|  |  |
| --- | --- |
|  | **Cognizant Academy**  **Manufacturing RFQ Management**  **FSE – Business Aligned Project**  **Case Study Specification**  **Version 1.0** |
| |  |  |  |  | | --- | --- | --- | --- | |  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** | | **Name** | Srilakshmi Jayaraman |  |  | | **Role** | Solution Designer |  |  | | **Signature** |  |  |  | | **Date** |  |  |  | |
|  |

Table of Contents

[1.0 Introduction 3](#_Toc46000628)

[1.1 Purpose of this document 3](#_Toc46000629)

[1.2 Project Overview 3](#_Toc46000630)

[1.3 Scope 3](#_Toc46000631)

[1.4 Hardware and Software Requirement 3](#_Toc46000632)

[2.0 Reference learning and requirement 3](#_Toc46000633)

[3.0 Menu item listing and adding to cart 4](#_Toc46000634)

[3.1 MenuItemListing WebApi 4](#_Toc46000635)

[3.2 OrderItem WebApi 5](#_Toc46000636)

[4.0 Cloud deployment and consumption in Console application 5](#_Toc46000637)

[5.0 Change Log 6](#_Toc46000638)

# Important Instructions

1. Associate must adhere to the Design Considerations specific to each Technolgy Track
2. Associate must not submit project with compile-time or build-time errors
3. Being a Full-Stack Developer Project, you must focus on ALL layers of the application development
4. Unit Testing is Mandatory, and we expect a code coverage of 100%. Use Mocking Frameworks wherever applicable.
5. All the Microservices, Client Application, DB Scripts, have to be packaged together in a single ZIP file. Associate must submit the solution file in ZIP format only
6. If backend has to be set up manually, appropriate DB scripts have to be provided along with the solution ZIP file
7. A READ ME has to be provided with steps to execute the submitted solution, the Launch URLs of the Microservices in cloud must be specified.

(Importantly, the READ ME should contain the steps to execute DB scripts, the LAUNCH URL of the application)

1. Follow coding best practices while implementing the solution. Use appropriate design patterns wherever applicable
2. You are supposed to use an In-memory database or sessions as specified, for the Microservices that will be deployed in cloud. No Physical database is suggested.

# Introduction

## Purpose of this document

The purpose of the software requirement document is to systematically capture requirements for the project and the system “Manufacturing RFQ Management” that has to be developed. Both functional and non-functional requirements are captured in this document. It also serves as the input for the project scoping.

The scope of this document is limited to addressing the requirements from a user, quality, and non-functional perspective.

High Level Design considerations are also specificed wherever applicable, however the detailed design considerations have to be strictly adhered to during implementation.

## Project Overview

A leading Manufacturing Industry wants to develop Middleware Services for handling “Request for Quote” Processing in their organization.

The Maufacturing Industry has a small plant that assembles various types of agriculture machineries. In order to do this, the Industry wants to buy several parts from its suppliers that addresses the machine design, material choices and price expectations. (Assume that the Industry has only 1 plant).

To identify a supplier who can support in providing high quality product with a reasonable price at the point of need, the Industry floats “Request for Quote” (RFQ) to all its suppliers.

Detailed RFQs allow suppliers to prepare accurate quotes that minimize the potential for cost overruns, delivery delays and at the same time meets the product requirements.

The Middleware services of the Manufacturing Firm will have to collate the needs of products/services from its manufacturing Plant. The inputs needed for preparing a comprehensive RFQ must be provided by the services.

The services must also identify the potential suppliers who can provide the required products.

There will also be an Admin Portal, called as “Fabricate” that has to be developed part of this scope that consumes these Microservices and helps the Admin Staffs with needed data for RFQ preparation.

## Scope

Below are the modules that needs to be developed part of the Project:

|  |  |  |
| --- | --- | --- |
| **Req. No.** | **Req. Name** | **Req. Description** |
| REQ\_01 | Supplier Module | Supplier Module is a Middleware Microservice that performs following operations:   * Get Potential Suppliers with Feedback Rating, for a given Part * Add Supplier Details * Update Feedback Rating of a Supplier |
| REQ\_02 | Plant Module | Plant Module is a Middleware Microservice that performs the following operations:   * View Reorder Point of all parts of the Plant, as of date * View Stock On Hand of a Part * Update Min/Max quantities for Calculating Re-order Points |
| REQ\_03 | RFQ Module | Request For Quote (RFQ) Module is a Middleware Microservice that performs the following operations:   * Get RFQ of a Part * Get Potential Suppliers of a RFQ |
| REQ\_04 | Admin Portal - Fabricate | An Web Portal that allows a member to Login and allows to do following operations:   * Login * View Stock Details like Stock on Hand, Part Requirements, Min/Max quantities / Re-order Points * Update Min/Max Quantities of a Part and refresh Re-order Points * View / Update Supplier Details * Provide Feedback Rating of Supplier * Get RFQ Details * Get Potential Suppliers of RFQ |

Note: The project phase is for 2 weeks. The first week is to be developed on local machine and the second week deals with Cloud deployment.

