SOFE 4790U: Distributed Systems (Fall 2023)

Instructor: Dr. Ahmed Badr

Assignment #1

Honour code: By submitting this assignment, I (name and banner id# below) affirm this is my own work, and I have not asked any of my fellow students or others for their source code or solutions to complete this assignment, and I have not offered my source code or solutions for this assignment to any of my fellow students.

Name: David Fung

Banner ID#: 100767734

GitHub link: https://github.com/UOITEngineering2/programming-assignment1-sofe-4970u-fall-2023-d-fung

Introduction

The project idea was to provide an interactable user interface for users to search for the real-time price of any stock and cryptocurrency in USD. This application also provides a functional sign-up/login verification for users before they can access the server functionality. The server and client communication were implemented using sockets, along with multi-threaded handling of client connections.

Core Functionalities

The first core functionality is to return a real-time price of any stock found on the NASDAQ Stock Market. This is done by passing a stock ticker (eg. AAPL, TSLA, AMZN) to the server-side method which sends an HTTP GET request to retrieve data from an API. The price quote will then be displayed onto the interface for the user to see. The API used in this project is called “Twelve Data” and can be found on the RapidAPI website.

The second core functionality is similar but instead retrieves the real-time price of any cryptocurrency currently listed on Binance. This was implemented by also sending an HTTP GET request to Twelve Data’s API.

Novel features

The first novel feature of the project is the implementation of a graphical user interface (GUI). A GUI is more intuitive than using the command-line interface and provides visual feedback. A GUI was developed for the user sign-up/login process that provides visual feedback on errors. Additionally, a GUI was also created for the stock and cryptocurrency price look-up functionality that allows users to intuitively interact with input fields and buttons to search for the price.

The second novel feature is the user sign-up/login system that can store a user's credentials in the server's shared resource. This means that any client can log-in if the user's credentials have already been saved from before even though a different thread will be serving the client. This system checks if there is already an existing matching username and password that has been stored and will allow the user to login if successful.

Challenges and Solution

The first main challenge was to implement the multi-threading aspect of the server and client. I found that keeping the server methods separated from the Thread class, and implementing a while loop with a client request listener was the best approach to this. The while loop will only terminate when the client disconnects.

The second challenge was to implement a way for the client to call a method on the server-side. This was done by passing in a string with the method request (eg. signUpRequest) using dos.writeUTF() on the client and br.readUTF() on the server. The server then uses a switch statement to call the correct method that the client has requested.
Another challenge that was encountered is to parse the data returned by the API. This data was returned as a response body in JSON format, and is not very user-friendly. I was able to use a Java library to parse the response as a JSON object and convert it to a decimal number to be displayed.

Testing and Description

Junit tests were used to test the functionality of the server methods.

```
public class ServerTests {
    private Server server;
    // Create a new server before running the unit tests
    @Before
    public void initializeServer() throws Exception {
        server = new Server("testing");
    }
```

This @Before method gets called before every unit test, which creates a new server in the testing environment.

```
    // This tests the method to add a new user to the userCredentials hashmap
    @Test
    public void testAddUser(){
        server.addUser("user1", "password1");
        assertTrue(server.checkUser("user1", "password1"));
    }
```

This unit test testAddUser tests that the user has been correctly added when they sign up an account on the interface.

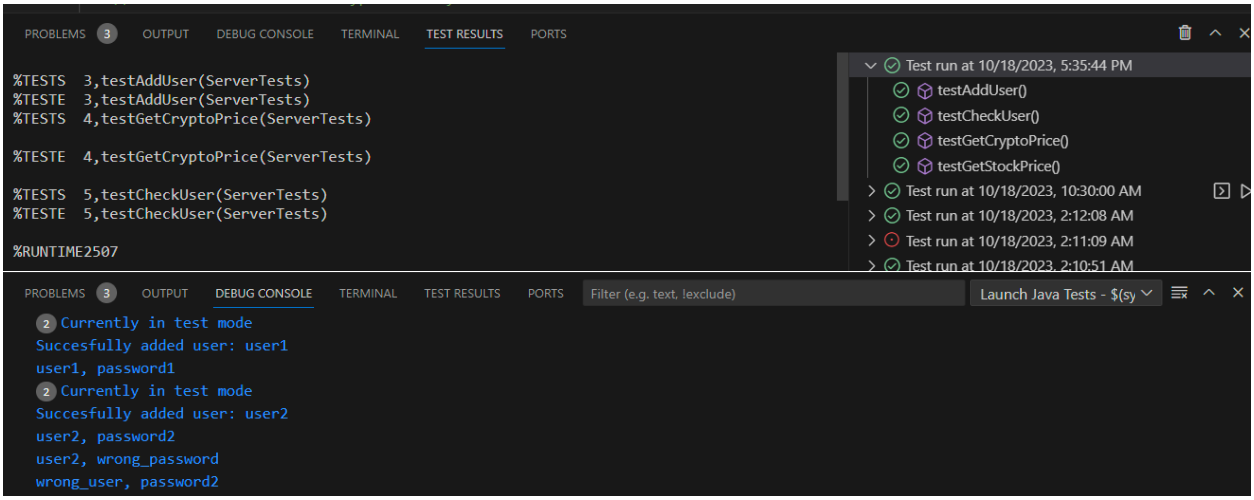
```
    // This tests the method of checking if the username and password matches
    // an existing user in the hashmap
    @Test
    public void testCheckUser(){
        server.addUser("user2", "password2");
        assertTrue(server.checkUser("user2", "password2"));
        assertFalse(server.checkUser("user2", "wrong_password"));
        assertFalse(server.checkUser("wrong_user", "password2"));
    }
```

