IIHT

Time To Complete: 10 to 12 hr

E-Stock Market

Contents

[1 Problem Statement 2](#_Toc68289351)

[2 PROPOSED E-STOCK MARKET WIREFRAME 3](#_Toc68289352)

[3 Application Architecture 4](#_Toc68289353)

[3.1 Microservice Architecture (Governance and Tooling/ Database and Storage): 4](#_Toc68289354)

[3.2 Microservice Architecture (Governance and Tooling/ Compute and Integration): 4](#_Toc68289355)

[3.3 Microservice Architecture (Compute and Integration/Presentation/Networking and Content Delivery): 5](#_Toc68289356)

[4 Cloud Architecture 6](#_Toc68289357)

[5 Tool Chain 7](#_Toc68289358)

[6 Business Requirements: 8](#_Toc68289359)

[7 Proposed Rest Endpoints to be exposed 9](#_Toc68289360)

[7.1 Rest APIs: 9](#_Toc68289361)

[8 Rubrics/Expected Deliverables 10](#_Toc68289362)

[8.1 Engineering Concepts (Compute & Integration): 10](#_Toc68289363)

[8.2 Engineering Concepts (Security & Identity): 10](#_Toc68289364)

[8.3 Products & Framework (Database & Storage): 10](#_Toc68289365)

[8.4 Products & Framework (Governance & tooling): 10](#_Toc68289366)

[8.5 Debugging & Troubleshooting 11](#_Toc68289367)

[8.6 Code Quality/Optimizations 11](#_Toc68289368)

[9 Platform 11](#_Toc68289369)

[9.1 Compute 11](#_Toc68289370)

[9.2 Compute, Identity & Compliance, Security& Content Delivery 11](#_Toc68289371)

[10 Methodology 12](#_Toc68289372)

[10.1 Agile 12](#_Toc68289373)

# Problem Statement

**E-StockMarket** Application is a Restful Microservice application, where it allows users to manage the stocks like create, view stock price details and company details.

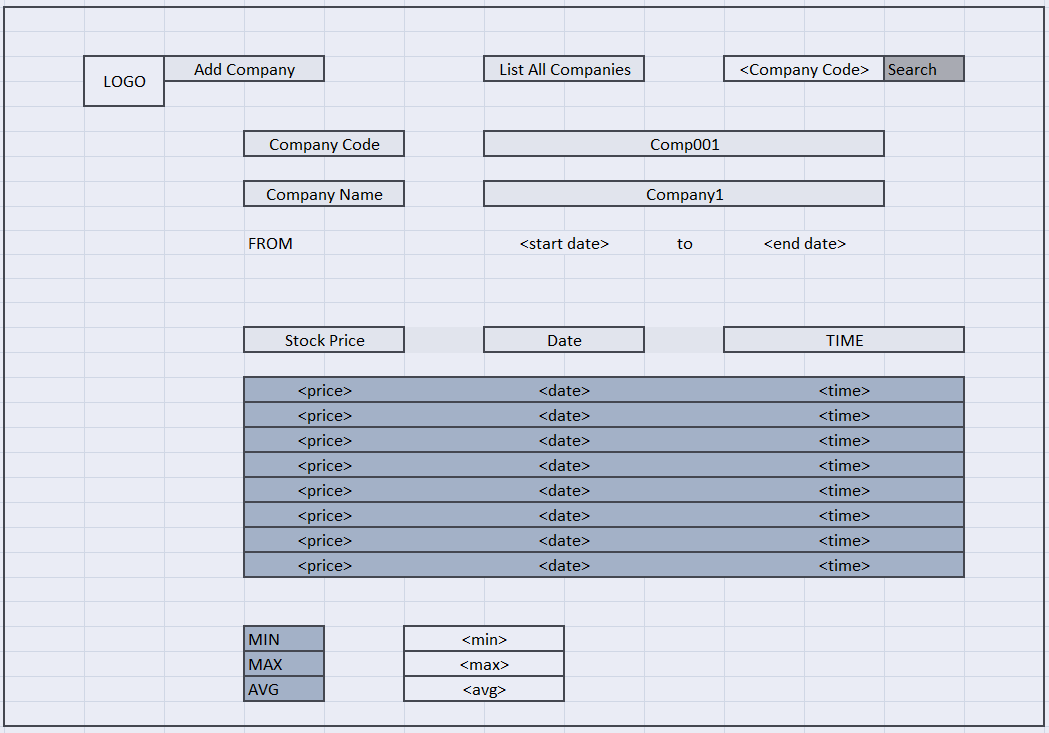
The core modules of E-Stock Market app are:

* Allows to add a new company details and a new stock price detail
* Allows to delete an existing company with all stock details
* Allows to search the company or stock based on company code
* Allows to display all company information or all stock detail
* Allows displaying max, min and average stock prices between the stipulated time periods. For this purpose, there should be a frontend SPA created using Angular

The scope includes developing the application using tool chain mentioned below.

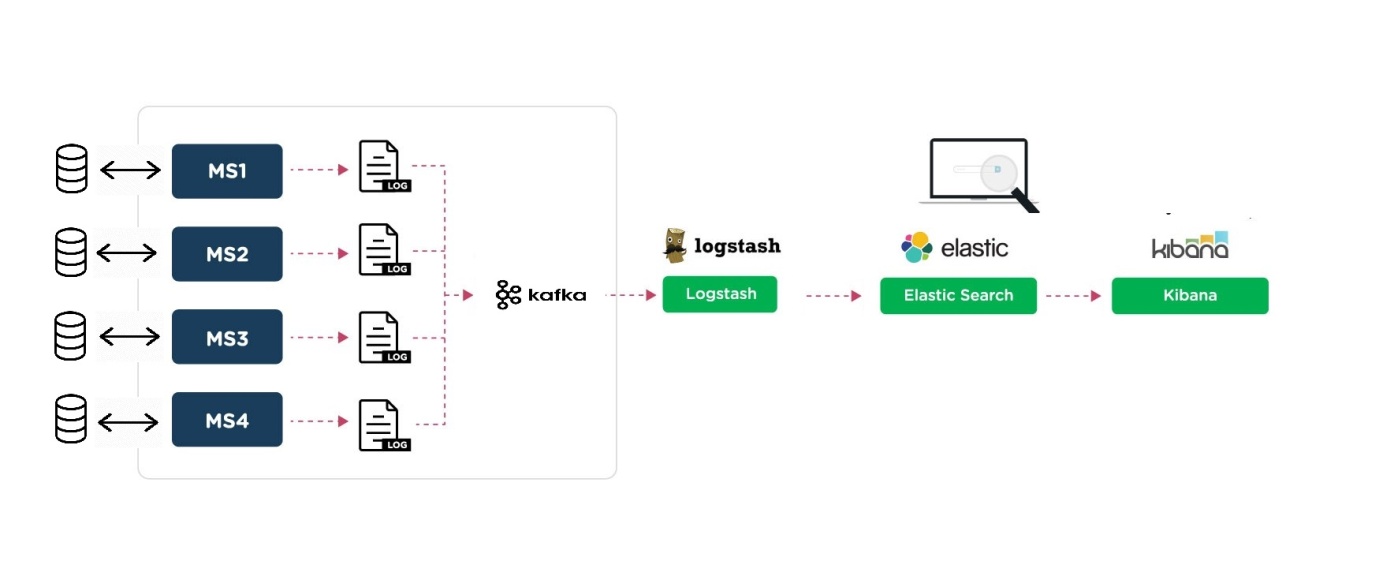
# PROPOSED E-STOCK MARKET WIREFRAME

UI needs improvisation and modification as per given use case.

