DATA STRUCTURE & ALGORITHM

PROJECT REPORT

Stock Market Management System Using BST and SFML

GROUP MEMBERS	ENROLLMENT
IQRA MUSHTAQ	01-134241-018
YUSRA SHEIKH	01-134241-046

Department of Computer Sciences
BAHRIA UNIVERSITY, ISLAMABAD

Abstract

This project implements a Stock Market Management System that allows users to register, login, view available stocks, and purchase them. The system uses a **Binary Search Tree (BST)** to manage and search for stocks efficiently, with **case-insensitive string comparisons**. The **SFML (Simple and Fast Multimedia Library)** is used for graphical rendering of the user interface. The application provides a command-based input UI with dynamic feedback for user interactions. It demonstrates a real-world application of data structures (BST) combined with basic GUI elements to offer a more engaging user experience.

Contents

1. Project Title

Stock Market Management System Using BST and SFML

2. Introduction

The project is designed to simulate a simple stock market system, where users can register, login, browse available stocks, and purchase them. It uses a **Binary Search Tree (BST)** for fast stock lookups and SFML for rendering an interactive UI. Unlike a traditional console application, this system uses a graphics window with text prompts, feedback messages, and input handling.

3. Objectives

- Implement a **Binary Search Tree** to store and retrieve stock data efficiently.
- Enable **user registration and login** functionality.
- Allow users to **view and purchase** available stocks.
- Use **SFML** to create a graphical window interface.
- Demonstrate the combination of data structures and basic GUI systems in C++.

4. Tools and Technologies Used

Component	Description
Programming Language	C++
Graphics Library	SFML (Simple and Fast Multimedia Library)
IDE	Code::Blocks / Visual Studio / Dev C++
OS	Windows / Linux
Font File	Times New Roman for SFML text rendering

5. System Features

5.1 BST-Based Stock Management

- Stocks are stored in a BST.
- Case-insensitive string comparison for ordering.
- Insertion and searching are done recursively.

5.2 User Account System

- Users can register with a username and password.
- Existing users can login and access additional features.

5.3 Graphical UI with SFML

- Prompts and feedback messages rendered in a graphical window.
- Input handled via keyboard events in the SFML loop.
- Highlights user interactions in real-time.

5.4 User Interaction Flow

- Home screen with Register/Login options.
- Once logged in: View Stocks, Buy Stock, or Exit.
- Stock list displayed using BST inorder traversal.

6. Detailed Code Structure and Explanation

6.1 Node Structure

```
struct StockNode {
    Stock data;
    StockNode* left;
    StockNode* right;

    StockNode(Stock s) : data(s), left(nullptr), right(nullptr) {}

};
```

6.2 BST Insertion

Recursively adds a node maintaining BST rules:

```
int cmp = compareIgnoreCase(s.name, node->data.name);
if (cmp < 0)
    node->left = insertRec(node->left, s);
else

   node->right = insertRec(node->right, s);
```

6.3 Inorder Traversal

```
void inorder(StockNode* node, string& result)
```

6.4 Stock Search

```
bool searchRec(StockNode* node, const string& name, Stock& foundStock);
```

6.5 UI Rendering using SFML

- Uses sf::RenderWindow, sf::Text, and sf::RectangleShape.
- Input is handled in the SFML event loop.
- Text prompts update based on the screen state.

