



REAL TIME STOCK MARKET TRACKER

MAMTHA SRI (RA2211056010161)
DHEEPAK(RA2211056010159)



TABLE OF CONTENTS

<u>TITLE</u>	<u>PAGE.NO.</u>
OBJECTIVE	3
ABSTRACT	4
METHODOLOGY	5-6
IMPLEMENTATION-CODE	7-10
RESULT-OUTPUT	11
CONCLUSION	12



OBJECTIVE

The objective of a real-time stock market tracker is to provide users with up-to-the-minute information and data about the financial markets. This type of software or service is typically used by investors, traders, financial professionals, and anyone interested in monitoring stock prices and related financial information.



ABSTRACT

The Real-Time Stock Market Tracker Application is a software solution designed to provide users with instantaneous access to critical financial market data. In today's fast-paced investment landscape, having up-to-the-minute information is essential for making informed decisions.

The primary objective of this project is to develop a real-time stock market tracker application that caters to the needs of investors, traders, and financial professionals.

METHODOLOGY

DATA STRUCTURE USED:

In the provided C++ code for the stock market tracker, several data structures are used. Here are the main data structures and their roles in the code:

1. **Stack (`std::stack<Stock>`):** The primary data structure in this code is a stack of `Stock` objects. It is declared as a global variable named `stockStack`. This stack is used to store and manage the stock data. Stocks are pushed onto the stack when added and popped from the stack when deleted. The stack follows the Last-In, First-Out (LIFO) order, which is suitable for tracking stock data.
2. **Struct (`Stock`):** The `Stock` struct is used to define the structure of individual stock records. It consists of two data members: `symbol` (a string to store the stock symbol) and `price` (a double to store the stock price). This struct is the basis for creating stock objects that are pushed onto the stack.
3. **Temporary Stack (`std::stack<Stock> tempStack`):** Temporary stacks are used in various functions for specific purposes. For example, when searching or deleting a stock, a temporary stack is created to hold elements while searching for the target stock or moving stocks around.

These data structures are used to manage and manipulate stock data within the program. The primary data structure, the stack, is used for storing and organizing stock records, while temporary stacks are used to assist in searching, deleting, and clearing the main stack as needed.



ALGORITHMS USED

- 1. Add Stock**
 - 2. Search Stock**
 - 3. Delete Stock**
 - 4. Create New Stack**
 - 5. Display Stocks (Stack)**
- 



IMPLEMENTATION-CODE


```
#include <iostream>
#include <string>
#include <stack>
#include <iomanip> // for fixed and setprecision

using namespace std;

struct Stock {
    string symbol;
    double price;
};

stack<Stock> stockStack;

// Function to add a stock to the stack
void addStock() {
    string symbol;
```




```
double price;
```

```
    cout << "Enter stock symbol: ";
```

```
    cin >> symbol;
```

```
    cout << "Enter stock price: ";
```

```
    cin >> price;
```

```
Stock newStock;
```

```
    newStock.symbol = symbol;
```

```
    newStock.price = price;
```

```
    stockStack.push(newStock);
```

```
// Set the output format to fixed with 2 decimal places
```

```
    cout << fixed << setprecision(2);
```

```
    cout << "Stock added to the stack: " << symbol << " -
```

```
Price: " << newStock.price << endl;
```

```
}
```

```
// Function to search for a stock by symbol
```

```
void searchStock() {
```

```
    string symbol;
```

```
cout << "Enter stock symbol to search: ";
cin >> symbol;

stack<Stock> tempStack = stockStack;

while (!tempStack.empty()) {
    Stock stock = tempStack.top();
    if (stock.symbol == symbol) {

cout << "Found " << symbol << " with a price of " << fixed
<< setprecision(2) << stock.price << endl;
        return;
    }
    tempStack.pop();
}

cout << symbol << " not found." << endl;
}

// Function to delete a stock by symbol
void deleteStock() {
    string symbol;
    cout << "Enter stock symbol to delete: ";
```

```
cin >> symbol;
    stack<Stock> tempStack;
    bool found = false;

    while (!stockStack.empty()) {
        Stock stock = stockStack.top();
        if (stock.symbol == symbol) {
            stockStack.pop();
            cout << "Stock deleted: " << symbol << endl;
            found = true;
        } else {
            tempStack.push(stock);
            stockStack.pop();
        }
    }

    while (!tempStack.empty()) {
        stockStack.push(tempStack.top());
        tempStack.pop();
    }

    if (!found) {
        cout << "No such stock available" << endl;
```

```
}
}
void createStack() {
    while (!stockStack.empty()) {
        stockStack.pop();
    }
    cout << "Stack created (cleared)." << endl;
}


void displayStocks() {
    if (stockStack.empty()) {
        cout << "No stocks stored yet" << endl;
    } else {
        stack<Stock> tempStack = stockStack;
        cout << "\nStock Market Tracker (Stack):" << endl;
        while (!tempStack.empty()) {
            Stock stock = tempStack.top();
            cout << "Symbol: " << stock.symbol << " - Price: "
                << fixed << setprecision(2) << stock.price << endl;
            tempStack.pop();
        }
    }
}
```



```
int main() {
    int choice;
    do {
        cout << "\nStock Market Tracker Menu:" << endl;
        cout << "1. Add Stock" << endl;
        cout << "2. Search Stock" << endl;
        cout << "3. Delete Stock" << endl;
        cout << "4. Create New Stack" << endl;
        cout << "5. Display Stocks (Stack)" << endl;
        cout << "6. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                addStock();
                break;
            case 2:
                searchStock();
                break;
            case 3:
                deleteStock();
                break;
        }
    } while (choice != 6);
}
```

```
case 4:
    createStack();
    break;
case 5:
    displayStocks();
    break;
case 6:
    cout << "Exiting the program." << endl;
    break;
default:
    cout << "Invalid choice. Please try again." << endl;
}
} while (choice != 6);
return 0;
}
```

RESULT OUTPUT




```
Stock Market Tracker Menu:
1. Add Stock
2. Search Stock
3. Delete Stock
4. Create New Stack
5. Display Stocks (Stack)
6. Exit
Enter your choice: 1
Enter stock symbol: $
Enter stock price: 70
Stock added to the stack: $ - Price: 70.00
```

```
Stock Market Tracker Menu:
1. Add Stock
2. Search Stock
3. Delete Stock
4. Create New Stack
5. Display Stocks (Stack)
6. Exit
Enter your choice: 5
```

```
Stock Market Tracker (Stack):
Symbol: $ - Price: 70.00
```

```
Stock Market Tracker Menu:
1. Add Stock
2. Search Stock
3. Delete Stock
4. Create New Stack
5. Display Stocks (Stack)
6. Exit
Enter your choice:
```



CONCLUSION

The provided C++ program for a basic stock market tracker demonstrates the fundamental principles of data management and user interaction.

This program allows users to interact with a simple stock market tracker that uses a stack data structure to manage and manipulate stock data.

The program employs a simple text-based menu to facilitate user interaction. Users can choose from various menu options to perform their desired actions.