The requirement details given below states in-memory database usage. **The first phase of the development which is done in the first week, SHOULD use the Database for related activities and NOT the in-memory database.**

The second phase of the development which is done in the second week, can use the in-memory database as mentioned in the requirement, with appropriate code modifications.

## Hardware and Software Requirement

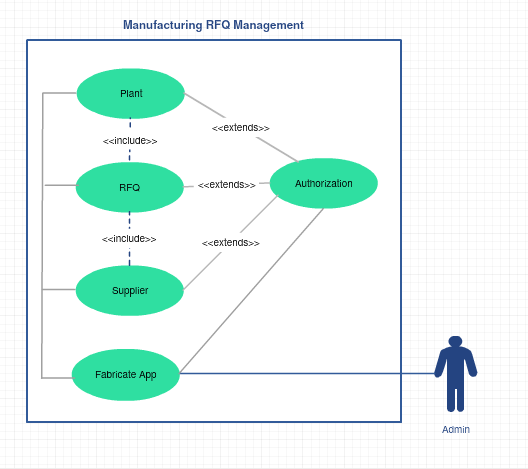
1. Hardware Requirement:
   1. Developer Desktop PC with 8GB RAM
2. Software Requirement (Java)
   1. Spring Tool Suite (STS) Or any Latest Eclipse
      1. Have PMD Plugin, EclEmma Code Coverage Plugin and AWS Code Commit Enabled
      2. Configure Maven in Eclipse
   2. Maven
   3. Docker (Optional)
   4. Postman Client in Chrome
   5. AWS Cloud Account
3. Software Requirement (Dotnet)
   1. Visual studio 2017 enterprise edition
   2. SQL Server 2014
   3. Postman Client in Chrome
   4. Azure cloud access

## System Architecture Diagram



# Functional Requirements and High Level Design

## Use Case Diagram



## Individual Components of the System

### **Supplier Microservice**

|  |  |
| --- | --- |
| Manufacturing RFQ Management | Supplier Microservice |
| **Functional Requirements**  Can assume that Fabricate App and RFQ Module are the clients to this Microservice. An authorized admin staff view the view the potential suppliers of a part, can add/update the supplier details, feedback rating  Post Authorization, the service will expose for the following functionalities:   * View the potential suppliers of the given Part * Add Supplier Details * Edit Supplier Details * Update Feedback Rating | |
| Entities   1. Supplier   <Details of Supplier, location, feedback>   1. Supplier\_Part   <Details of Parts that can be supplied, in what quantitiy, in what time period etc.>  **REST End Points**  Supplier Microservice   * + GET: /getSupplierOfPart (Input: Part\_ID | Output: Supplier Details like ID, Name, Location, Possible Supply Quantity, Supply Time around etc.)   + POST: /addSupplier (Input: Part Details and Supplier Details | Output: Status, Status Description)   + POST: /editSupplier (Input: Part Details and Supplier Details | Output: Status, Status Description)   + POST: /updateFeedback (Input: Feedback Details of Supplier | Output: Status, Status Description) | |
| **Trigger** – Can be invoked from Fabricate Application and also from RFQ Microservice | |
| **Steps and Actions**   1. Authorized Admin Staff logs in to the Fabricate. Based on the authentication and authorization at Supplier service, the above mentioned operations are exposed as REST end points 2. Assume that the Supplier details will be stored in database through a separate process, which is out of system boundary. 3. While Adding/Editing Supplier details, perform the basic data checks like valid Email/Valid Phone/Valid Location etc., before addition. | |
| **Non-Functional Requirement:**   * Only Authorized Member can access these REST End Points * If Add/Edit Supplier or Provide Feedback end points are accessed concurrently, then the data consistency has to be maintained. | |

