|  |
| --- |
|  |

|  |
| --- |
| **Stock Market Charting (Mid Tier) Phase5 v2.0** |
| Case Study |
|  |
| This document covers Software Requirements of Stock Market Charting, along with list of Technologies to be used to develop this Software System, and also includes some details on the Architecture |
|  |
| **IIHT** |
| **8/7/2019** |
|  |

Table of Contents

[1. Business Requirement (Stock Market Charting) 2](#_Toc13184301)

[1.1. Admin Use Cases: 2](#_Toc13184302)

[1.2. User Use Cases: 3](#_Toc13184303)

[1.3. Company related Data Fields: 3](#_Toc13184304)

[1.4. Stock Price details Excel: 4](#_Toc13184305)

[1.5. IPOs planned: 4](#_Toc13184306)

[1.6. Sectors data Fields: 4](#_Toc13184307)

[1.7. User DB Table: 4](#_Toc13184308)

[1.8. Stock Exchange Data Fields: 4](#_Toc13184309)

[2. Full Stack Technologies 5](#_Toc13184310)

[3. Microservices Integration and Security 5](#_Toc13184311)

[3. Spring Microservices Tools to be used 6](#_Toc13184312)

[4. JWT Authentication 6](#_Toc13184313)

[5. Architecture/Design 7](#_Toc13184314)

[6. Technical Spec – Solution Development Environment 7](#_Toc13184315)

[a. Front End Layer 7](#_Toc13184316)

[b. Middle Tier Layer 7](#_Toc13184317)

[c. Database & Integration Layer 7](#_Toc13184318)

[d. Ancillary Layer 8](#_Toc13184319)

[Controllers can be tested using Postman Tool 8](#_Toc13184320)

[e. Security 8](#_Toc13184321)

[f. Deployment & Infrastructure 8](#_Toc13184322)

[g. Editors 8](#_Toc13184323)

[7. Other Design constraints 8](#_Toc13184324)

[8. Assessment Deliverables 8](#_Toc13184325)

[9. Important Instructions 9](#_Toc13184326)

# Business Requirement (Stock Market Charting)

This Software System lets Admin to upload Stock Price of a Company(which is listed in a Stock Exchange) at different points of time. It need to support multiple Stock Exchanges. And the registered Users should be able to generate various charts to perform Stock Market performance of various Companies or Sectors over certain period of time. More details on the features which need to be supported are specified, below.

This Case study supports below two different Roles:

1. Admin
2. User

## Admin Use Cases:

Admin can perform below operations. It is mandatory to implement all the requirements, except the ones mentioned as optional.

1. **Login/Logout:** Login and Logout. To avoid Complexity, there can be a predefined username and password for Admin
2. **Manage Stock Exchanges:** This feature lists all the Stock Exchanges currently supported. BSE, NSE Stock Exchanges need to be available by default. It should be possible to add new Stock Exchanges. Deletion of Stock Exchange need not be supported.
3. **Manage Company:**
   1. Add a new company details with fields or edit an already existing Company details
   2. Deactivate an already existing company
   3. Update any IPO related data
4. **Import Data(Excel Format):**
   1. Data can be imported(in Excel format), it's basically to feed stock price of a company at various points of time.
   2. Uploaded Excel need to be in a specific format, if not error message need to be displayed. While uploading Excel, specify the Stock Exchange to which the uploaded data belong to. If such company exists.
   3. The company code, date ranges need to be appropriately checked, if any data will be over written.
   4. After successfully imported, data need to get stored in a database and Uploaded Summary need to be displayed like which company, Stock Exchange, how many records imported, from and to date range, etc…
5. **Missing Data:** Able to check the dates for which Stock Price of a Company is not available

## User Use Cases:

A User can perform below operations. It is mandatory to implement all the requirements, except the ones mentioned as optional.

1. **User Signup/Login/Logout:** An User can
   1. Signup for a new Account. When signed up, an email needs to be sent to the User with confirmation link.
   2. Login to an existing and Email confirmed account. User should be able to Login only after E-Mail Confirmation is done.
   3. Logout from an account.
2. User can update profile, password of an already existing Account
3. Able to search for a company to display Company profile & Turn over, CEO, board members, Industry, brief write up, current/latest Stock market price
4. Whenever user requests charts/data for certain period, the period need to be divided into appropriate intervals(Week or Month or Quarter or Year), to display the chart
5. View IPOs planned in a Chronological order
6. When user types in 2 or more characters for a company name or company code, it should display matching company names(using ajax), so that user can select one of them, if required
7. Comparison Charts. It should be possible to perform below comparisons of
   1. a single company over different periods of time
   2. different companies over a specific period
   3. a single sector over different periods of time
   4. different sectors over a specific period
   5. between a Sector and a company over a specific period of time
8. Should be possible to select if comparisons need to be displayed in a single chart
9. Use different colors when multiple Companies/sectors are displayed in a single chart and display legend
10. Should be possible to select Chart type(line chart, bar chart, pie chart, etc…)
11. For a displayed chart, it should be possible to export data and download in Excel
12. Whenever a chart is displayed, display Average, Min, Max, Growth for that specific period
13. Possibility to perform multiple comparisons between Company’s or Sectors, over a period of time.
14. When data does not exist for certain period in between, that need to be appropriately indicated in the chart
15. View future Trend /Basic Prediction for a Company or Sector - Optional