This unit test testCheckUser will check for a matching username and password entry on the hashmap that stores the accounts. It will return true if it exists, and false if it does not.

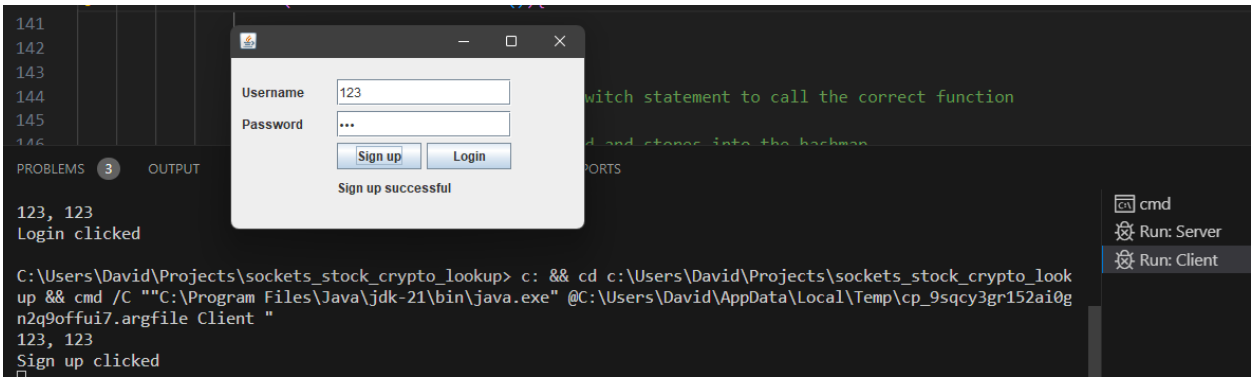
```
    // This tests the method of checking if the API request is working
correctly
    // Since the real-time price is always changing, we can just check if it is
not null
    @Test
    public void testGetStockPrice(){
        assertNotNull(Server.getStockPrice("aapl"));
    }

    // Same as above but for cryptocurrency
    @Test
    public void testGetCryptoPrice(){
        assertNotNull(Server.getCryptoPrice("btc"));
    }
```

These next two tests checks that the server method to create an HTTP request for the API is working correctly. However, since the real-time price is always changing, it is not possible to directly assert a value comparison. Instead, it is possible to check if it does not return a null object instead.