### **Plant Microservice**

|  |  |
| --- | --- |
| Manfacturing RFQ Management | Plant Microservice |
| **Functional Requirements**  In supply chain, the stock replenishment is an operation that consists in ordering more stocks in order to fulfill the customer demand. Replenishment is typically triggered when the stock level hits the reorder point a setting from the system.  In the Min-Max inventory model, the Min value represents the reorder point, while the Max represents the targeted quantity. Reorder quantity is Max minus Min (i.e. the difference between Min and Max).  Plant Microservice takes care of the stock replenishment of the Plant, will be invoked from Fabricate application and also from RFQ Microservice.  Post authentication and authorization, the service performs the following operations:   * Update the Min Quantity / Max Quantity of the Part based on assembly demands of the Plant   + Min Quantity – if a stock reduces to this number, a reorder is essential   + Max Quantity – The target numbers needed to meet the demand * View Reorder Point of any Part as of current date   + The list of Parts that needs tobe re-ordered as of date * View Stock in Hand of a given part as of current date   + To get the actual stock in hand of a Part, as of date | |
| **Entities**   1. **Part**   <Part Details, ID, Description, Part Specification, Stock In Hand etc.>   1. **Demand**   <Demand ID, Demand Raied date etc.>   1. **Reorder\_Rules**   <Min, Max Quantiity of the Product, Re-order Frequency etc.>   1. **Plant\_Reorder\_Details**   <Part Details, Re-order Request Status, Date etc.>  **REST End Points**  **Plant Microservice**   * + GET: /viewPartsReOrder (Input: None | Output: Part Details)   + GET: /viewStockInHand (Input: partId | Output: Stock Details)   + POST: /updateMinMaxQuantities (Input: partId, minQuantity, maxQuantity) | Output: Status, Status Description) | |
| **Trigger** – Can be invoked from RFQ Microservice and Fabricate App | |
| **Steps and Actions**   1. If /viewStockInHand is invoked, the stock has to be returned for the given part ID. 2. If /viewPartsReorder end point is invoked by it has to calculate the re-order count based on demand number and the Min Quantity number. If the stock of the part reaches less than the min quantity number, immediately it can be considered for re-order. 3. If /updateMinMaxQuantities end point is invoked then the min and max counts will be updated. Certain business rules for this is given below:    * Min must be 30% - 50% of the Max    * Max must not exceed 20% of the demand | |
| **Non-Functional Requirement:**   * Viewing a Stock is considered as a frequent operation hence a different REST endpoint is needed. * Re-order calculation must not happen sequentially part by part, instead try doing in parallel. | |

### **RFQ Microservice**

|  |  |
| --- | --- |
| Manufacturing RFQ Management | RFQ Microservice |
| **Functional Requirements**  RFQ Microservice interacts with Supplier and Plant Microservices. Post authorization of request, RFQ Microservice allows the following operations:   * Process the details required for RFQ and expose it * Provide the potential vendors for each Part, to circulate the RFQ | |
| **Entities**   1. **RFQ**   <RFQ Details like RFQ ID, Demand ID, Part Details, Quantity, Expected Supply Date, Specificiation etc.>  **REST End Points**  **RFQ Microservice**   * + GET: /getRFQOfPlant (Input: Plant\_ID | Output (RFQ Details)   + GET: /getPotentialVendorsOfRFQ (Input: RFQ ID | Output: Based on the parts in RFQ, identify the vendors who has feedback rating > 7 out of 10) | |
| **Trigger** – Can be invoked from Fabricate App | |
| **Steps and Actions**   * + RFQ of Plant must be calculated based on response from Plant Microservice, on the parts that needs reorder   + Potential Vendors have to be returned by reaching the Supplier Service who has good feedback to supply the given part based on the feedback rating | |

### **Authorization Microservice**

|  |  |
| --- | --- |
| Manufacturing RFQ Management | Authorization Microservice |
| **Security Requirements**   * Service to Service communication has to happen using JWT * Pass End User Context across Microservices * Have the token expired after specific amount of time say 15 minutes. * Have this service configured in the cloud along with other services | |

### **Swagger**

|  |  |
| --- | --- |
| Manufacturing RFQ Management | Swagger |
| **Documentation Requirements**   * All the Microservices must be configured with Swagger for documentation   **For Java based Implementation:**   * Register the swagger resources in the Swagger Microservice and enable them as REST end points * Configure this service along with other services in the cloud | |

### **Fabricate Portal (MVC)**

|  |  |
| --- | --- |
| Manuracturing RFQ Management | Fabricate Portal |
| **Fabricate Portal Requirements**   * Fabricate Portal must allow a member to Login. Once successfully logged in, the member do the following operations:   + View Stock Details like Stock on Hand, Part Requirements, Min/Max quantities / Re-order Points   + Update Min/Max Quantities of a Part and refresh Re-order Points   + View / Update Supplier Details   + Provide Feedback Rating of Supplier   + Get RFQ Details   + Get Potential Suppliers of RFQ * Each of the above operations will reach out to the middleware Microservices that are hosted in cloud. | |

# Cloud Deployment requirements

* All the Microservices must be deployed in Cloud
* All the Microservices must be independently deployable. They have to use In-memory database or user sessions wherever applicable
* The Microservices has to be dockerized and these containers must be hosted in Cloud using CI/CD pipelines
* The containers have to be orchestrated using Azure or AWS Kubernetes Services.
* These services must be consumed from an MVC app running in a local environment.