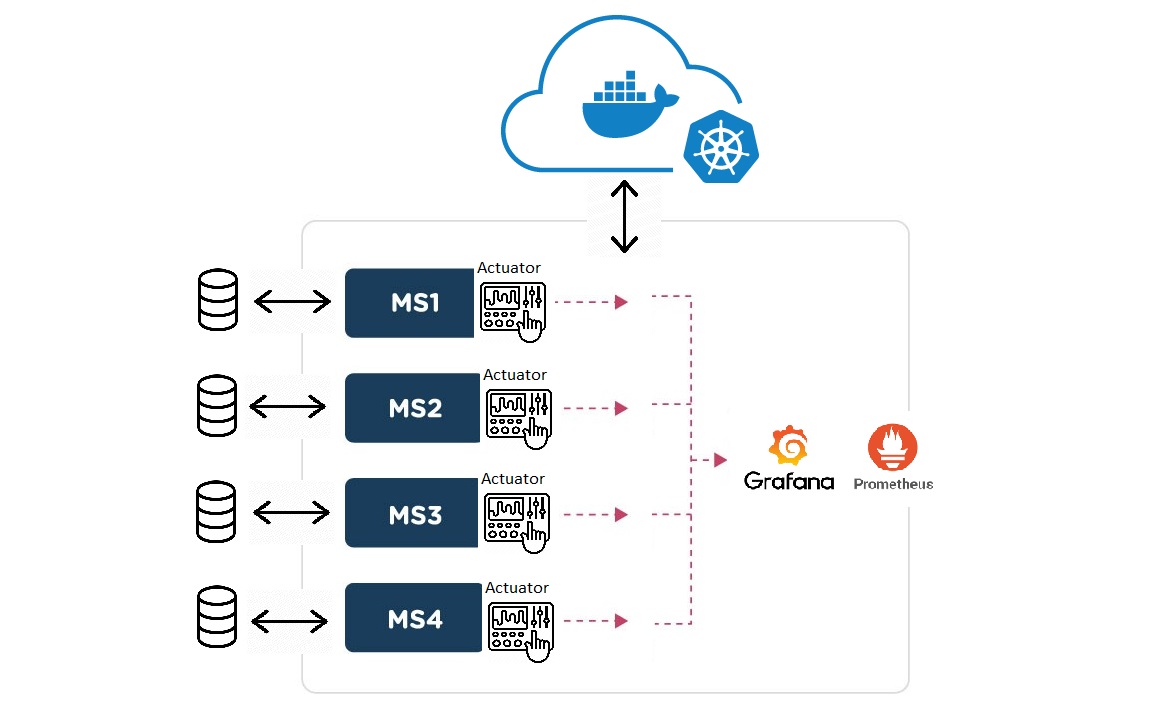


# Application Architecture

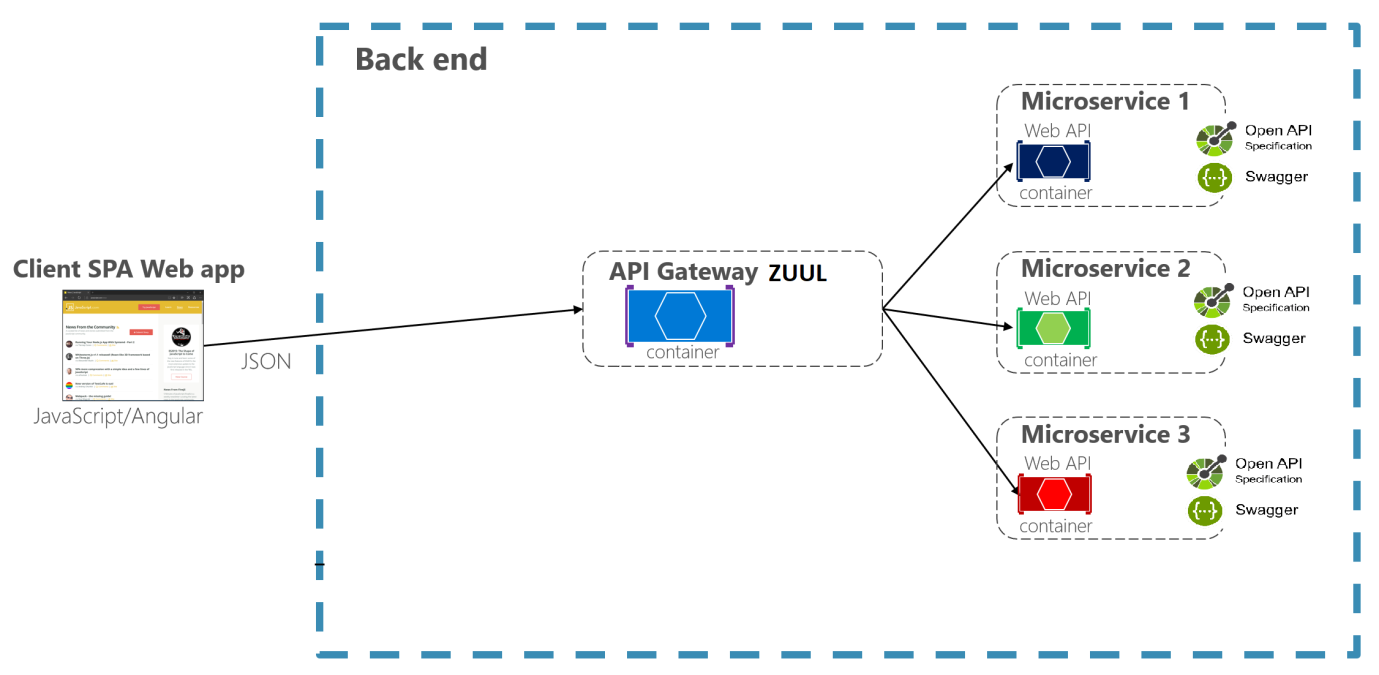
## Microservice Architecture (Governance and