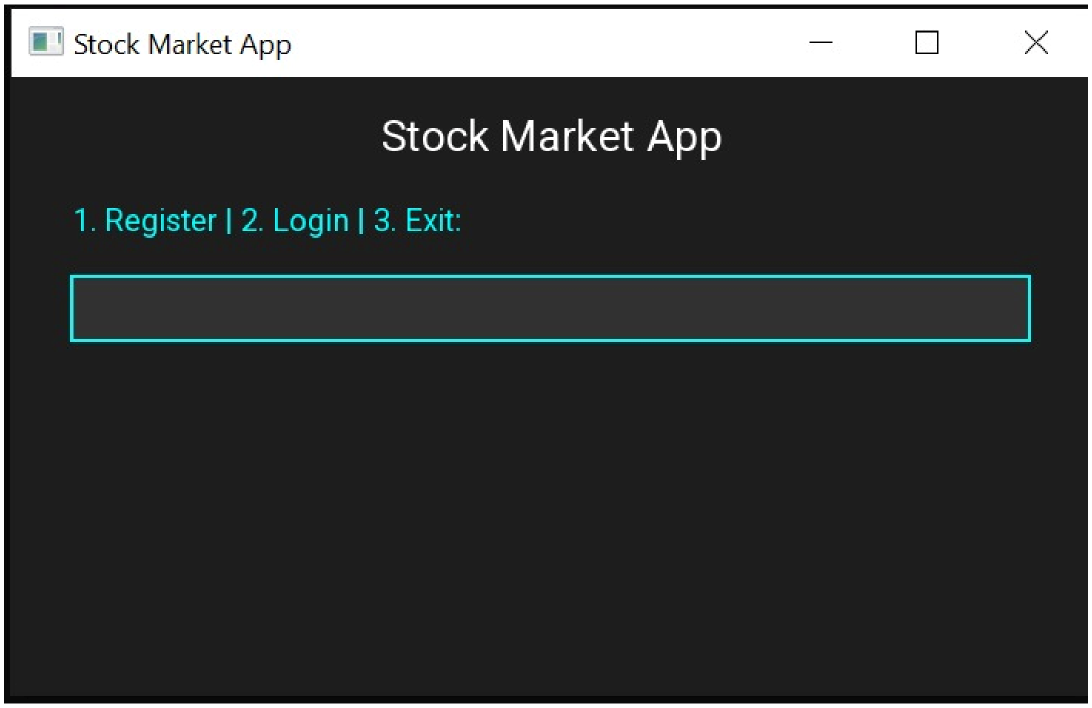
7.Code

```cpp
#include <SFML/Graphics.hpp>
#include <string>
#include <iostream>
#include <cctype>

using namespace std;

sf::Font globalFont;

struct Stock {
    string name;
    float price;
};

struct StockNode {
    Stock data;
    StockNode* left;
    StockNode* right;

    StockNode(Stock s) : data(s), left(nullptr), right(nullptr) {}
};

string toLower(const string& str) {
    string res = str;
    for (char& c : res) c = tolower(c);
    return res;
}

int compareIgnoreCase(const string& a, const string& b) {
    string la = toLower(a);
    string lb = toLower(b);
    if (la < lb) return -1;
    if (la > lb) return 1;
    return 0;
}

class StockTree {
public:
    StockNode* root;

    StockTree() : root(nullptr) {}

    void insert(Stock s) {
        root = insertRec(root, s);
    }

    void inorder(StockNode* node, string& result) {
        if (node == nullptr) return;
        inorder(node->left, result);
        result += node->data.name + " - $" + to_string(node->data.price) + "\n";
        inorder(node->right, result);
    }

    bool search(const string& name, Stock& foundStock) {
        return searchRec(root, name, foundStock);
    }
```

```cpp
private:
    StockNode* insertRec(StockNode* node, Stock s) {
        if (!node) return new StockNode(s);
        int cmp = compareIgnoreCase(s.name, node->data.name);
        if (cmp < 0)
            node->left = insertRec(node->left, s);
        else
            node->right = insertRec(node->right, s);
        return node;
    }

    bool searchRec(StockNode* node, const string& name, Stock& foundStock) {
        if (!node) return false;
        int cmp = compareIgnoreCase(name, node->data.name);
        if (cmp == 0) {
            foundStock = node->data;
            return true;
        }
        if (cmp < 0)
            return searchRec(node->left, name, foundStock);
        else
            return searchRec(node->right, name, foundStock);
    }
};

struct User {
    string username;
    string password;
    string ownedStocks[100];
    int stockCount = 0;
};

class StockApp {
private:
    sf::RenderWindow window;
    StockTree stockTree;
    User users[100];
    int userCount = 0;
    string currentInput;
    string tempUsername, tempPassword;
    sf::Text titleText, promptText, inputText, feedbackText;
    sf::RectangleShape inputBox;
    User* currentUser = nullptr;
    bool showFeedback = false;
    int screen = 0;

    void centerTitle() {
        sf::FloatRect bounds = titleText.getLocalBounds();
        float x = (window.getSize().x - bounds.width) / 2.f;
        titleText.setPosition(x, 20);
    }

    void drawUI() {
        window.clear(sf::Color(30, 30, 30));
        centerTitle();
        window.draw(titleText);
        promptText.setPosition(40, 80);
        window.draw(promptText);
        inputBox.setPosition(40, 130);
        window.draw(inputBox);
        inputText.setString(currentInput);
        inputText.setPosition(50, 135);