# Design Considerations

Java and Dotnet specific design considerations are attached here. These design specifications, technology features have to be strictly adhered to.



# Reference learning

Please go through all of these k-point videos for Microservices deployment into AWS.

|  |
| --- |
| <https://cognizant.kpoint.com/app/video/gcc-6e36500f-c1af-42c1-a6c7-ed8aac53ab22> |
| [https://cognizant.kpoint.com/app/video/gcc-92f246c9-024a-40b7-8bfc-96b3ce7c1a39](https://apc01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fcognizant.kpoint.com%2Fapp%2Fvideo%2Fgcc-92f246c9-024a-40b7-8bfc-96b3ce7c1a39&data=02%7C01%7Ckumar.mahadevan%40cognizant.com%7C1278a7e184c6454d69c108d7fbe06c69%7Cde08c40719b9427d9fe8edf254300ca7%7C0%7C0%7C637254813626816518&sdata=9A4V%2F2ippq99uff4iyxYxHAr1qyLptaQgjcAJjvw5Kw%3D&reserved=0) |
| <https://cognizant.kpoint.com/app/video/gcc-cfedd9c1-e29e-4e3e-b3e2-1960277f72a3> |
| <https://cognizant.kpoint.com/app/video/gcc-900a7172-43b7-42f3-a6cc-e301bd9cc9b3> |

Microservices deployment into Azure Kubernetes Service.

|  |
| --- |
| [AzureWithCICD-1](https://cognizant.kpoint.com/app/video/gcc-19532393-d4e0-4fd9-8a0c-80ecbdb349d3) |
| [AzureWithCICD-2](https://cognizant.kpoint.com/app/video/gcc-6633a958-ab72-4c69-b926-fe832e4b56a1) |
| [AzureWithCICD-3](https://cognizant.kpoint.com/app/video/gcc-553eb186-c1cf-448e-96fc-a96fe37b2e6a) |
| [AzureWithCICD-4](https://cognizant.kpoint.com/app/video/gcc-fad7d4af-d651-4501-99c6-2785190670c2) |

**Other References:**

|  |  |
| --- | --- |
| Java 8 Parallel Programming | <https://dzone.com/articles/parallel-and-asynchronous-programming-in-java-8> |
| Feign client | [https://dzone.com/articles/Microservices-communication-feign-as-rest-client](https://dzone.com/articles/microservices-communication-feign-as-rest-client) |
| Swagger (Optional) | [https://dzone.com/articles/centralized-documentation-in-Microservice-spring-b](https://dzone.com/articles/centralized-documentation-in-microservice-spring-b) |
| ECL Emma Code Coverage | <https://www.eclipse.org/community/eclipse_newsletter/2015/august/article1.php> |
| Lombok Logging | <https://javabydeveloper.com/lombok-slf4j-examples/> |
| Spring Security | <https://dzone.com/articles/spring-boot-security-json-web-tokenjwt-hello-world> |
| H2 In-memory Database | <https://dzone.com/articles/spring-data-jpa-with-an-embedded-database-and-spring-boot>  <https://www.baeldung.com/spring-boot-h2-database> |
| AppInsights logging | <https://www.codeproject.com/Tips/1044948/Logging-with-ApplicationInsights> |
| Error response in WebApi | <https://stackoverflow.com/questions/10732644/best-practice-to-return-errors-in-asp-net-web-api> |
| Read content from CSV | <https://stackoverflow.com/questions/26790477/read-csv-to-list-of-objects> |
| Access app settings key from appSettings.json in .Netcore application | <https://www.c-sharpcorner.com/article/reading-values-from-appsettings-json-in-asp-net-core/>  <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration/?view=aspnetcore-3.1> |

# Change Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Changes Made | | | |
| V1.0.0 | Initial baseline created on <21-Jul-2020> by <Srilakshmi Jayaraman> | | | |
|  |  | | | |
| **Section No.** | **Changed By** | **Effective Date** | **Changes Effected** |
|  |  |  |  |