## Company related Data Fields:

1. Company Name
2. Turnover
3. CEO
4. Board of Directors
5. Listed in Stock Exchanges
6. Sector
7. Brief writeup
8. Stock code in each Stock Exchange

## Stock Price details Excel:

1. Company Code
2. Stock Exchange
3. Current Price
4. Date
5. Time

## IPOs planned:

1. id
2. Company Name
3. Stock Exchange
4. Price per share
5. Total number of Shares
6. Open Date Time
7. Remarks

## Sectors data Fields:

1. Id
2. Sector Name
3. Brief

You may consider 3 or 4 sample sectors, as a sample data

## User DB Table:

1. Id
2. Username
3. Password
4. UserType(if Admin or normal User)
5. E-mail
6. Mobile number
7. Confirmed

## Stock Exchange Data Fields:

1. Id
2. Stock Exchange
3. Brief
4. Contact Address
5. Remarks

# Full Stack Technologies

The technologies included in Full Stack are not limited to following but may consist of:

* UI Layer (HTML5, CSS3, Bootstrap 4, JavaScript, Jquery, Angular 4/6)
* Middleware Restful API (Spring Boot Restful & MicroServices, JAX-RS, Spring MVC)
* Database Persistence ( Hibernate)
* Database layer (MySQL, MongoDB)
* Ancillary skills (GIT, Jenkins(CI/CD), Docker, Maven) etc.

To complete this case study, you should be comfortable with basic single page web application concepts including REST and CRUD. You may use angular-cli to create your template project. All web pages need to be responsive.

Ref1: https://cli.angular.io/

Ref2: <https://github.com/angular/angular-cli>

# Microservices Integration and Security

Assuming that you are done with developing individual Microservices in previous Phase, current Phase includes creating and integrating Zuul gateway, Eureka Server and Eureka client in each Microservice. This is shown in architecture Diagram, in next section.

Zuul Gateway(create a Zuul based Project using Spring Initilaizer or STS IDE), add required annotation. Authentication and JWT Token validation can be performed in Zuul’s Pre Filter.

Add below details to yml or property file

1. add route configurations
2. port number & url of eureka Server

Eureka Server(create a Eureka Discovery Server using Spring Initializer or STS IDE), add required annotation & port number in yaml configuration file

Add Eureka Discovery Client to all the Microservice

Now open Eureka Server Dashboard by opening and crosscheck if all Microservices are registered in the dashboard

Now start sending the requests to Zuul Gateway which further routes to a specific Microservice based on the url pattern

Develop code for Unit Testing

PostMan, to test REST end points

# Spring Microservices Tools to be used

As already specified under Full Stack Technologies Microservice Architecture need to be followed. Ensure that the Application is divided into multiple Microservices, along with database/tables each Microservice Manages. Below Spring Microservices Tools need to be used

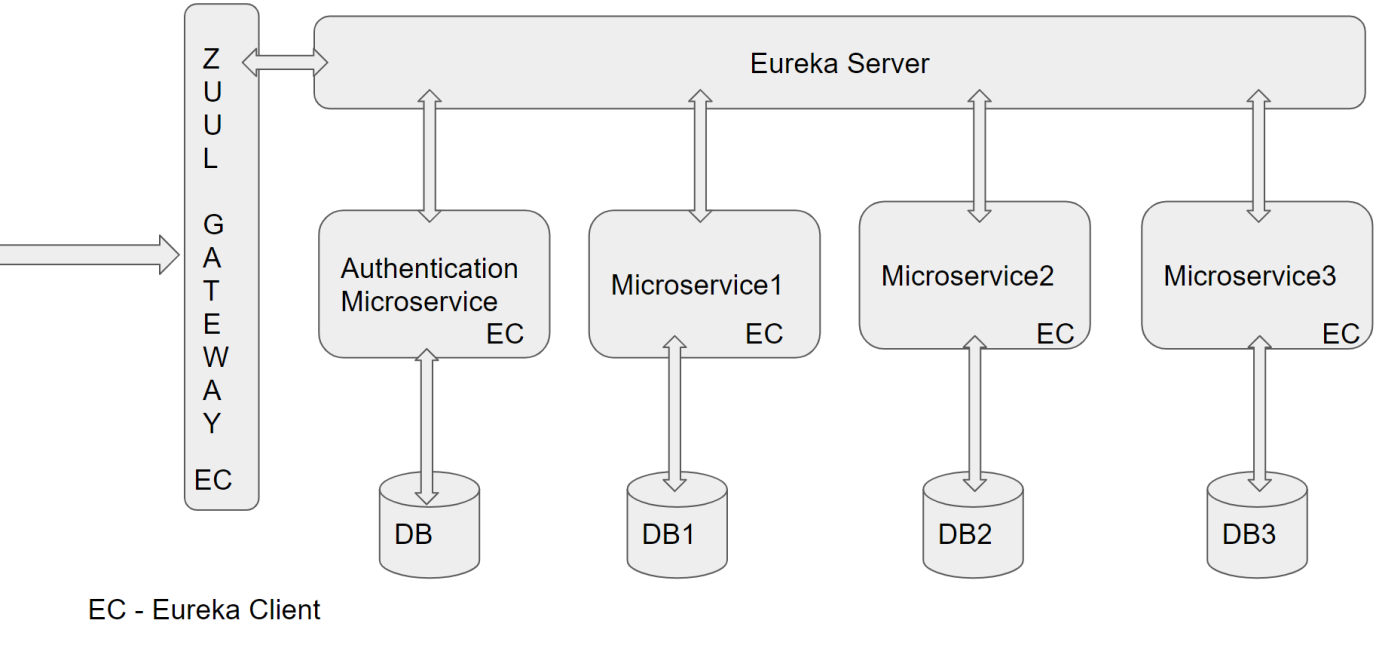
* Zuul API Gateway
* Eureka Service Registry & Discovery
* Ribbon Client side Load Balancer(optional)
* Feign Client
* Hystrix Circuit Breaker & Fault Tolerant Tool(optional)