        window.draw(inputText);
        if (showFeedback) {
            feedbackText.setPosition(40, 200);
            window.draw(feedbackText);
        }
        window.display();
    }

    void showMainPage() {
        screen = 0;
        promptText.setString("1. Register | 2. Login | 3. Exit:");
        currentInput.clear();
        showFeedback = false;
        currentUser = nullptr;
    }

    void handleInput(const string& input) {
        showFeedback = true;
        if (screen == 0) {
            if (input == "1") {
                promptText.setString("Enter username:");
                screen = 1;
            }
            else if (input == "2") {
                promptText.setString("Enter username:");
                screen = 2;
            }
            else if (input == "3") {
                window.close();
            }
            else {
                feedbackText.setString("Invalid option.");
            }
        }
        else if (screen == 1) {
            tempUsername = input;
            promptText.setString("Enter password:");
            screen = 11;
        }
        else if (screen == 11) {
            tempPassword = input;
            users[userCount++] = { tempUsername, tempPassword };
            feedbackText.setString("Registered successfully. Welcome " + tempUsername +
"!");
            showMainPage();
        }
        else if (screen == 2) {
            tempUsername = input;
            promptText.setString("Enter password:");
            screen = 21;
        }
        else if (screen == 21) {
            for (int i = 0; i < userCount; i++) {
                if (users[i].username == tempUsername && users[i].password == input) {
                    currentUser = &users[i];
                    promptText.setString("1. View Stocks | 2. Buy Stocks | 3. Exit:");
                    feedbackText.setString("Login successful.");
                    screen = 3;
                    return;
                }
```

```cpp
                }
                feedbackText.setString("Invalid credentials.");
                showMainPage();
            }
            else if (screen == 3) {
                if (input == "1") {
                    string stockList = "Available Stocks:\n";
                    stockTree.inorder(stockTree.root, stockList);
                    feedbackText.setString(stockList);
                }
                else if (input == "2") {
                    promptText.setString("Enter stock name to buy or 'exit':");
                    screen = 4;
                }
                else if (input == "3") {
                    showMainPage();
                }
                else {
                    feedbackText.setString("Invalid option.");
                }
            }
            else if (screen == 4) {
                if (input == "exit") {
                    promptText.setString("1. View Stocks | 2. Buy Stocks | 3. Exit:");
                    showFeedback = false;
                    screen = 3;
