Stock Price Monitoring System

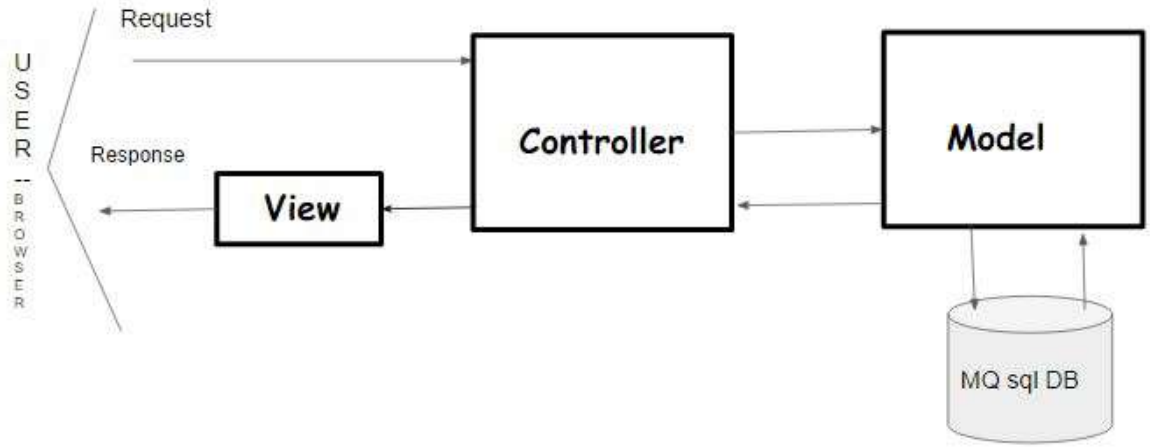
Table of Contents

1. Introduction:

i. **Purpose:** This purpose of the stock monitoring web application is to allow the user to add stock details to his existing database and also refresh the existing values of stocks every five minutes. There are various companies viz. Yahoo, NASDAQ etc. which allow user to query their database to get the latest price of the stock and other details if necessary. In this application user can add the stocks that he wants and monitor their performance on a timely basis.

ii. **Scope:** The scope of this document is to describe the milestones, tool & technologies used for implementing this project. The milestone for this small project was divided into major three phase viz. design, implement, and test.

2. Flow Diagram:



The basic flow is described in the image above. As soon as the application runs and the user opens the home page, the user is provided with few functionalities for adding, viewing, deleting the entries (mentioned below in the doc). When the user chooses a respective functionality a request is sent to the backend server and depending on the request a response is received.

3. Component:

I used MVC design pattern for the development of the project. This project has the following parts:

i. A spring boot web application with REST API to perform the CRUD operations. As soon as the application is launched, it starts fetching data that already exist in the database and update them.

ii. A UI where a user can add companies for monitoring, delete an existing company, get the latest prices in the database, and get the history of prices of a particular company if exists in the database.

iii. Database which stores all the necessary information and a backend java server which does the required operations.

Database:

I created two tables "real_time_stock_info" and "all_time_stock_info" which stores the data.

- **Real_time_stock_info**: It contains the latest data of stocks that was ***most recently fetched*** from the Yahoo finance API. This table is updated in two ways. ***Firstly***, if the user adds a company stock that **already** exists in this table then the entry is **updated** with current timestamp. ***Secondly***, after 5 mins when the java program runs, it updates **all** the values that exist in this database.

- **All_time_stock_info:** It contains all the data of stock since the application was first started.

Classes:

The data fetched from the Yahoo Finance API is converted into defined type **MyStock** which contains only the required information (Symbol, Company Name, Price, time it was fetched). Once data was fetched and taken into required format, it was inserted into MySQL database using JDBC. **StockDaoImpl** is the DAO layer which does the required operation.

The **data fetching** from Yahoo finance API is taken care by **StockService**. Once the fetched data is converted into the required type it is passed to DAO for inserting into database.

StockController is the class which takes the appropriate request from the user and sends the respective response. To update the stock values every five mins; I have used spring boot's "enable scheduling Annotation". The schedule annotation allows a function to be triggered every t mins (300000 ms in our

case) which is passed as an argument to the annotation from the start of the application. It is called from controller as mentioned below:

```
@Scheduled(fixedRate = 300000)
public void updateEveryFiveMin(){
        stockService.updateEveryFiveMin();
}
```