Tooling/ Database and Storage):



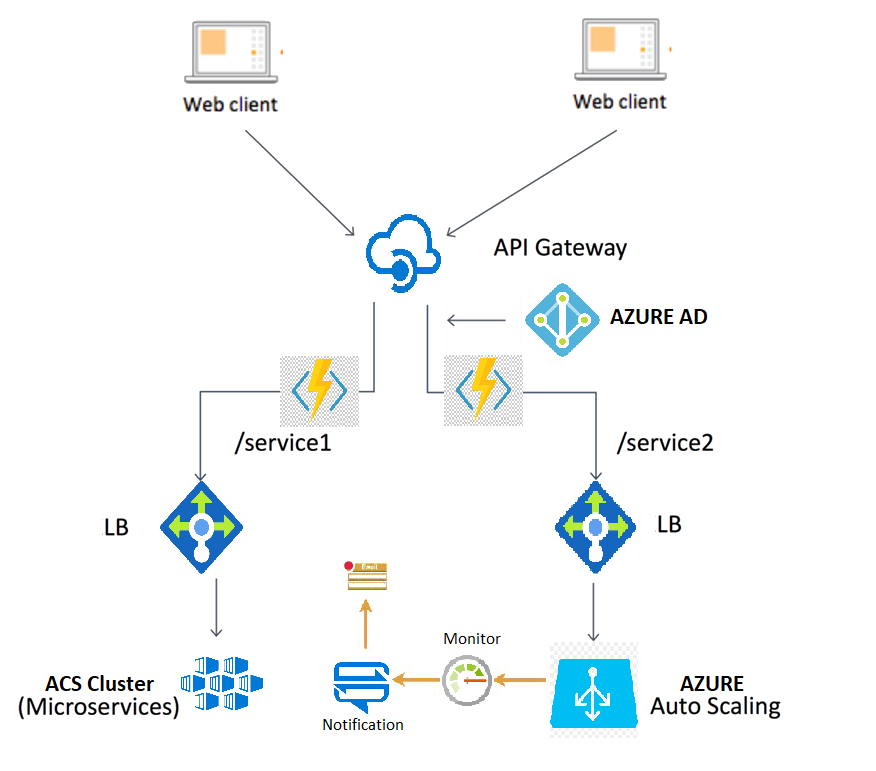
## Microservice Architecture (Governance and Tooling/ Compute and Integration):



