



# **Stock Market Charting (FSD) v3.0**

## **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**  
**10/22/2019**

## Table of Contents

1. Business Requirement (Stock Market Charting)	2
1.1. Admin Use Cases:	2
1.2. User Use Cases:	3
1.3. Company related Data Fields:	4
1.4. Stock Price details Excel:	4
1.5. IPOs planned:	4
1.6. Sectors data Fields:	4
1.7. User DB Table:	4
1.8. Stock Exchange Data Fields:	5
2. Design Inputs	5
3. UI/UX Wireframes	5
4. Entity Classes – Mid Tier	9
5. Model Classes	11
6. Architecture Diagram	11
7. Development of individual Microservices	13
8. Database Tables	14
9. Microservices Integration and Security	14
10. Spring Microservices Tools to be used	15
11. JWT Authentication	16
12. Architecture/Design	16
13. DevOps Activity	16
14. Diagram	17
15. Jenkins CI/CD	17
16. Configure Jenkins and Docker for the Project	18
17. Perform CI/CD	18
18. Diagram	19
19. Full Stack Technologies	19
20. Technical Spec – Solution Development Environment	19
20.1. Front End Layer	19
20.2. Middle Tier Layer	20
20.3. Database & Integration Layer	20
20.4. Ancillary Layer	20

20.5.	Security	20
20.6.	Deployment & Infrastructure	20
20.7.	Editors	20
21.	Assessment Deliverables	21
22.	Important Instructions	21

## 1. 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

### 1.1. 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:**
  - a. Add a new company details with fields or edit an already existing Company details
  - b. Deactivate an already existing company
  - c. Update any IPO related data
4. **Import Data(Excel Format):**
  - a. Data can be imported(in Excel format), it's basically to feed stock price of a company at various points of time.  
Below is sample Excel format, which can be used. (Associates can add more fields, if required)  
[https://github.com/vskreddy652/Genc\\_BatchB/blob/master/StockExchange\\_CaseStudy/sample\\_stock\\_data.xlsx](https://github.com/vskreddy652/Genc_BatchB/blob/master/StockExchange_CaseStudy/sample_stock_data.xlsx)

- b. 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.
  - c. The company code, date ranges need to be appropriately checked, if any data will be over written.
  - d. 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

## 1.2. 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
  - a. Signup for a new Account. When signed up, an email needs to be sent to the User with confirmation link.
  - b. Login to an existing and Email confirmed account. User should be able to Login only after E-Mail Confirmation is done.
  - c. 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, sector, 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
  - a. a single company over different periods of time
  - b. different companies over a specific period
  - c. a single sector over different periods of time
  - d. different sectors over a specific period
  - e. 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

### **1.3. 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, about companies Services/Product, etc...
8. Stock code in each Stock Exchange

### **1.4. Stock Price details Excel Many to Many**

1. Company Code – to which Company this Stock Price Info belongs to
2. Stock Exchange – the Stock Price of the Company in this Stock Exchange
3. Current Price – Stock Price
4. Date – Date of the Stock Price
5. Time – Stock Price at this Specific time

### **1.5. IPOs planned: Many to Many**

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

### **1.6. Sectors data Fields:**

1. Id
2. Sector Name
3. Brief

You may consider 3 or 4 sample sectors, as a sample data. For example Finance, Healthcare Services, Pharmaceuticals, Hotels, Internet Software & Services

### **1.7. User DB Table:**

1. Id

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

### **1.8. Stock Exchange Data Fields:**

1. Id
2. Stock Exchange
3. Brief
4. Contact Address
5. Remarks
6. [12:05 PM] Ram R (Naren) (Guest)
7. Company Stock code in each Stock Exchange Companytostockexchange\_mtm
8. Id Company Id
9. Stockexchange Id
10. Stock Code
- 11.
12. [12:06 PM] Ram R (Naren) (Guest)
13. remove the Company Stock code in each Stock Exchange

## **2. Design Inputs**

Next sections in this document provides inputs on designing the solution for above requirements.