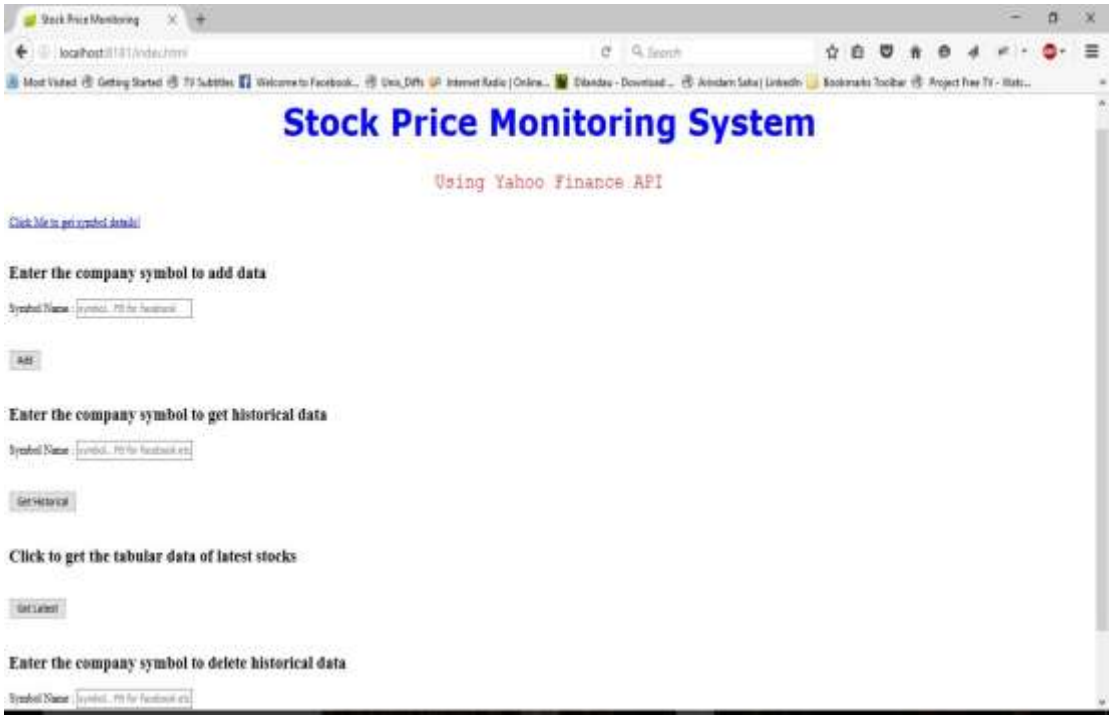
UI Component:

The UI is implemented using HTML and few JS functionality. To render the graph I used Google's chart visualization. The functionalities provided to user are:

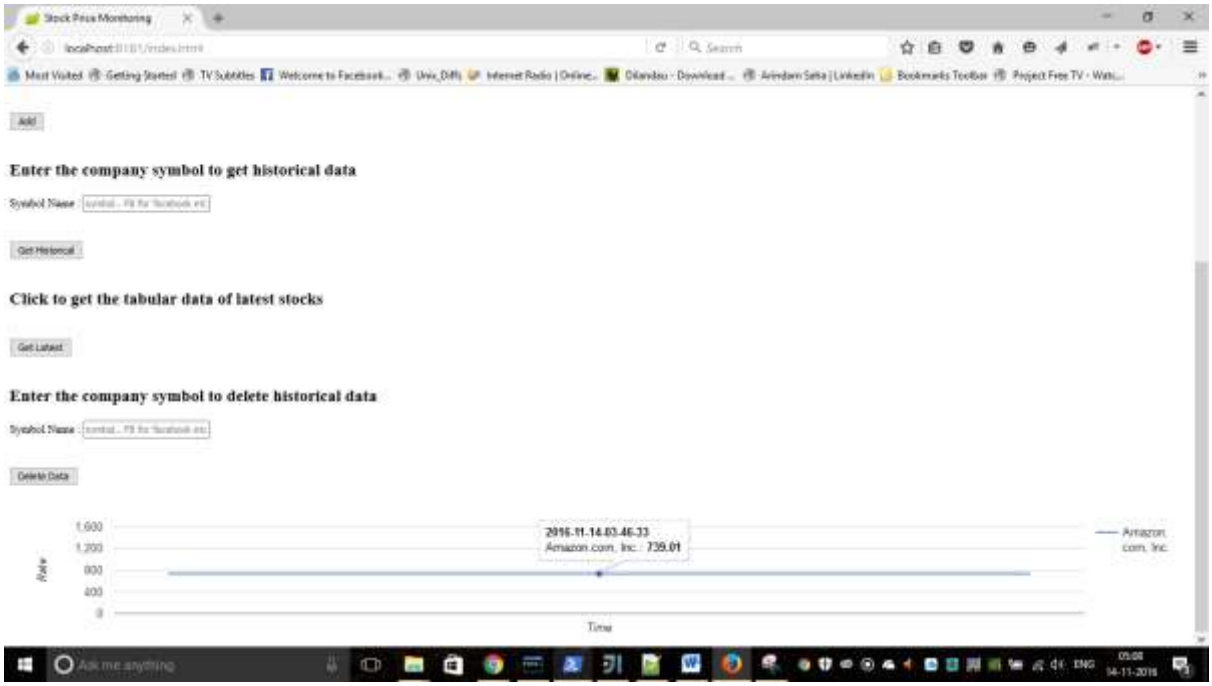
- **Add** a company to the Database by entering the company symbol. The correct companies' symbol can be accessed from the link provided on the top of HTML page.
- To **get** the **historical** data of the company present in the database. **Note: Please launch the application in Mozilla as the graph rendering had some issues in Chrome.**
- To **get** the **latest** stock values present in "real_time_stock_info" table.
- **Delete** all stock data (real_time and all_time) of the entered company.