                    return;
                }
                Stock found;
                if (stockTree.search(input, found)) {
                    currentUser->ownedStocks[currentUser->stockCount++] = found.name;
                    feedbackText.setString("Stock bought: " + found.name);
                    promptText.setString("1. View Stocks | 2. Buy Stocks | 3. Exit:");
                    screen = 3;
                }
                else {
                    feedbackText.setString("Stock not found. Try again or type 'exit'.");
                }
            }
        }

    public:
        StockApp() : window(sf::VideoMode(700, 400), "Stock Market App") {
            if (!globalFont.loadFromFile("Roboto.ttf")) {
                cerr << "Failed to load Roboto.ttf" << endl;
            }

            stockTree.insert({ "Apple", 150.0f });
            stockTree.insert({ "Microsoft", 305.5f });
            stockTree.insert({ "Amazon", 3300.5f });
            stockTree.insert({ "Tesla", 920.0f });
            stockTree.insert({ "PTCL", 8.5f });

            titleText.setFont(globalFont);
            titleText.setCharacterSize(28);
            titleText.setFillColor(sf::Color::White);
            titleText.setString("Stock Market App");

            promptText.setFont(globalFont);
            promptText.setCharacterSize(20);
            promptText.setFillColor(sf::Color::Cyan);
            inputBox.setSize(sf::Vector2f(620, 40));
            inputBox.setFillColor(sf::Color(50, 50, 50));
            inputBox.setOutlineThickness(2);
            inputBox.setOutlineColor(sf::Color::Cyan);

            inputText.setFont(globalFont);
            inputText.setCharacterSize(20);
            inputText.setFillColor(sf::Color::White);



            feedbackText.setFont(globalFont);
            feedbackText.setCharacterSize(18);
            feedbackText.setFillColor(sf::Color::Green);

            showMainPage();
        }

        void run() {
            while (window.isOpen()) {
                sf::Event event;
                while (window.pollEvent(event)) {
                    if (event.type == sf::Event::Closed)
                        window.close();

                    if (event.type == sf::Event::TextEntered) {
                        if (event.text.unicode == 8 && !currentInput.empty()) {
                            currentInput.pop_back();
                        }
                        else if (event.text.unicode == 13 || event.text.unicode == '\n') {
                            handleInput(currentInput);
                            currentInput.clear();
                        }
                        else if (event.text.unicode < 128) {
                            currentInput += static_cast<char>(event.text.unicode);
                        }
                    }
                }
                drawUI();
            }
        }
    };

    int main() {
        StockApp app;
        app.run();
        return 0;
    }
```

