CS 157A Intro to Database Management Systems

Project Requirements

Project Title: Stock Data Aggregation Web-App

TEAM 34

Sachin Shah

Yang Li

En-Ping Shih

Professor: Dr. Mike Wu

TA: Sriram Priyatham Siram

San Jose State University

September 24th, 2019

Project Overview

Our project will be a web-based application which can handle user/client input and report back to the user all relevant information regarding a certain company's stock. The application will provide up-to-date, useful information on the security of the client's choice and also recommend to the client whether they should purchase the security in question. The application will do this by first retrieving information from various online sources for financial information. Then the application will perform operations using the retrieved data to determine if the security should be purchased by the user. For example, retrieving the beta value, which is a measure of a securities volatility, could be compared to a benchmark value. Based on this comparison, the security could be assigned a Boolean value of "buy" or "sell." Another function of our application would be storing the results of client queries in a database which will be hosted on a remote server.

The primary stakeholders for this application are people who may not have extensive knowledge of the stock market and are looking for a way to get started with investing. Furthermore, these are people who are interested in investing in public securities and not within the private sector, as a lot of the information on private companies is difficult to come by online. Our stakeholders value time and convenience when it comes to retrieving reliable information about publicly traded securities. Since this is a web based application and it meeting the functional requirements requires it to be used by end-users, this stakeholder group outlined above is of the upmost importance. Our application is important because nowadays speculators and passive investors are looking for a way to get a buy or sell decision without having to do extensive research. Our site will retrieve information and not require the user to do anything other than enter the ticker name of a security. The user saves time, energy, and resources while receiving a user-oriented service.

System Environment

To build this application successfully, we will need to set up the environment based on the three-tier architecture (Figure 1) which contains the client, the server, and the database. On the client part, we have to make sure our web-based application is working on internet browsers on any computer or laptops for our users and clients. (ex. Google Chrome and Safari), and JavaScript will be our frontend development language. We plan to use Apache Tomcat as our application server which will host the Java application. Tomcat contains a web-layer and an application layer, so it can handle incoming HTTP requests and outgoing connections to the MySQL server that we plan on using. The user will be able to interact with the Java application through a link on their browser. We will use JSP to allow for the dynamic generation of HTML which will display the application on our user's browser. The Java GUI will be developed using the swing library. The use of this library to develop a GUI will allow us to develop a fully functioning GUI. The type of GUI we plan to use is a Process User Interface. Our intended application flow intends to allow the user to make step-by-step decisions for querying and storing data. For the stock data, we will be using Alpha Vantage API which have real-time and historical stock data as our default data. Since AWS (Amazon Web Services) has a toolkit that is compatible with Eclipse, we will be using the AWS toolkit to handle the database instances (Figure 2.)