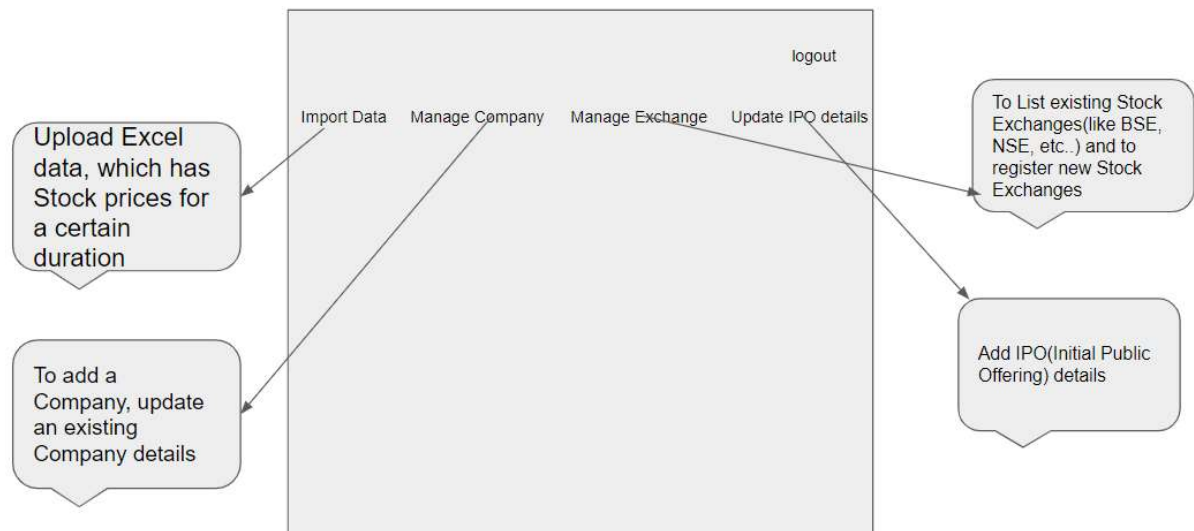
Design inputs provided in this document are just for your reference purpose, Associates can make changes or additions to the Design, based on their analysis.

## **3. UI/UX Wireframes**

In this Phase you will develop Angular Client Application, with responsive UI using HTML5, HTML5 API, CSS3, Bootstrap/Material and Angular.

Below are some sample Wireframes

## Admin Landing Page



## Admin - Upload Excel to import Data

Admin - Upload Excel to import Data

logout

Import Data   Manage Company   Manage Exchange   Update IPO details

**Import Excel**

Select Excel file to be uploaded

[Click Here to download sample Excel file](#)

Admin - Upload Excel to import Data

logout

Import Data   Manage Company   Manage Exchange   Update IPO details

**Summary of Upload**



Company Name	Abc LTD
Stock Exchange	BSE(Bombay Stock Exchange)
No. of Records Imported	80
From Date	To Date

## Manage Companies

logout

Import Data   Manage Company   Manage Exchange   Update IPO details

**List of Companies**

	Company1	BSE, NSE	<Brief Writeup>	<input type="button" value="Edit"/>
	Company2	BSE	<Brief Writeup>	<input type="button" value="Edit"/>

<< other Companies >>

If you have too many companies, you may support searching companies based on search Pattern (optional functionality)