8.Menu-driven Interface

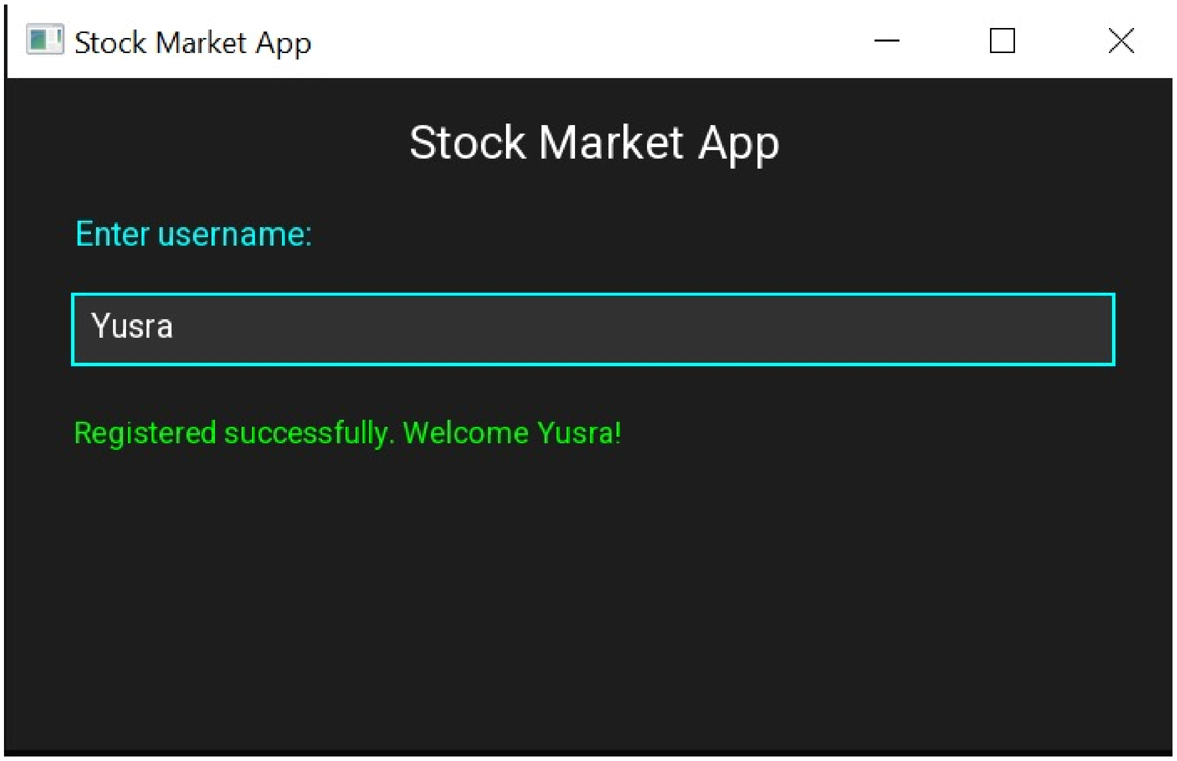


Stock Market App

Stock Market App

1. Register | 2. Login | 3. Exit:

9. Execution Flow



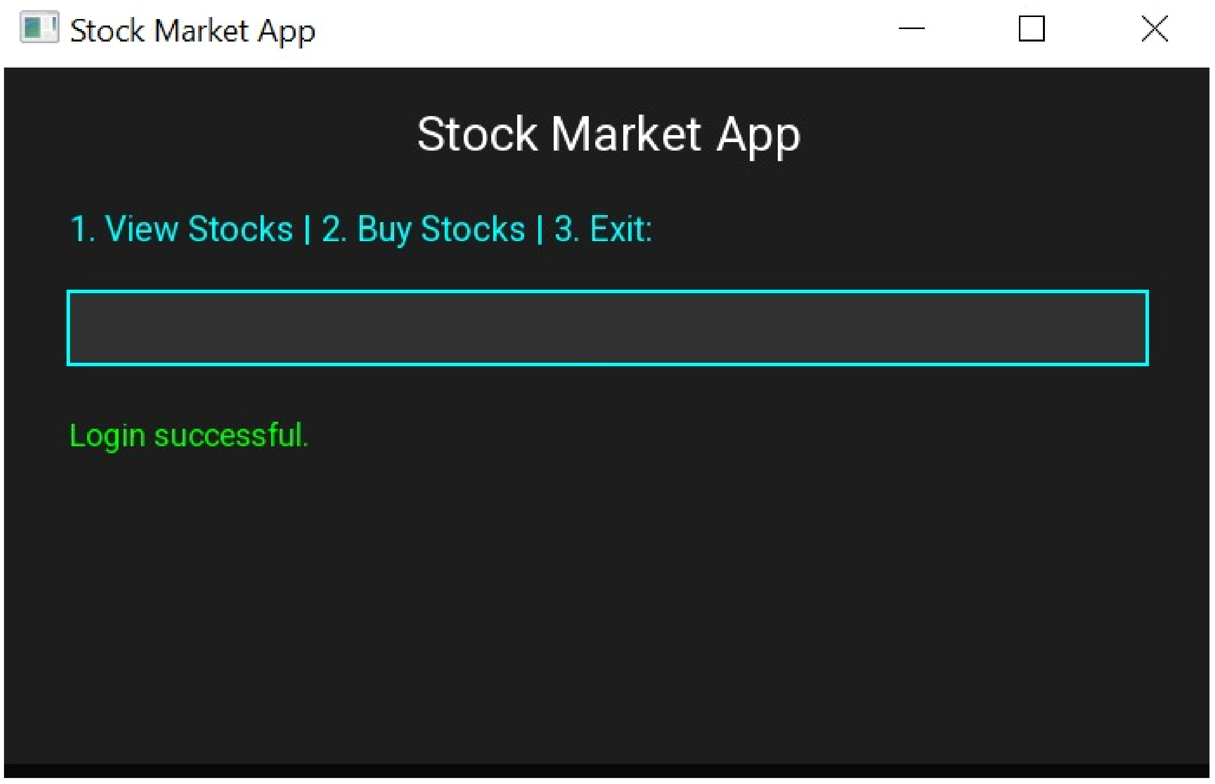
Stock Market App

Stock Market App

Enter username:

Yusra

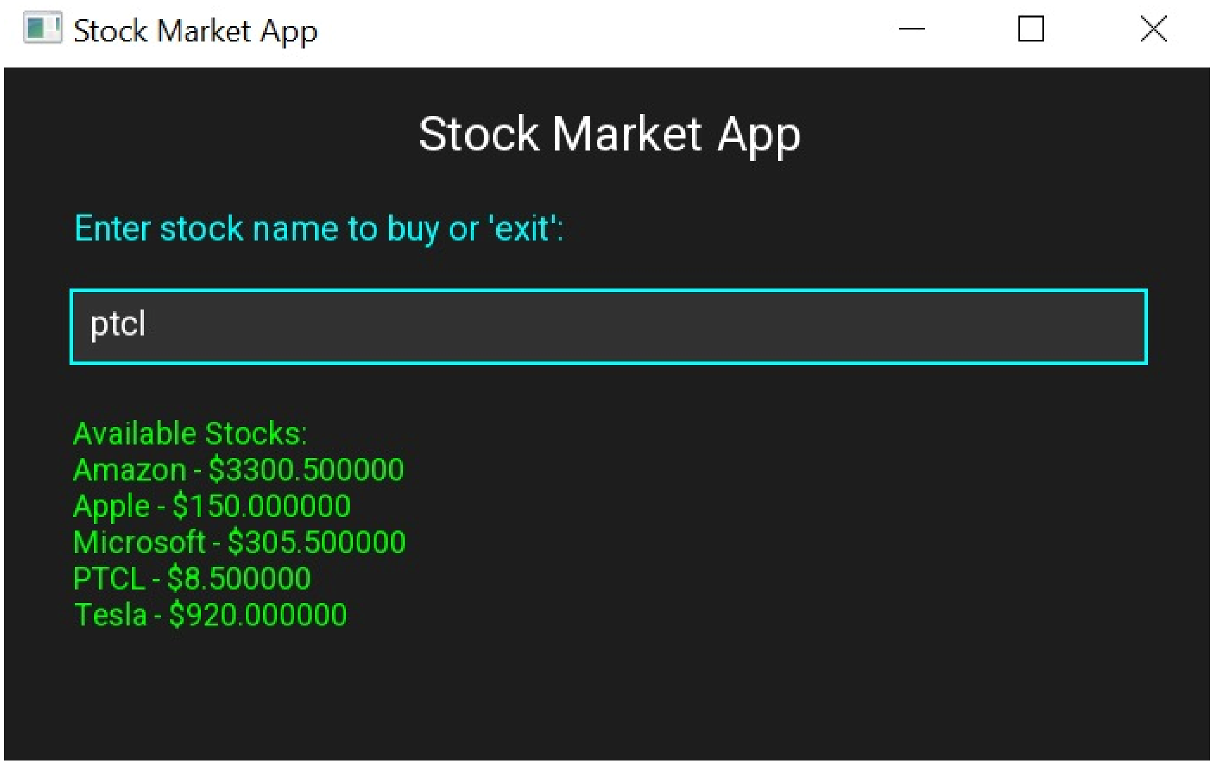
Registered successfully. Welcome Yusra!