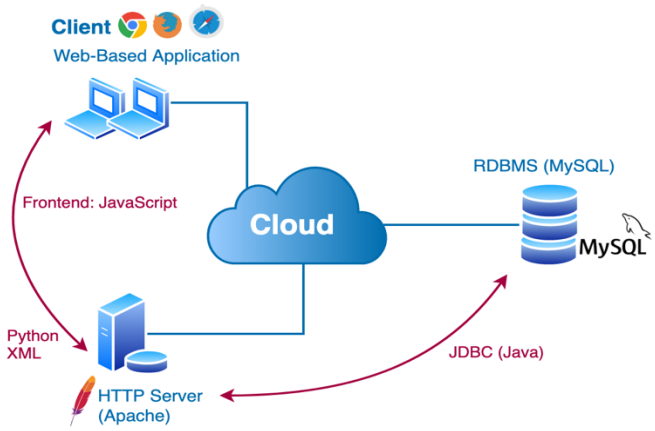

Figure 1. Three-Tier Architecture

- **HW/SW used**

  Apache Tomcat, AWS, Google Identity Platform, Alpha Vantage API

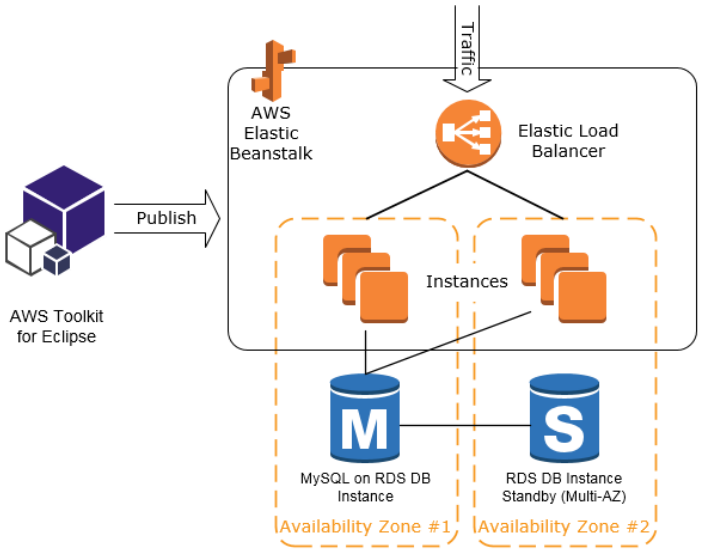

Figure 2. Deployment Diagram for application on AWS

- **RDBMS used**

  MySQL Community Edition 8.0.17

- **Application Languages**

  Java 8, Java EE, HTML, JSON, JSP, SQL, Javascript

Functional Requirements

How Users will Access the System

Our application is targeted towards consumers, and we will keep track of registered accounts using Google's Identity Platform. Users can access our application using a URL which they can access through their browser. Hosting our application on AWS will allow for this. The Identity platform incorporates a sign-in API where users can create an account on our application using their existing Gmail account. Users will have limited read and write capabilities. They will have the ability to query stock information from the GUI and based on their preferences, add the stock

to their own watchlist. This watchlist is the only entity that users will have read, write, and delete capabilities for. The watchlist will be modeled using a table in the SQL database which will be associated with a user's unique PID, which will be generated upon account creation. Query results will also be stored, and users will have ONLY READ CAPABILITIES.

Functions:

Search for Stock Information

- User's should be able to enter input in the form of a stock ticker, and the application should display that stock's information on the presentation tier.
- Information about the query should be stored in the database layer so that the user can access the search results in the future

View Past Search Results

- Users should be able to see a view of their past search results after entering a time frame
- The system should be able to display past search results within the time frame that the user provides
- Users' READ ONLY capabilities of all search results

Add Stock to Watchlist

- After a user sees a given search result, they can choose to add the stock to their personal watchlist. This is the user's write capability to their own watchlist

View Entire Watchlist

- User's should be able to see a view of their entire watchlist.
- Input should be a button click, and the system should produce the output of a user's entire stock watchlist

Delete Stock from Watchlist

- User's should be able to remove a stock from their own watchlist by entering the ticker of the stock they want to delete
- If the stock is in the watchlist, the stock will be deleted from the watchlist table
- If the stock is not in the watchlist, the system will display a message saying "Stock not found in Watchlist"

Compare Two Stocks from Watchlist

- User's should be able to enter two tickers of stocks that are currently on their watchlist and see a side-by-side comparison of metrics
- The system shall generate a view on the client-side of the two stocks selected and their information

Buy-or-Sell Decision

- User's shall be able to select a given stock from their watchlist as input and be given an output of buy or sell
- The system shall be able to query from the AlphaVantage API and provide a Boolean value based on selected metrics

Non-Functional Requirements

- **Front-end:** HTML, CSS, JavaScript / React
- **Server**: Node.Js
- **Database**: using MySQL and JDBC, to store stocks information and personal stocks watchlists.

Access control and Security: using Google Identity Platform, using firebase to store user account information such as username and password.

A second non-functional requirement is an issue regarding the security of user data. When our users create a new account, their information, such as user id, password, and email need to be

stored in a secure location. To address this issue, we plan on using the Google Sign-in API via the Google Identity platform, a third-party service which allows developers to incorporate a secure way for users to use their existing google accounts to log in to our web-app.

In terms of access to the database storing users' stock watchlists, access control will be handles in the logic layer of our application. Once a user is logged in, they can edit and generate a view of the table associated with their unique user ID. They will only have access to their own watchlist.