## Create New Company

logout

Import Data   Manage Company   Manage Exchange   Update IPO details

**Create New Company**

Company Name

CEO Name & Board Members

TurnOver

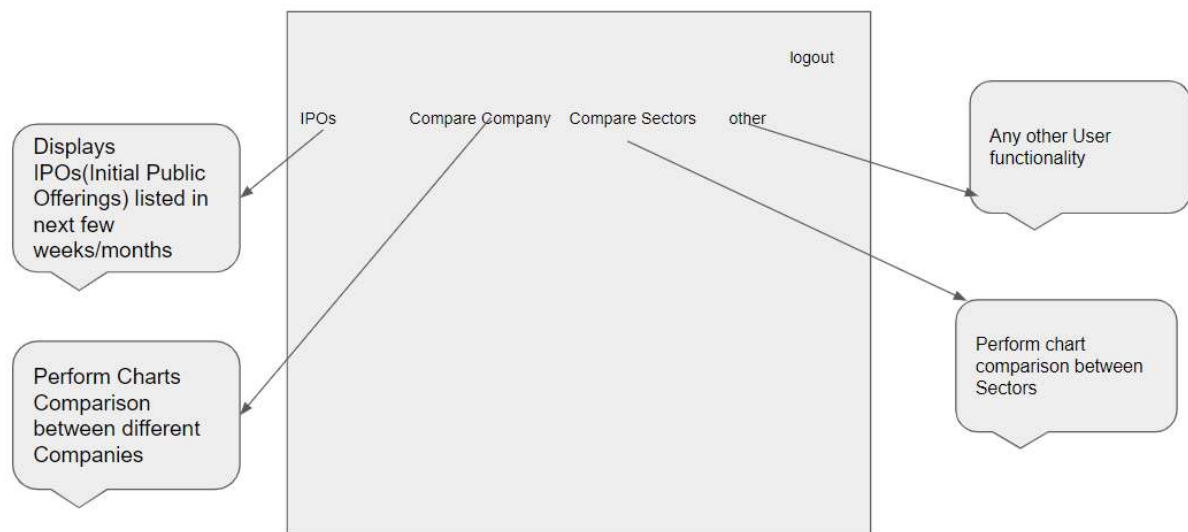
Brief Description

IPO date

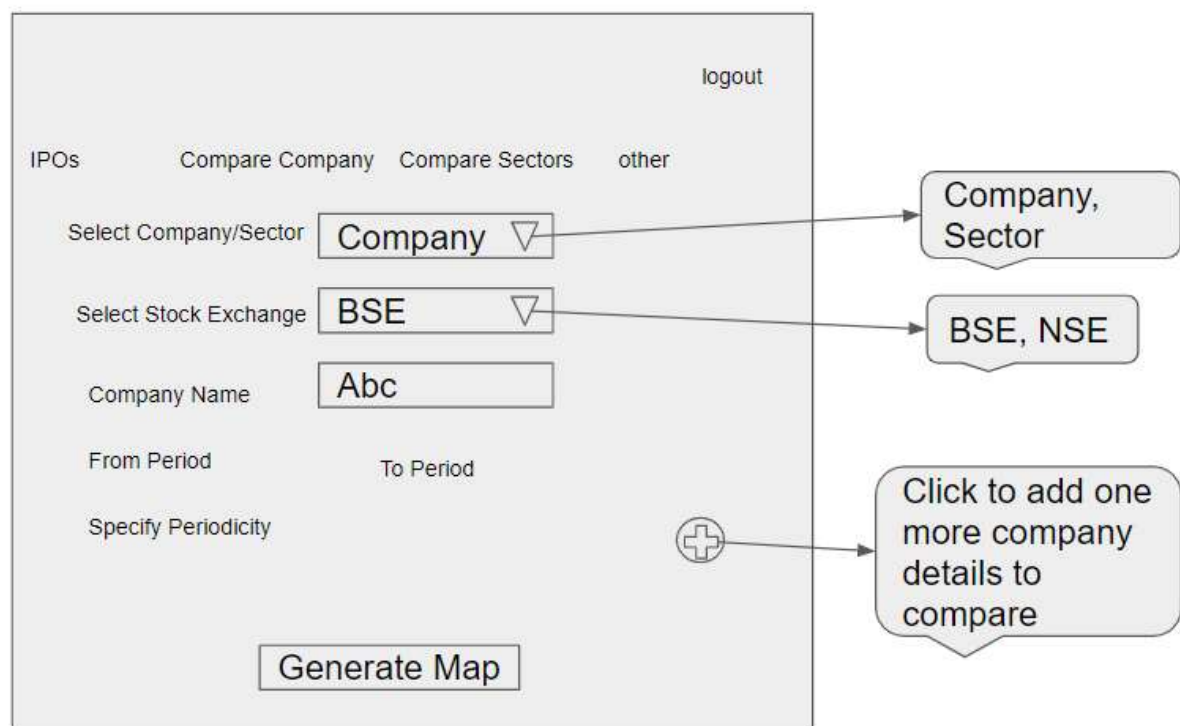
... any other fields ....



