System Requirements Specification Index

For

E-Stock Market Application

Version 4.0

IIHT Pvt. Ltd.
IIHT Ltd, No: 15, 2nd Floor, Sri Lakshmi Complex, Off MG Road, Near SBI LHO,
Bangalore, Karnataka – 560001, India
fullstack@iiht.com

E-STOCK MARKET SYSTEM

System Requirements Specification

1.Business-Requirement:

1.1 Problem Statement:

Stock Market Application is a simple .Net Core 3.1 RESTful Web API application with MS Sql server, where it allows any unregistered users to manage the stocks at any stock exchanges like create, view and delete stock price details and company details.

Every Company Details should have the properties like company code, company name, company CEO, stock exchange, turnover, board of directors and company profile.

Every Stock Price Details should have the properties like company code, current stock price, stock price date, and stock price time.

The relationship between these two entities is that every company would have multiple stock prices and can generate max, min and average stock prices between the stipulated time period.

1.2 Following is the requirement specification:

	E-Stock Market Application
USERS	
1	Company Info
2	Stock Price
Company Info Module Functionalities	
	1.Create a Company Info
	2.Can delete a Company Info
	4.Get Company Info by company code
	5.Fetch all Companies
Stock Price Module Functionalities	

	1.Add a new Stock Price Details
	2.Delete Stock with given company code
	3.Fetches the Stock with the given company code
	4.Fetches all the Stock Price Details
	5.Fetches Stock Price Index with companyCode and duration

2. Assumptions, Dependencies, Risks / Constraints

2.1 Company Info Constraints

- While adding a company details, if company code is already existing, it should throw a custom exception
- While deleting the company, ensure that company code already exists, if not, the operation should throw a custom exception

2.2 Stock Price Constraints

- The Company Details and Stock Price Details are connected through a field company code – applying integrity constraint

2.3 Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the database operations must be implemented on entity object only
- Do not change, add, remove any existing methods in service layer
- In Repository interfaces, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data wrapped in ResponseEntity

- **All business logic CRUD operations under repository class and write your business logic validation in Services class and related validation use proper user defined exceptions mentioned in above document.**
- **Controller must validate before processing any logic on the database.**

2.4 Visitors can perform the follow actions

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

- Allows to display max, min and average stock prices between the stipulated time period.

2.5 ToolChain

- .NET Core 5.0, RESTful Web Services, MS SQL Server.

3. Business Validations

3.1 Company Info Class Entities

- Company Code (long) must be not null and unique
- Company Name (string) is not null, min 3 and max 100 characters.
- Company CEO (string) is not null, min 5 and max 100 characters.
- Company Turnover (double) is not null, precision 10 and scale 2.
- Company Board of Directors(string) is not null, min 5 and max 200 characters.
- Company profile (string) is not null, min 5 and max 255 characters.
- Stock Exchange (string)is not null, min 3 and max 100 characters.
- IsDeleted (bool).

3.2 Stock Price Class Entities.

- Stock Price Id (long) is not null, key attribute
- Current Stock Price is not null, precision 10 and scale 2.
- Stock Price Date is not null and never exceed current date
- Stock Price time is not null and never exceed current time
- IsDeleted (bool).

4. Considerations

- For Role of application users 2 possible values must be used: -
    1. CompanyInfo
    2. StockPrice

5. Rest Endpoints

Rest End-points to be exposed in the controller along with method details for the same to be created

5.1 CompanyInfoController

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>/company/addCompany<table><tr><td>Http Method</td><td>Post</td></tr><tr><td>Parameter 1</td><td>CompanyInfo companyInfo</td></tr><tr><td>Return</td><td>HttpResponse status code</td></tr></table></td><td>Add a new Company Details</td></tr>
<tr><td>/company/deleteCompany/{companyCode}<table><tr><td>Http Method</td><td>DELETE</td></tr><tr><td>Parameter 1</td><td>companyCode</td></tr><tr><td>Return</td><td>HttpResponse status code</td></tr></table></td><td>Delete Company with given Company Code.</td></tr>
<tr><td>/company/getCompanyInfoById/{companyCode}<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>companyCode</td></tr><tr><td>Return</td><td><CompanyInfo></td></tr></table></td><td>Fetches the Company Details with the given Company Code</td></tr>
<tr><td>/company/getAllCompanies<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>-</td></tr><tr><td>Return</td><td><<IEnumerable<CompanyInfo>>></td></tr></table></td><td>Fetches all the Company Details</td></tr>
</table>

5.2 StockPriceController

<table>
<tr><th>URL Exposed</th><th>Purpose</th></tr>
<tr><td>/stock/addStock<table><tr><td>Http Method</td><td>POST</td></tr><tr><td>Parameter 1</td><td>StockPrice stockprice</td></tr><tr><td>Return</td><td>HttpResponse status code</td></tr></table></td><td>Add a new Stock Price Details</td></tr>
</table>