4. Screenshot:

Front Page:

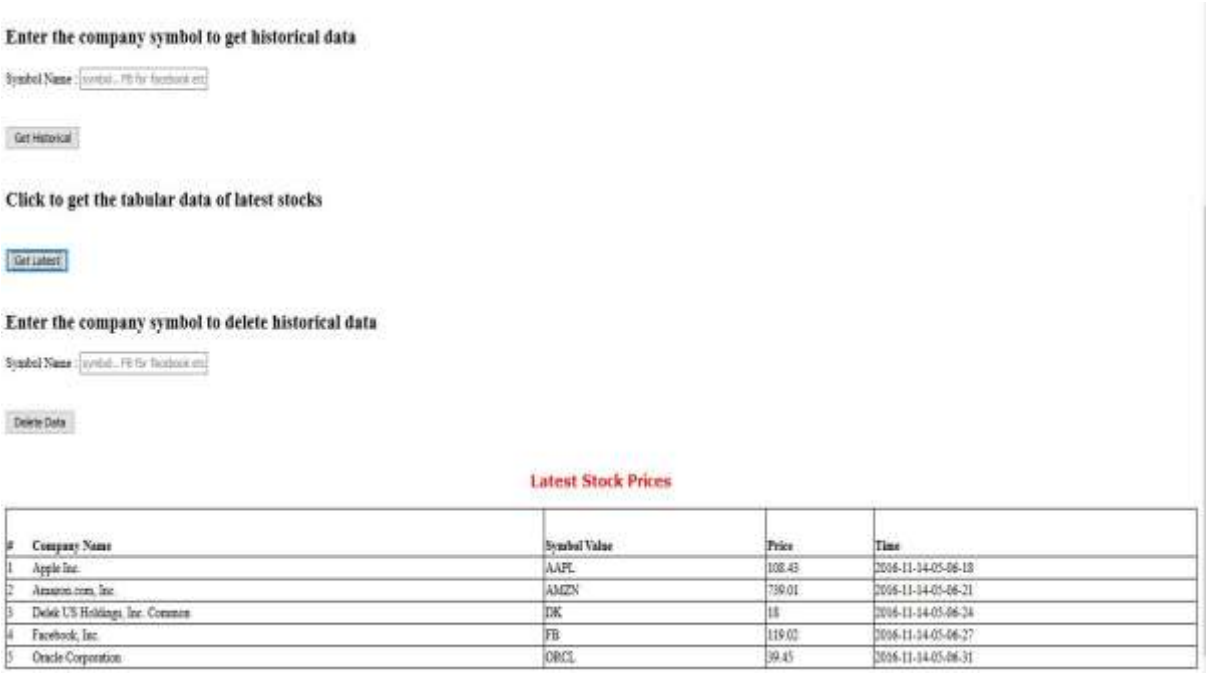


Graph:



NOTE: **The values on y axis (price) in the graph are same as the data fetched from yahoo finance API is too frequent and stored in the database. As per the requirement mentioned on Hackerrank, we have to plot the historical stock values which will be taken from our database. The chance of change in price on same day between every 5 mins is less. If we fetch data less frequently, say couple of days, and then plot the values; the chances of having different prices on y axis will be high.**

Tabular Data (Get latest stock price from table real_time_stock_info):



5. Build and Run:

Assuming **maven** is installed and **PATH** is set:

Edit **application.properties** (src/main/resources) file to change the following values:

- `spring.datasource.username=yourSqlserverUserName`
- `spring.datasource.password=yourSqlserverPassword`

Rest values can remain unchanged if db is created with the following values:

i. Set up MySQL environment first.

   a. **Port :** 3306
   b. **Db name :** stock_info
   c. **Two table :**
      i. all_time_stock_info -> ***with column name*** -> symbol,comp_name,stock_value,time_val
      ii. real_time_stock_info -> ***with column name*** -> symbol,comp_name,stock_value,time_val

ii. Go to project root **i.e. Solution** and run the following commands

   a. mvn clean
   b. mvn compile
   c. mvn spring-boot:run

6. **Link:**

GitHub repo link: https://github.com/amishra22/StockPricesMonitoring