## User Landing Page



## Comparison Charts



Below are some of the wireframes

1. Login/Signup for User/Admin
2. Display Comparison Charts
3. Admin – List existing Stock Exchanges

4. Admin – Add Stock Exchange
5. Admin – List/Add IPO Details
6. User – View IPO Details
7. User – Update Profile, Update Password

This Phase includes development of

1. Angular components with Templates
2. Angular Routing
3. Angular Services, to invoke REST End points.

Based on the requirements, Front End need to be divided into multiple components to accommodate above Wireframes. Angular Routing can be used to create navigation Links. For Authentication, store JWT token in Local or Session Storage. REST APIs are invoked from the corresponding Services,

Below are some of the Angular Components

1. ImportExcel
2. ManageCompanies
3. ManageIPO
4. ManageStockExchanges
5. CreateCompany
6. ComparisonCharts
7. UserLogin – Login for both user and Admin
8. UserSignup

Above Angular Components is just a sample list, more additional components need to be identified and developed.

Angular Routing feature is used to display Navigation links on Web Pages. When User clicks a Navigation link, the corresponding Component is displayed.

As known, Angular Services is used to interact with the REST end points, using Observable or Promises.

Fusion Charts library can be used to display Charts.

As known JWT token is generated on the Server side and received by Angular Client on successful authentication. Angular's HttpInterceptor can be used to automatically send JWT Token thru Header of every HttpRequest.

## 4. Entity Classes – Mid Tier

Below are the activities which need to be performed as part of this

1. Identify all Entity Classes. An Entity class is the one which is mapped to underlying DB Table
2. Identify relationship(such as One to One, One to Many, Many to One, Many to Many) between Entity classes, if required

3. Develop the source code of Entity classes

Below are sample Entity Classes

**CompanyEntity Class:** Below can be fields in Company Entity Class

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

Snapshot of Entity class below

```
import javax.persistence.Column;

@Entity
@Table(name = "company")
public class CompanyEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @Column(name = "cname")
    private String company_name;

    @Column(name = "turnover")
    private float turnover;

    @Column(name = "ceo")
    private String ceo;
    //...more columns...

    //constructor(s)
    //setter & getter methods
}
```

**StockPriceEntity Class:** StockPrice Entity class represents Stock Price of a company at a specific point of time. Below are the fields

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

**IPODetailEntity Class:** Indicates IPO details of a specific Company

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

**UserEntity class:** User login details

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

**StockExchangeEntity class:**

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

Companytostockexchange\_mtm

1. Id
2. Company Id
3. Stockexchange Id
4. Stock –Company Code

## 5. Model Classes

Model Classes are the classes which are required to transfer the data between

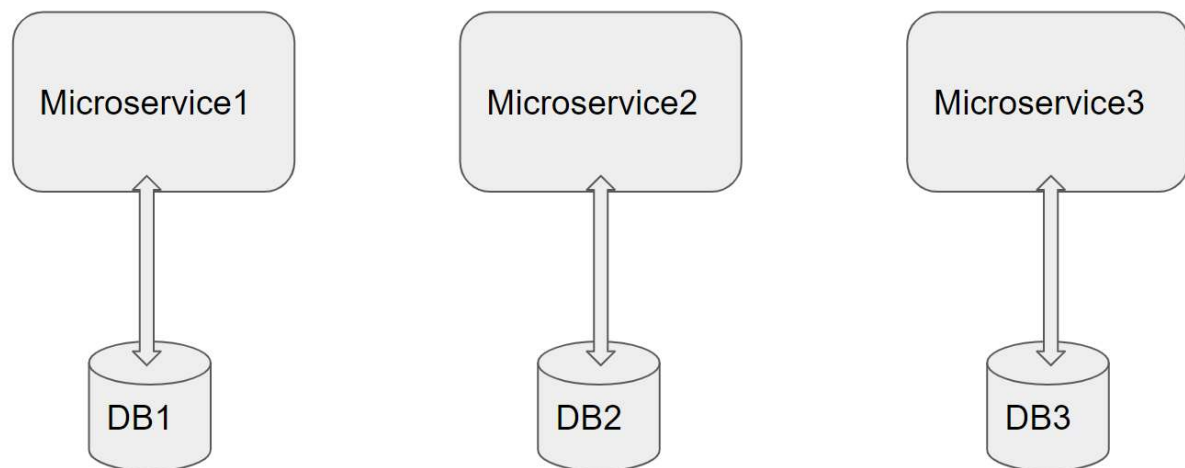
1. REST APIs and Angular Client,
2. REST Controller and Service Layer
3. Service Layer and Repository Layer

As part of this Phase identify all Model classes, and develop source code for the same.