# JWT Authentication

Create additional Microservice which takes care of authentication and role activities, and JWT Token validation. Spring Security need to be used for Authentication. On successful authentication or token validation the actual request need to be forwarded to the corresponding Microservice. Invoke authentication REST endpoints from Zuul Gateway. Use PreFilter to perform JWT Token validation by invoking REST endpoint of this Microservice.

Instead of JWT, any other security protocol such as OAuth2 can be used. Authentication data can be stored in MySQL DB or LDAP or any other data source.

# Architecture/Design



# Technical Spec – Solution Development Environment

## Front End Layer

|  |  |
| --- | --- |
| **Framework(s)/SDK/Libraries** | **Version** |
| Angular with TypeScript | 4/6 |
| Bootstrap | 3.0 or above |
| CSS | 3 |
| HTML | 5 |
| JavaScript | 1.8 or above |
| JQuery | 1.3 |

## Middle Tier Layer

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Java Stack | Spring Boot | 1.5 or above |
| Spring MVC | 4.0 or above |
| JDK | 1.7 or above |
| Maven | 3.x or above |

## Database & Integration Layer

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Java Stack | Hibernate | 4.0 or above |
| JAX-RS Jersey/ Spring Restful |  |
| MySQL | 5.7.19 |
| MongoDB | MongoDB | 3.4 |
| NoSQL |  |

## Ancillary Layer

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Source Code Management Tool | GIT | 2.14.2 |
| Build Tool/JAVA Stack | Maven | 3.x |
| Testing Tool/JAVA Stack | JUnit/Mockito | 4.x |
| Testing Tool/JAVA Stack | Spring Test | 4.x |

## Controllers can be tested using Postman Tool

## Security

|  |  |
| --- | --- |
| **Name** | **Version** |
| Spring Boot Security |  |
| JWT |  |

## Deployment & Infrastructure

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Docker | - |  |
| Apache Tomcat | - |  |
| Jenkins(CI/CD) | - |  |
| Node | - |  |

## Editors

|  |  |
| --- | --- |
| **Name** | **Version** |
| STS(Spring Tool Suite) |  |
| Visual Studio Code |  |

# 

# Other Design constraints

Below are other Design constraints to be considered

* Integrate with any SMTP Server, to send Emails in appropriate Use cases
* Integrate with any Payment Gateway to process Payments

# Assessment Deliverables

1. Updated Design document by adding Zuul, Eureka into the Architecture
2. Check in below project Zuul Gateway Project, Eureka Server Project, all Microservices with Eureka Client added
3. Screen shots showing sample invocation of Zuul Gateway end points from Post Man Tool.
4. Few Steps on how to run the solution.
5. Test code need to be included

# Important Instructions

1. Above sample Design provided is just for reference, Associates can make changes over it or use their own Design.
2. Please make sure that your code does not have any compilation errors while submitting your case study solution.
3. The final solution should be a zipped code having solution. Solution code will be used to perform Static code evaluation.
4. Implement the code using best design standards/family Design Patterns.
5. Use Internationalization for all the labels and messages in Rest API Development.
6. Do not use System out statements or console.log for logging in Rest API and FrontEnd respectively. Use appropriate logging methods for logging statements/variable/return values.
7. If you are using Spring Restful or Jersey JAX-RS to develop Rest API, then use Maven to build the project and create WAR file.
8. Write web service which takes input and return required details from database.
9. Use JSON format to transfer the results.