**CS 6314.502 Web Programming Languages**

**Project Report**

**Group Name: FULLSTACK ROVERS**

Group Members:

Adarsh Raghupati Hegde

Akash Prakash Akki

Keerti

**Architecture**

**A close up of a device

Description automatically generated**

**Module Description:**

**Database:**

Technologies considered: Oracle database, PostgreSQL

Selected Technology: PostgreSQL

Reason: PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features thathelp you manage your data no matter how big or small the dataset.

**Stock Brokerage App and Stock Exchange App:**

Technologies considered: Spring Boot

Selected Technology: Spring Boot

Reason: Spring Boot is a java framework which provides ORM libraries for database operations. It provides connectors to distributed cache like Redis and connectors to distributed message servers like kafka. Easy to understand and implement rest api with the help of various annotations.

**Cache:**

Technologies considered: Memcached, Redis

Selected Technology: Redis

Reason: Both tools are powerful, fast, in-memory data stores that are useful as a cache. Both can help speed up your application by caching database results, HTML fragments, or anything else that might be expensive to generate.

**Queue for Asynchronous Service:**

Technologies considered: Kafka, RabbitMQ

Selected Technology: Kafka

Reason: Kafka offers much higher performance than message brokers like RabbitMQ. It uses sequential disk I/O to boost performance, making it a suitable option for implementing queues.

**Services available to the end users:**

1. **User Registration**:User can create an account using Username(>6 char), Address, E-Mail, Security Question(used in case of resetting the password) and a password(>8 char).
2. **Login**: User can login to the application using his username and password to access special functionalities like Buy, Sell of stocks.
3. **Logout**: User can log out of the application thus maintaining the confidentiality of his data
4. **User profile information display and editing**: Once the user logs in, he can view his information like name, address, email and the stocks he owns currently and his linked bank details. Also, can edit his profile information like email, address add/remove bank account information.
5. **Forgot password functionality**: If the user forgets his password then he can reset it using the security question he had used while registering.
6. **Add one or more bank accounts**: The following details can be used to connect to the bank accounts which can be used in stock transactions.

Bank Routing Number

Bank Account Number

1. **Ability to transfer money from/to bank accounts**: User can transfer money from his multiple bank accounts linked within the application**.**
2. **A real-time stock exchange Web Application that return the price for a particular stock**

Current price b. Price history for given date range.

1. **Search and find a stock of interest**: Any user (Registered and unregistered User) can view the current stock price or the history based on the day, month, year etc

Display current Stock price

2. Display selected Stock price history

1. **Ability to buy/sell stocks**: Registered User can buy/ sell a stock as follows

1. Schedule:

a. One-time

b. Recurring based on a set schedule

2. User can select one or more stocks and quantity to sell/buy

3. User can modify/delete a recurring stock buy/sell schedule

**Web Services Supported:**

1. **TLS/SSL:** Encrypted the communication channel between the client (i.e. browser), web site server, and Web Services server using TLS/SSL.
2. **Scheduled buy and sell of shares:**scheduled stock sell/buy processing request to the web service API results in the request being added to a queue. The relevant “stock selling processor” web service retrieves this request from the queue and processes it.
3. **Caching:**Highperformance distributed caching technique - Rediscacheis used to stores the stock details which are repeatedly searched and thereby increases the throughput of the application.
4. **Asynchronous buy and sell services:** “stock sell/buy processing” request to the web service API results in the request being added to a queue. The relevant “stock selling processor” web service retrieves this request from the queue and processes it.

**Problems encountered:**

1. Getting real time stock data: Fetching the real time data from a third party api was challenging. We were able to solve this by understanding the api specifications given by the Alphavantage.
2. Achieving TLS/SSL communication between different components: Achieving TLS communication between two spring boot servers was solved by setting up the custom SSL context with self-signed certificates in each spring servers.
3. We found difficulties in constructing the JSON data from the HTML input forms. After exploring few JavaScript and jQuery we were able to solve the problem.
4. Configure and modify scheduled task: We faced few challenges while implementing the recurring buy feature. By using the scheduled annotation feature provided by spring boot and by storing the user specific scheduling parameters in the database we were able implement the feature.