IIHT

Time To Complete: 10 to 12 hr

E-Stock Application

Contents

[1 Problem Statement 2](#_Toc76387085)

[2 Proposed E-Stock Application Wireframe 2](#_Toc76387086)

[3 Tool Chain 3](#_Toc76387087)

[4 Business Requirements: 4](#_Toc76387088)

[5 Proposed Rest Endpoints to be exposed 7](#_Toc76387089)

[5.1 Rest APIs: 7](#_Toc76387090)

[6 Rubrics/Expected Deliverables 7](#_Toc76387091)

[7 Implementation/Functional Requirements 8](#_Toc76387092)

[7.1 Product and Frameworks: 8](#_Toc76387093)

[7.2 Governance and Tooling: 9](#_Toc76387094)

[7.3 Code Quality/Optimizations 9](#_Toc76387095)

[8 Platform 9](#_Toc76387096)

[8.1 Cloud Specific Design 9](#_Toc76387097)

[8.2 Design Specification - 1 9](#_Toc76387098)

[8.3 Design specification - 2 9](#_Toc76387099)

[8.4 Design specification – 3 10](#_Toc76387100)

[9 Methodology 10](#_Toc76387101)

[9.1 Agile 10](#_Toc76387102)

# Problem Statement

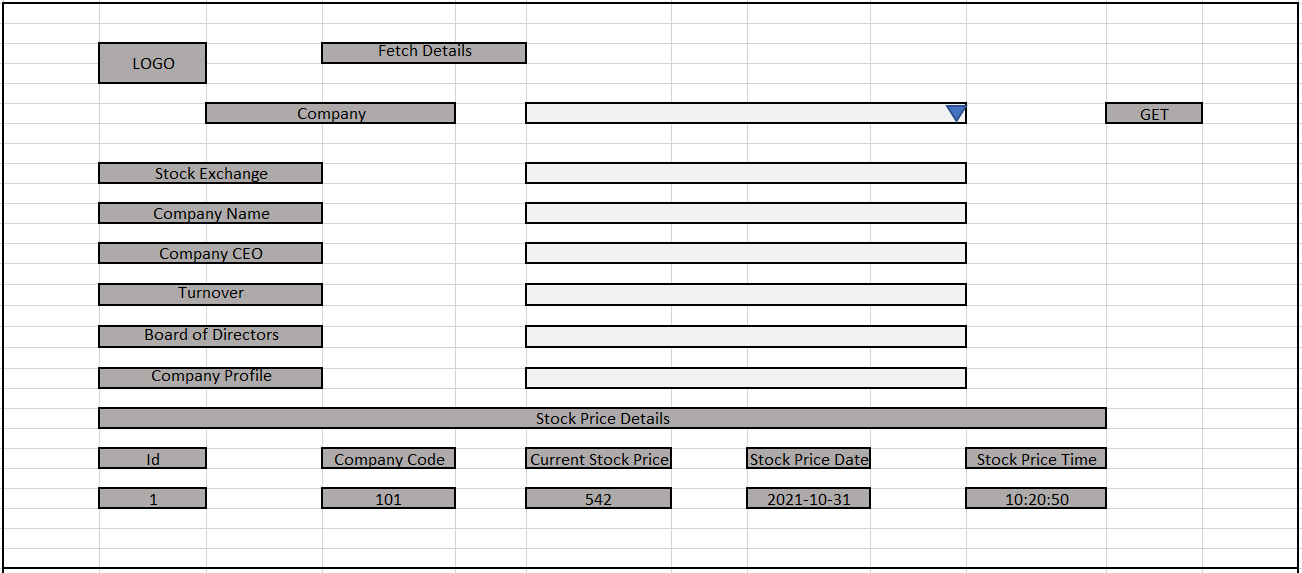
**E-Stock Application** is simple Spring boot application with MySQL, where it allows any unregistered users to manage the stocks at any stock exchanges like create, view, modify and delete stock price details and company details. The core modules of E-Stock app are:

* Allows to add a new company detail and a new stock price detail
* Allows to delete an existing company or existing stock
* Allows to display all company information or all stock detail
* Allows to search the company or stock on the basis of company code

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

# Proposed E-Stock Application Wireframe

Below is the wireframe for reference.