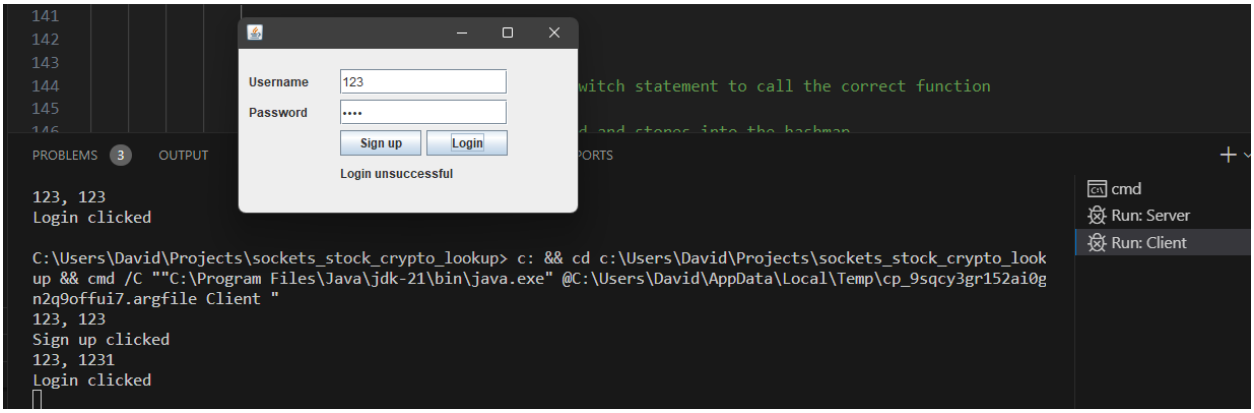
Application Demonstration



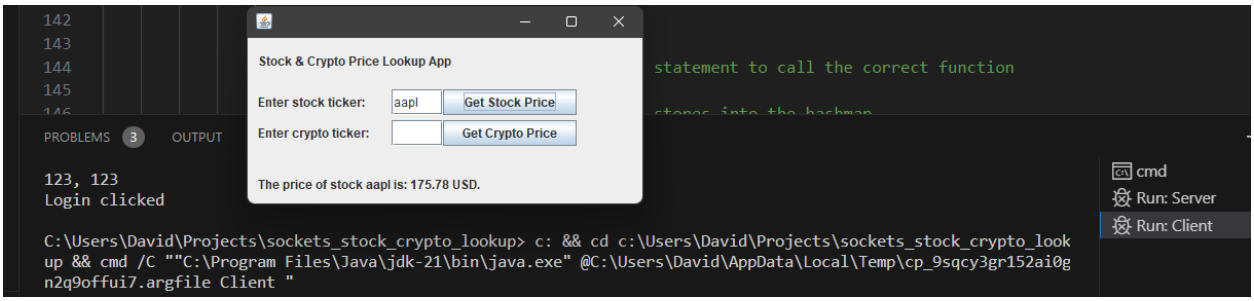
Here we can see that all the unit tests have passed along with the debug console showing the parameters being passed.



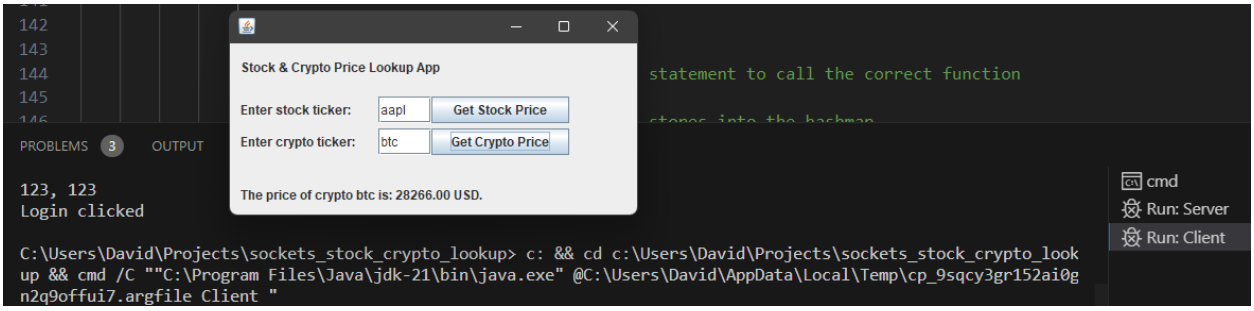
The screenshot above shows the user signup process.



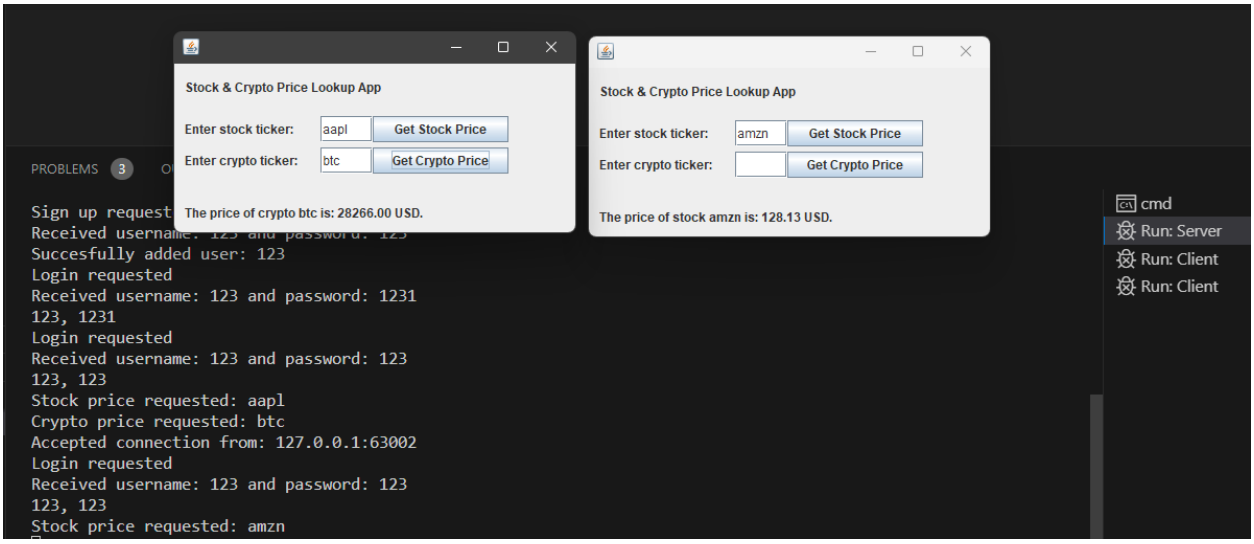
The screenshot above shows an unsuccessful signup.



The screenshot above shows page being opened after the login process, and the stock price retrieval and result.



The screenshot above displays the cryptocurrency price retrieval and result.



The screenshot above demonstrates multithreading because the server is able to serve multiple clients simultaneously.