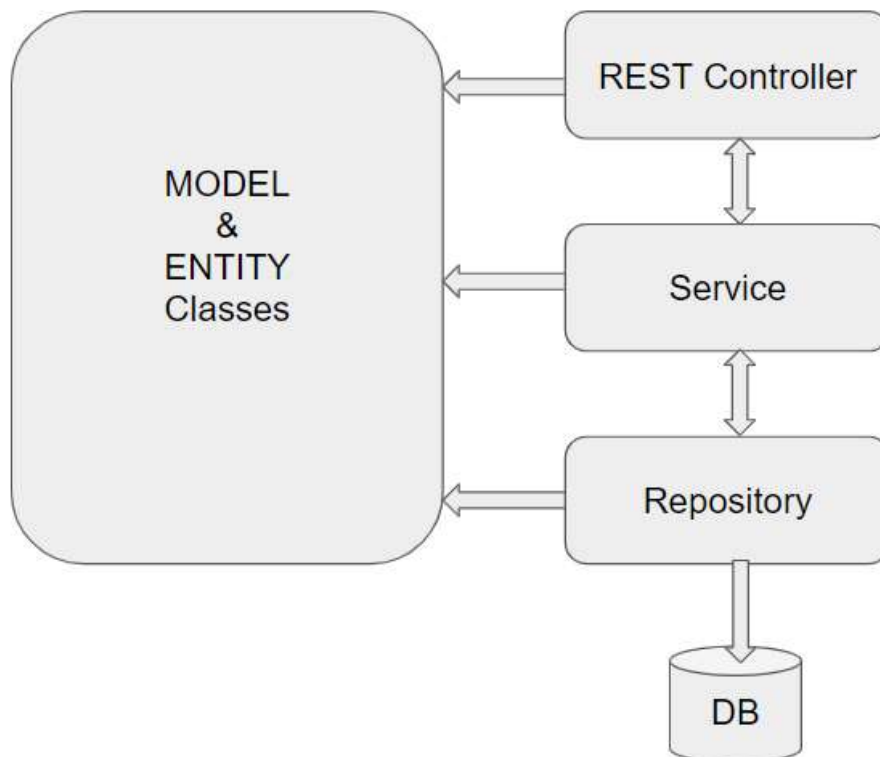
Model classes are just normal POJO classes with data members, constructors, setter/getter methods

## 6. Architecture Diagram

Class diagram



Architecture of a Single Microservice with REST Controller, Service, Model & Entity Classes and Repository classes



## 7. Development of individual Microservices

This specific Phase is to design/develop individual Microservices. Analyze the requirement and divide back end functionality into multiple Microservices. Based on the Mentor on Demand requirements, below can be possible Microservices

1. Microservice to upload excel. All required Error Handling including below cases need to be implemented  
uploadExcel
2. Microservice to retrieve data of a company, for certain period,  
getCompanyStockPrice i/p Company ID, From Period, To Period, periodicity  
getMatchingCompanies – used to retrieve list of Companies based on pattern matching of Company Name  
getCompanyDetails – Basic information about company  
getCompanyIPODetails – IPODetails of Company

3. Microservice to retrieve data of a sector, for certain period: Calculations need to be performed  
 getList of Companies in a Sector  
 getSectorPrice Sector ID, From Period, To period, periodicity
4. Microservice for User Login, Signup  
 User Login  
 Admin Login  
 User Signup
5. StockExchange  
 getStockExchangesList  
 addStockExchange  
 getCompaniesList – in a specific Stock Exchange

Each of above Microservice need to comprise below functionality, which need to be developed

1. Each Microservice is a Spring Boot Rest application by specifying required spring boot starter packages in pom.xml or by using Spring Initializr
2. REST Controllers, with the appropriate REST End points to perform corresponding CRUD operations. Along with End Points which are exposed to Angular Client, you may need additional End point(s) for interaction between Microservices
3. As known, each Microservice is a self-sufficient and standalone application, and owns data stored in specific DB tables or databases.
4. Services – Service Layer
5. Entity & Model classes, including appropriate relationship (like One-One, Many-One, etc...) between Entity Classes, if required. (Entity and Model classes have been developed in the Previous Phases)
6. In case specific Entity or Model classes are required across multiple Microservices, it is recommended to maintain separate copy of Entity or Model classes for each Microservice.
7. Microservice interaction with corresponding DB tables or Databases it owns.
8. It is possible that one Microservice need to interact with other Microservice(using RestTemplate or FeignClient)
9. Repository class which implements JPA or CrudRepository, if RDBMS is used

