": 160,
  "TSLA": 720,
  "AMZN": 3050
}
```

Total Portfolio Value

```
# Formula:
1. Total Value = ∑ (Quantity × Real-Time Price)

# Calculation:
- Apple = 5 * 160 = 800
- Tesla = 3 * 720 = 2160
- Amazon = 2 * 3050 = 6100

- Total Value = 800 + 2160 + 6100 = 9060
```

Top-Performing Stock

```
# Formula:
2. Top Stock = max ( Quantity × Real-Time Price )

# Calculation:
- Apple: 800
- Tesla: 2160
- Amazon: 6100

- Top-Performing Stock: Amazon with value: 6100
```

Portfolio Distribution

```
# Formula:
3. Distribution (%) = (Stock Value ÷ Total Portfolio Value) × 100

# Calculation:
- Apple: (800 / 9060) * 100 = 8.83%
- Tesla: (2160 / 9060) * 100 = 23.84%
- Amazon: (6100 / 9060) * 100 = 67.33%
```

Send Response to Frontend

```
# Portfolio Metrics:
========================
- Total Portfolio Value: $9060

- Top-Performing Stock: Amazon (AMZN) - $6100
```

```
# Portfolio Distribution:
- Apple (AAPL): 8.83%
- Tesla (TSLA): 23.84%
- Amazon (AMZN): 67.33%
========================
```

Frontend Technologies

1. **React.js**: For creating the user interface.
2. **CSS**: For styling the frontend.
3. **Axios**: For making API requests.

Backend Technologies

1. **Node.js**: Backend runtime environment.
2. **Express.js**: Web framework for APIs.
3. **MongoDB**: Database to store user and stock information.
4. **JWT**: For secure user authentication.
5. **bcryptjs**: For password hashing.
6. **dotenv**: For managing environment variables.
7. **Twelve Data API**: For fetching real-time stock prices.
8. **CORS**: Middleware for enabling cross-origin requests between frontend and backend.

Limitations

1. Limited API requests per day due to using external stock price APIs from TwelveData.com (Not premium).
2. Dependent on the availability of the Twelve Data API for real-time stock prices.

Installation and Setup

Backend Part

Install Dependencies

```
cd backend
npm install
```

Set up environment variables in a .env file

```
PORT=1324
KEY="a Simple Portfolio Tracker"
APIKEY="db523456236347c5a0d65bc5d37b6dad"
MONGODB="mongodb://localhost:27017/stock?directConnection=true"
```

Start the server

```
npm start
```

Frontend part

Navigate to the frontend directory

```
cd frontend
```

Install dependencies

```
npm install
```

Start the application

```
npm start
```

API Key from TwelveData.com

- **Twelve Data API Key :** db523456236347c5a0d65bc5d37b6dad

- Example API - Sample API to Fetch Data from External API from TwelveData.com

1. It fetches data that is updated minute by minute.
   Sample API - Link

Backend Structure

```
backend/
├── controllers/
│   ├── stock.js
│   └── user.js
├── middleware/
│   └── auth.js
├── models/
│   ├── index.js
│   ├── myStock.json
│   ├── presentStock.json
│   ├── stock.js
```

```
│   ├── stockPrice.js
│   └── user.js
├── plan/
│   └── flowchart.png
├── routes/
│   ├── stock.js
│   └── user.js
├── utils/
│   └── index.js
├── .env
├── index.js
└── package.json
```

Frontend Structure

```
frontend/
├── pages/
│   ├── AddStock.jsx
│   ├── Dashboard.jsx
│   ├── Login.jsx
│   ├── Register.jsx
│   ├── Sidebar.jsx
│   ├── StockPrices.jsx
│   └── MyStocks.jsx
├── src/
│   ├── App.css
│   ├── App.jsx
│   └── main.jsx
├── index.html
├── package.json
└── package-lock.json
```

Sample Companies

Technology Companies:

1. GOOGL - Alphabet Inc. (Google)
2. MSFT - Microsoft Corporation
3. TSLA - Tesla, Inc.
4. META - Meta Platforms, Inc. (Facebook)
5. NFLX - Netflix, Inc.
6. NVDA - NVIDIA Corporation

Consumer Goods:

7. PG - Procter & Gamble Co.
8. KO - Coca-Cola Company
9. PEP - PepsiCo, Inc.
10. WMT - Walmart Inc.

Financials:

11. JPM - JPMorgan Chase & Co.
12. BAC - Bank of America Corporation
13. C - Citigroup Inc.
14. GS - Goldman Sachs Group, Inc.

Energy:

15. XOM - Exxon Mobil Corporation
16. CVX - Chevron Corporation
17. SLB - Schlumberger Limited

Healthcare:

18. JNJ - Johnson & Johnson
19. PFE - Pfizer Inc.
20. MRK - Merck & Co., Inc.

Industrials:

21. BA - Boeing Company
22. CAT - Caterpillar Inc.
23. GE - General Electric Company

Miscellaneous:

24. AMAT - Applied Materials, Inc.
25. ADBE - Adobe Inc.
26. INTC - Intel Corporation