Stock Market App

Stock Market App

1. View Stocks | 2. Buy Stocks | 3. Exit:

```
Login successful.
```

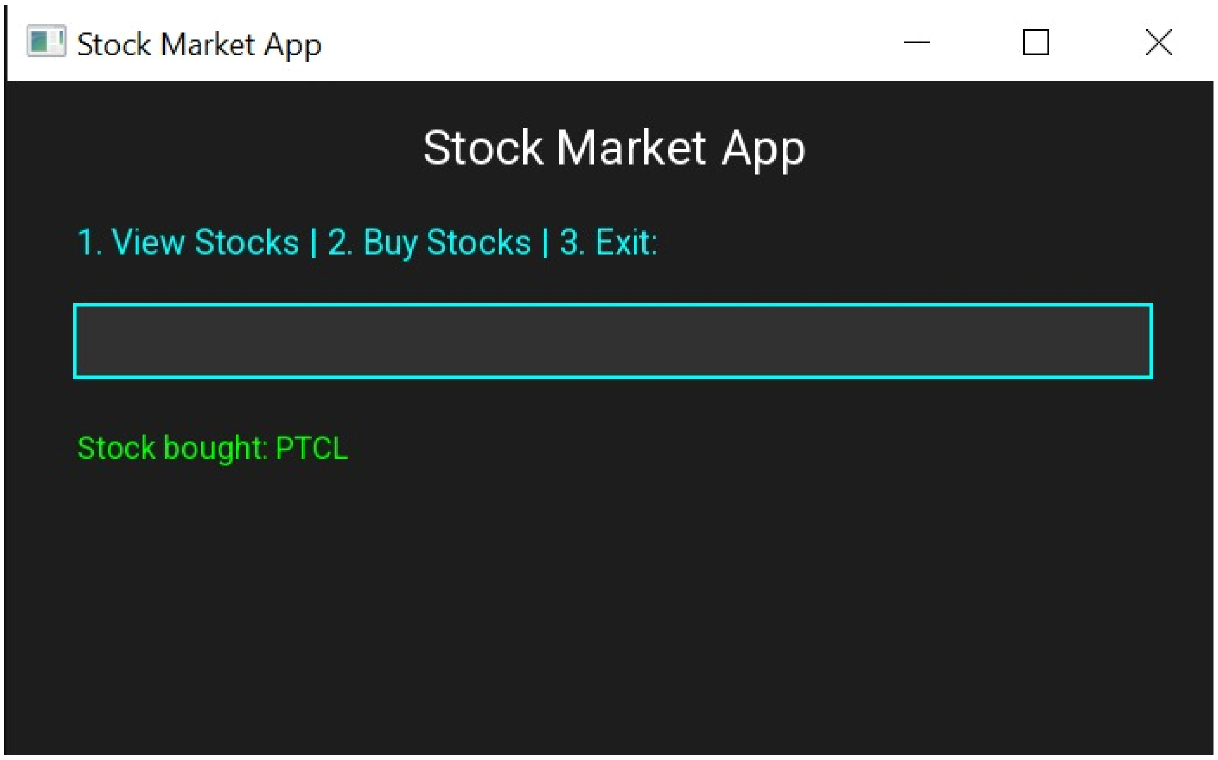

Stock Market App

Stock Market App

Enter stock name to buy or 'exit':

ptcl

```
Available Stocks:
Amazon - $3300.500000
Apple - $150.000000
Microsoft - $305.500000
PTCL - $8.500000
Tesla - $920.000000
```



Stock Market App

1. View Stocks | 2. Buy Stocks | 3. Exit:

```
Stock bought: PTCL
```

10.Challenges Faced

- Handling string comparisons case-insensitively in BST
- Managing screen transitions cleanly within the SFML loop
- Designing user feedback and prompt flows
- Keeping UI responsive with keyboard-only input

11.Future Improvements

- Add stock selling functionality.
- Implement stock quantity and price updates.
- Use file I/O to persist users and stocks.
- Show owned stocks in UI.
- Add animation or visual representation of the tree.

12.Conclusion

This project effectively integrates **Binary Search Tree data structures** with a **graphical UI** using SFML to build an educational and functional stock market simulation. It demonstrates how C++ data structures can be applied to real-world-style applications, with a focus on clean logic, user management, and efficient searching.

13.References

- SFML - Simple and Fast Multimedia Library
- cplusplus.com
- GeeksforGeeks - Binary Trees
- TutorialsPoint - Trees

14. Links

Github link:
Click here
LinkedIn link:
Click here