10. Usage of Custom Queries in JPA or CrudRepository using @Query where ever custom functionality required
  11. Feign Client or restTemplate can be used to invoke one Microservice, from another Microservice
  12. Use Postman to test the Microservices by directly passing requests to each REST end Point, of each Microservice
  13. Unit Testing code can be developed using JUnit, Mockito, and perform Unit Testing
- 

## 8. Database Tables

Below are list of Database Tables, for actual fields refer corresponding Entity classes. Though, ideally each Microservice need to use separate database, it should be fine to place all below DB Tables in a single database

Table Name	Purpose
Company	Stores list of Company details
StockPrice	Stores the Stock prices of all companies, which are imported thru Excel Sheet
IPODetails	Stores the IPO details of companies
Sectors	Stores the list of Sectors(like Health Care, banking, etc...) supported
Users	Stores profile & login details of Admin and normal Users
StockExchange	Stores the list of Stock Exchanges supported(like

Refer Entity classes to identify Columns in each of the DB Table.

## 9. 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

Use PostMan, to test REST end points

---

## 10. Spring Microservices Tools to be used

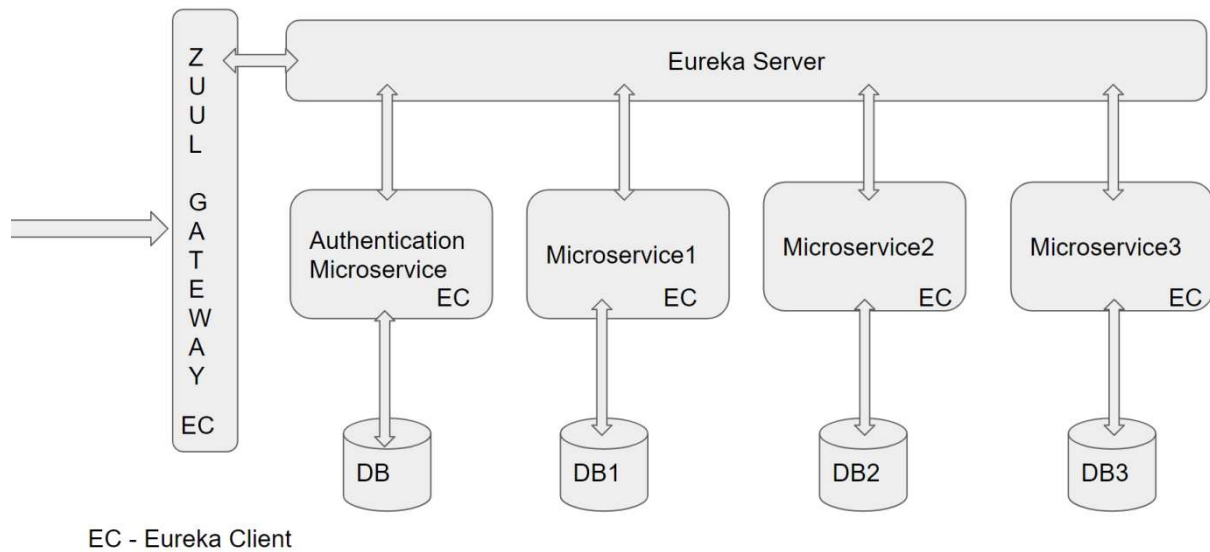
As already specified under Full Stack Technologies Microservice Architecture need to be followed. Ensure that the Application is divided into multiple Microservices, along with database/tables each Microservice Manages. Below Spring Microservices Tools need to be used