## Microservice Architecture (Compute and Integration/Presentation/Networking and Content Delivery):



# Cloud Architecture



# Tool Chain

|  |  |  |
| --- | --- | --- |
| Competency | Skill | Skill Detail |
| Engineering Mindset | Networking and Content Delivery |  |
|  | Ways of Working |  |
|  | Consulting Mindset |  |
|  | DevOps |  |
| Programming Languages | Application Language | Java |
| Products & Frameworks | Presentation | Angular |
|  |  | Javascript/Typescript |
|  | Networking and Content Delivery | Zuul |
|  | Security and Identity | OpenIAM |
|  | Compute & Integration | Spring Boot |
|  |  | ELK Stack |
|  |  | Docker |
|  |  | Kubernetes |
|  | Database & Storage | MongoDB |
|  |  | MySQL |
|  | Governance & Tooling | Git |
|  |  | Junit |
|  |  | Mockito |
| Engineering Quality | Secure Coding/ Code Quality | Veracode |
|  |  | PMD/SonarQube |
| Platform | Cloud Tools | AzureACS (Container Services) |
|  |  | Azure CosmosDB/SQL DB |
|  |  | Azure Redis Cache/Storage |
|  |  | Azure DevOps/Pipeline |
|  |  | Azure API Gateway |
|  |  | Azure Load Balancer |
|  |  | Azure Notification Hubs |
|  |  | Azure Functions |

# Business Requirements:

As an application developer, develop microservices with below guidelines:

|  |  |  |  |
| --- | --- | --- | --- |
| User  Story # | User Story Name | User Story | Development |
| US\_01 | Company Registration | As a user I am be able to register a new company in E-Stock Market application  Acceptance criteria:   1. As a user I am be able to furnish following details at the time of company registration    1. Company Code    2. Company Name    3. Company CEO    4. Company Turnover    5. Company Website    6. Stock Exchange it is enlisted in (BSE, NSE etc) 2. All details fields are be mandatory 3. Company Code must be unique 4. Company Turnover must be greater than 10Cr. | Only API to be developed |
| US\_02 | Add Stock Price | As a user I am be able to add stock price for any company  Acceptance criteria:   * Stock price must be a fractional value. * Date & Time of the stock price is automatically fetched from server system | Only API to be developed |
| US\_03 | View and delete Company Details | As a user I am able to view and delete details of company.  Acceptance criteria:   * If a single company, details are demanded it show complete details of company along with latest stock price. * If all company details are demanded it t show list of all companies with their latest stock price * If company is deleted, it also deletes all stock price data associated with the company. | Only API to be developed |
| US\_04 | View Stock Details | As a user I am able to view stock details of company  Acceptance criteria:  List of stock price of a company is viewed based on start date and end date.  Along with listing the stock price, application display max, min and average stock price between mentioned time frame  UI should provide a drop-down to select company  UI should provide options to select start date and end date  UI should then show the stock details as table along with max, min and average stock price between mentioned time frame | API and Frontend to be developed |

# Proposed Rest Endpoints to be exposed

## Rest APIs:

|  |  |  |
| --- | --- | --- |
| **POST** | **/api/v1.0/market/company/register** | **Register a new company** |
| **GET** | **/api/v1.0/market/company/info/<companycode>** | **Fetches the Company Details** |
| **GET** | **/api/v1.0/market/company/getall** | **Fetches all the Company Details** |
| **DELETE** | **/api/v1.0/market/company/delete/<companycode>** | **Deletes a company** |
| **POST** | **/api/v1.0/market/stock/add/<companycode>** | **Add new stock price** |
| **GET** | **/api/v/1.0/market/stock/get/<companycode>/<startdate>/<enddate>** | **Fetches Stock Price List** |