<table>
<tr><td></td><td></td></tr>
<tr><td>/stock/deleteStock/{companyCode}<table><tr><td>Http Method</td><td>DELETE</td></tr><tr><td>Parameter 1</td><td>companyCode</td></tr><tr><td>Return</td><td>HttpResponse status code</td></tr></table></td><td>Delete Stock with given companyCode</td></tr>
<tr><td>/stock/getStockByCompanyCode/{companyCode}<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>companyCode</td></tr><tr><td>Return</td><td><StockPrice></td></tr></table></td><td>Fetches the Stock with the given companyCode</td></tr>
<tr><td>/stock/getAllStock<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>-</td></tr><tr><td>Return</td><td><IEnumerable<StockPrice>></td></tr></table></td><td>Fetches all the Stock Price Details</td></tr>
<tr><td>/stock/getStockPriceIndex/{companyCode}/{startDate}/{endDate}<table><tr><td>Http Method</td><td>GET</td></tr><tr><td>Parameter 1</td><td>companyCode</td></tr><tr><td>Parameter 2</td><td>startDate</td></tr><tr><td>Parameter 3</td><td>endDate</td></tr><tr><td>Return</td><td><IEnumerable<StockPrice>></td></tr></table></td><td>Fetches Stock Price Index with companyCode and duration</td></tr>
</table>

6. Template Code Structure

6.1 Package: E-StockMarket

Resources

Names	Resource	Remarks	Status
Package Structure			
controller	CompanyInfo Controller StockPrice Controller	Controller class to expose all rest-endpoints for auction related activities.	Partially implemented
Startup.cs	Startup CS file	Contain all Services settings and SQL server Configuration.	Already Implemented

Properties	launchSettings.json file	All URL Setting for API	Already Implemented
	appsettings.json	Contain connection string for database	Already Implemented

6.2 Package: E-StockMarket.BusinessLayer

Resources

Names	Resource	Remarks	Status
Package Structure			
Interface	ICompanyInfoServices interface IStockPriceServices interface	Inside all these interface files contains all business validation logic functions.	Already Implemented
Service	CompanyInfo Services CS file StockPrice Services CS file	Using this all class we are calling the Repository method and use it in the program and on the controller.	Partially Implemented
Repository	ICompanyInfoRepository CompanyInfo Repository IStockPriceRepository StockPriceRepository CS file and interface.	All these interfaces and class files contain all CRUD operation code for the database. Need to provide implementation for service related functionalities	Partially Implemented
ViewModels	RegisterCompanyInfoViewModel, RegisterStockPriceViewModel,	Contain all view Domain entities for show and bind data. All the business validations must be implemented.	Partially Implemented

6.3 Package: E-StockMarket.DataLayer

Resources

Names	Resource	Remarks	Status
Package Structure			
DataLayer	StockMarketDbContext cs file	All database Connection and collection setting class	Already Implemented

6.4 Package: E-StockMarket.Entities

Resources

Names	Resource	Remarks	Status
Package Structure			
Entities	CompanyInfo, StockPrice CS file	All Entities/Domain attribute are used for pass the data in controller. Annotate this class with proper annotation to declare it as an entity class with **Id** as primary key. Generate the **Id** using the **IDENTITY** strategy	Partially Implemented

6.5 Package: E-StockMarket.Tests

Resources

The E-StockMarket.Tests project contains all test case classes and functions for code evaluation. Don't edit or change anything inside this project.

7. Execution Steps to Follow

1. All actions like build, compile, running application, running test cases will be through Command Terminal.

2. To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) Terminal → New Terminal.

3. On command prompt, cd into your project folder (cd <Your-Project-folder>).

4. To connect SQL server from terminal:
   (E-StockMarket/sqlcmd -S localhost -U sa -P pass@word1)
   - To create database from terminal -
     ```
     1> Create Database StockMarket_Db
     2> Go
     ```

5. Steps to Apply Migration(Code first approach):
   - Press Ctrl+C to get back to command prompt
   - Run following command to apply migration-
     (E-StockMarket/dotnet-ef database update)

6. To check whether migrations are applied from terminal:
   (E-StockMarket/sqlcmd -S localhost -U sa -P pass@word1)

   ```
   1> Use StockMarket_Db
   2> Go
   1> Select * From __EFMigrationsHistory
   2> Go
   ```

7. To build your project use command:
   (E-StockMarket/dotnet build)

8. To launch your application, Run the following command to run the application:
   (E-StockMarket/dotnet run)

9. This editor Auto Saves the code.

10. To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.

11. **To test web-based applications on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**

    Note: The application will not run in the local browser

12. **To run the test cases in CMD, Run the following command to test the application:**
    (E-StockMarket/dotnet test --logger "console;verbosity=detailed")
    (You can run this command multiple times to identify the test case status,and refactor code to make maximum test cases passed before final submission)

13. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B - command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**

14. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**

15. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**