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

## 11. 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.

## 12. Architecture/Design



## 13. DevOps Activity

This phase includes performing below Activities

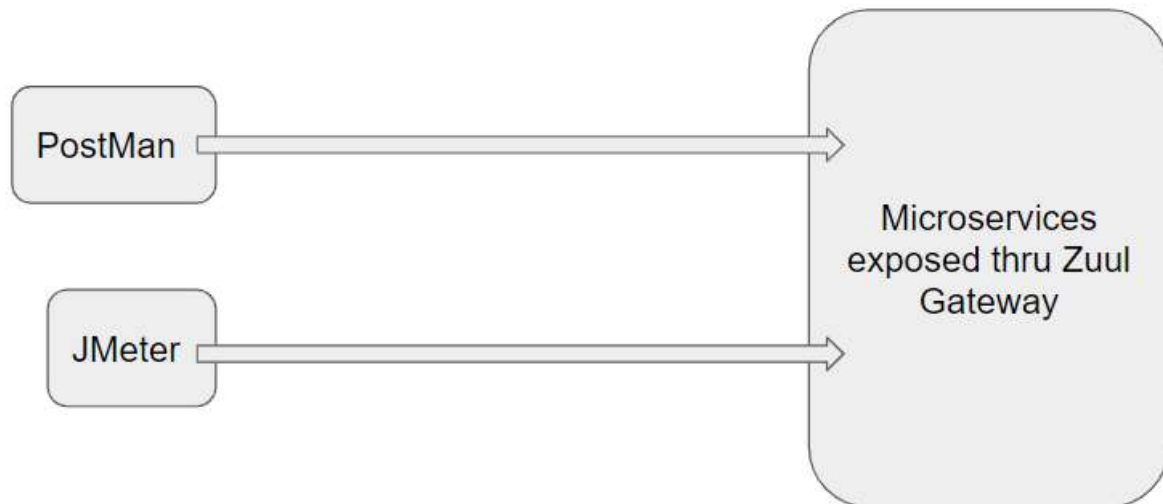
**Dockerization:** Dockerize atleast Front End or any one Microservice of Mid Tier. Provide Dockerfile and the docker commands used to create image and run Container. Share Screen shots of running Docker.

To Setup Docker Client on your VM please refer [https://github.com/vskreddy652/Genc\\_BatchB/blob/master/Docker%20Remote%20Host%20Access%20Steps%20\(3\).docx](https://github.com/vskreddy652/Genc_BatchB/blob/master/Docker%20Remote%20Host%20Access%20Steps%20(3).docx)

**JMeter:** As already known JMeter is used to perform Performance or Load Testing. Create a JMeter Test Case, which invokes a REST End point, with multiple threads. Check in jmx file and share the report generated after Performance Testing. Repeat this for atleast two REST end points.

**Code Coverage:** Code coverage is a Quality Metric to check if sufficient number of Test Cases are created. EclEmma tool can be used as Code Coverage Tool. Code Coverage can be performed on any one Microservice. Ensure that Code Coverage need to be atleast 80%

## 14. Diagram



## 15. Jenkins CI/CD

**Jenkins CI/CD:** As already known Jenkins is popular tool to perform CI/CD. When the code is pushed to GIT, build need to be automatically created and deployed. If possible create a Docker image and run the Container on Docker Host

This Phase also includes completion of Integration of Front end with Mid Tier.

**Deployment on Cloud(optional):** Any of the Microservices or Front End can be deployed on any Cloud(AWS, Azure, etc...) of your choice. Heroku

## 16. Configure Jenkins and Docker for the Project

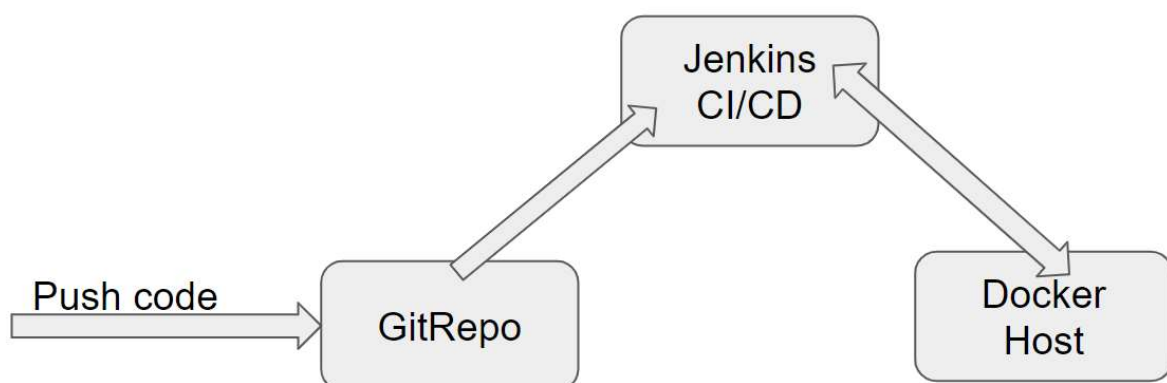
- Import the project (as discussed above) in Spring Tool Suite and configure it locally to run it as Spring Boot App.
- You may need to configure MySQL credentials and database name.