Use Creational design pattern for composing the model object to be sent back as response on following end-point:

* 1. **/api/v/1.0/market/stock/get/<companycode>/<startdate>/<enddate>**

(Fetches stock price list)

# Rubrics/Expected Deliverables

## Engineering Concepts (Compute & Integration):

1. As an application developer:
   1. Develop the application as a microservice architecture.
   2. Implementation as follows:
      1. Use Domain Driven Design to implement distributed architecture
      2. Follow the Single Data Store per microservice practice
      3. Document REST endpoints with OpenAPI or Swagger
      4. Add CQRS pattern for Event Sourcing for saving and retrieving stock details
      5. Expose all rest Endpoints using a common API Gateway Zuul

## Engineering Concepts (Security & Identity):

* + - 1. As an Application Developer:
         1. Restrict the access over all write operation (secured operations) by adding authentication
         2. Secure all Rest EndPoints by configuring SSL Certificate for Cloud
         3. Use OpenIAM to add OTP based two factor authentication for secured operations

## Products & Framework (Database & Storage):

1. As an application developer:
   1. Implement ORM with Spring Data MongoRepository and MongoDB. For complex and custom queries, create custom methods and use @Query, Aggregations (AggregationOperation, MatchOperation, AggregationResults), implementation of MongoTemplateetc as necessary.
   2. Use MySQL for maintaining data for atleast one of the microservice
   3. Introduce a backup mechanism, such that when record count crosses 10,000 rows, a backup should trigger

## Products & Framework (Governance & tooling):

1. As an application developer
   1. Containerize the complete application, using docker and Dockerfile.
   2. Orchestrate container instances using kubernetes
   3. Use .dockerignore as necessary to avoid containerizing un-necessary packages.
   4. Integrate Spring Boot Actuator with Prometheus/Grafana to monitor middleware.
   5. Assemble logs from all microservice through kafka in Logstash.
   6. Pass on logstash data to ElasticSearch and organize them index. One index for each microservice.
   7. Pass this indexed data from ElasticSearch to Kibana and generate charts for each microservice logs

## Debugging & Troubleshooting

1. Generate bug report & error logs - Report must be linked with final deliverables which should also suggest the resolution for the encountered bugs and errors.

## Code Quality/Optimizations

1. Optimize MongoDB implementations using index search and then report the updated response time metrics.
2. Use Veracode to scan the backend application for security vulnerabilities, and submit the report of the same
3. Associates should have written clean code that is readable
4. Associate should have used the Code Analyzer (PMD/SonarQube) to ensure code quality and standard code style.

# Platform

## Compute

Use User Story-4 from the Business Requirements to implement the below.

* + - 1. Use ACS CLI (as an alternative to Azure Management Console) for container management and deployment of microservices. You should be able to explain and demonstrate the same in interview.
      2. Use NoSQL instance of Azure CosmosDB/(SQL) as a database for the Market Application

## Compute, Identity & Compliance, Security& Content Delivery

1. Use Azure Functions and Azure CosmosDB to build a backend process for handling requests for Market App.
2. Orchestrate containers with ACS to run the Microservices
3. Use Azure API Gateway to expose the built Azure functions to be accessible on public internet.
4. Use Azure LB to configure the load balancing of the instances.
5. Configure Azure Notifications to issue messages whenever Auto Scaling is launching or terminating the Container Instances in the Auto Scaling group.
6. Use Azure Security Centre for authentication.
7. Use Azure AD to Sign Up the user for secured operations.
8. Configure Azure Monitor to receive, maintain and expose metrics of all the write operations (Adding company and new stock price details)

Note : Minimum 3 API’s to be hosted in cloud

# Methodology

## Agile

1. As an application developer, use project management tool along to update progress as you start implementing solution.
2. As an application developer, the scope of discussion with mentor is limited to:
   1. Q/A
   2. New Ideas, New feature implementations and estimation.
   3. Any development related challenges
   4. Skill Gaps
   5. Any other pointers key to UI/UX and Middleware Development