# Tool Chain

|  |  |  |
| --- | --- | --- |
| Competency | Skill | Skill Detail |
| Engineering Mindset | Code Quality |  |
|  |  |  |
| Programming Languages | Application Language | Java |
| Products & Frameworks | Presentation | Angular/React |
|  |  | Javascript/Typescript |
|  |  | Bootstrap |
|  | Compute & Integration | Spring Boot |
|  |  |  |
|  | Governance & Tooling | JUnit |
|  |  | Mockito |
|  |  | Jasmine |
|  |  | JaCoCo |
|  |  | SonarQube |

# Business Requirements:

Below are the user stories for the given problem statement

|  |  |  |  |
| --- | --- | --- | --- |
| **User**  **Story #** | **User Story Name** | **User Story** | **Development** |
| US\_01 | Company Functionalities | Add a new company detail for the Stock Exchange  *While adding a company, following information is required.*   * Stock Exchange * Company Name * Company CEO * Turnover * Board of Directors * Company Profile   Validations:   1. Stock Exchange is not null, min 5 and max 50 characters. 2. Company Name is not null, min 5 and max 50 characters. 3. Company CEO is not null, min 3 and max 50 characters. 4. Company Turnover is not null, precision 10 and scale 2 5. Company Board of Directors is not null, min 5 and max 200 characters 6. Company profile is not null, min 5 and max 255 characters | Only API to be developed |
| US\_02 | Company Functionalities | Delete a Company with given company code  Constraints:   1. If company is tried to be deleted and company does not exist, it must throw a custom exception. | Only API to be developed |
| US\_03 | Stock Price Functionalities | Add Stock price details for a Company  *While adding Stock Price Details, following information is required.*   * Company Code * Current Stock Price * Stock Price Date * Stock Price Time   Precondition:   1. Company Code must be of an existing Company available for Stock details (Can get from Database)   Constraints:   1. Stock Price date cannot be past date if it is past date, it must throw a custom exception.   Validations:   1. Stock Exchange is not null, min 3 and max 100 characters. 2. Current Stock Price is not null, precision 10 and scale 2. 3. Stock Price Date is not null and never exceed current date 4. Stock Price time is not null and never exceed current time | Only API to be developed |
| US\_04 | Stock functionality | Fetches the Stock with the given companyCode  Considerations:   1. While fetching the above details the API must return the Company details (Stock Exchange, Company Name, Company CEO, Turnover, Board of Directors, Company Profile) along with Stock Price details. | API and Frontend to be developed |
| US\_05 | Stock functionality | Display the max, min and average stock prices between the stipulated time period  Constraints:   1. If the company code does not exist, it must throw a custom exception | Only API to be developed |

# Proposed Rest Endpoints to be exposed

## Rest APIs:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **URL Exposed** | **Purpose** | | /e-stock/api/v1/company/add-company | Adds a new company | | /e-stock/api/v1/ company/deleteCompany/{companyCode} | Delete Company with given Company Code | | /e-stock/api/v1/stock/add-stock | Add a new Stock Price Details | |
| |  |  | | --- | --- | | /e-stock/api/v1/stock/getStockByCompanyCode/{companyCode} | Fetches the Stock with the given Company Code | | / e-stock/api/v1/ stock/getStockPriceIndex/{companyCode}/{startDate}/{endDate} | Fetches Stock Price Index with Company Code and duration | |

# Implementation/Functional Requirements

## Product and Frameworks:

1. **Presentation**

Develop the frontend for User Story 4. (Implementation as follows)

1. Implement using either Angular or React.
2. Implement any one of the Gang of four Patterns to compose data using typescript before presenting the same on UI.
3. Implement at least one approach for UI performance consideration.
4. Identify and Implement client-side Optimization Techniques for Bootstrap.
5. **Compute and Integration**

Develop the backend application as a RESTful Application. (Implementation as follows)

* + 1. Use any one of the Creational Design patterns for composing the model object to be sent back as response on following endpoint:

/ e-stock/api/v1/company/getCompanyInfoById/{companyCode}

(Fetches details of a company based on the company code)

* + 1. Optimize you REST endpoints to allow filtering, sorting, and pagination.

## Governance and Tooling:

* + - 1. Follow the practise of Creating Testable Component
      2. Configure your frontend application to implement Unit Testing using Jasmin & Karma.
      3. Configure your backend application to implement Unit Testing using Junit and Mockito.
      4. Test suites must contain exception situation testing.
      5. Generate the Code Coverage report of the same.
      6. Generate the Code Quality Report of the backend application.

## Code Quality/Optimizations

1. Associates should have written clean code that is readable.
2. Associates need to follow SOLID programming principles.
3. Generate the code quality report of the same.