- Execute the project locally and access the app at <http://localhost:portnumber>
- Once, it is working fine in local development environment; Configure CI/CD in Jenkins, along with Dockerization
- Push the app source in internal GIT server. Pl. ask your mentor for the Internal GIT server URL.
- Configure Jenkins locally to pull the source from internal GIT repository
- Jenkins should build the project and create the deployable (war/jar). It should run the unit tests created in "Maven, GIT, Junit, Tomcat Micro Layer for the Project"
- From Jenkins, invoke Docker commands to perform, below
- Creation of Docker Image(docker build . )
- Create and run Docker Container(docker run <image\_id>)

## 17. Perform CI/CD

1. Make few changes in the project (source code)
2. Make it sure that project is running locally in development environment without errors.
3. If it running locally without errors, push the changes to the internal GIT repository which was connected
4. If Project was Setup properly, Jenkins will automatically pull the code updates from internal GIT repo and build and deploy the project with updated code.
5. Now, when you visit <http://localhost>; you should see the changes in the browser window

## 18. Diagram



## 19. 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>

## 20. Technical Spec – Solution Development Environment

### 20.1. Front End Layer

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

### 20.2. Middle Tier Layer

Technology	Framework(s)/SDK/Libraries	Version
Java Stack	Spring Boot	2.x
	Spring MVC	4.0 or above
	JDK	1.7 or above
	Maven	3.x or above

### 20.3. 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	

## 20.4. Ancillary Layer

Technology	Framework(s)/SDK/Libraries	Version
Source Code Management Tool	<b>GIT</b>	2.14.2
Build Tool/JAVA Stack	<b>Maven</b>	3.x
Testing Tool/JAVA Stack	<b>JUnit/Mockito</b>	4.x
Testing Tool/JAVA Stack	<b>Spring Test</b>	4.x
Controllers can be tested using Postman Tool		

## 20.5. Security

Name	Version
<b>Spring Boot Security</b>	
<b>JWT</b>	

## 20.6. Deployment & Infrastructure

Technology	Framework(s)/SDK/Libraries	Version
<b>Docker</b>	-	
<b>Apache Tomcat</b>	-	
<b>Jenkins(CI/CD)</b>	-	
<b>Node</b>	-	

## 20.7. Editors

Name	Version
STS(Spring Tool Suite)	
Visual Studio Code	
<b>Eclipse IDE for Enterprise Java Developers.</b>	

Agile/Scrum Software development Model can be used

## 21. Assessment Deliverables

Below deliverables need to be checked in(to internal GIT or github)

1. FrontEnd Source code
2. Mid Tier Source code of all Microservices
3. Screen shots of Usage of Post Man tool to test each End Point of all Microservices
4. Few Steps on how to run the solution.
5. Test code of Angular and Mid Tier need to be included
6. Jmeter's JMX file to test atleast one REST End point, and Screenshot of report
7. Dockerfile

8. Jenkinsfile or Jenkins UI ScreenShot
9. URL where the Project is deployed

## 22. Important Instructions

1. Consider using below Java features
  - a. Lambda Expressions
  - b. Collection Streams
  - c. Generics
2. Sample Design provided is just for reference, Associates can make changes over it or follow their own Design.
3. Based on your current work, alternate Technologies can be used, for example ReactJS instead of Angular, etc..., however prior approval from the Mentor is required.
4. Please make sure that your code does not have any compilation errors while submitting your case study solution.
5. The final solution should be a zipped code having solution. Solution code will be used to perform Static code evaluation.
6. Implement the code using best design standards/family Design Patterns.
7. Use Internationalization for all the labels and messages in Rest API Development.
8. 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.
9. 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.
10. Write web service which takes input and return required details from database.
11. Use JSON format to transfer the results.