Cognizant Academy

Return Order Management System

Digital Honors Project
Case Study Specification

Version 1.0

	Prepared By / Last Updated By	Reviewed By	Approved By
Name	Seshadri M R		
Role	Solution Designer		
Signature			
Date			

Table of Contents

1.0 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 Unit testing and Mocking Frameworks wherever applicable.
5. If backend has to be set up manually, appropriate DB scripts have to be provided along with the solution ZIP file.
6. Follow coding best practices while implementing the solution. Use appropriate design patterns wherever applicable.
7. You are supposed to use an In-memory database or code level + Cloud data as specified, for the Microservices that should be deployed in cloud.

2.0 Introduction

2.1 Purpose of this document

The purpose of the software requirement document is to systematically capture requirements for the project and the system “Return Order Management System” 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.

2.2 Project Overview

A leading Supply chain Management Organization wants to automate the return orders, by classifying them to repair or replacement. Repair is for all main or integral part of their product. Replacement is for accessories.

2.3 Scope

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

Req. No.	Req. Name	Req. Description
REQ_01	Component processing	Component processing Module is a Middleware Microservice that performs following operations: • Determine if the request is for Repair or Replacement • Determine the repair or replacement cost along with the consideration if it's a priority request or not. Determine the date of process completion • Invoke Packaging and Delivery service to determine the cost and date of delivery • Return the Processing response detail object
REQ_02	Packaging and Delivery module	Packaging and Delivery Module is a Middleware Microservice that performs the following operations: • Determine the packaging and delivery charge for the item based on a pre-defined logic • Provide the expected date of delivery
REQ_03	Authorization service	This microservice is used with anonymous access to Generate JWT
REQ_04	Return order portal	A Web Portal that allows a member to Login and allows to do following operations: • Login • Provide detail for Return order

		• View the processing detail • Confirm processing

The requirement details given below states in-memory database or code level data usage. **<u>On Cloud deployment, towards the end of the Cloud access and before the evaluation, this could be modified to use Cloud database.</u>**

<u>The front-end application to be done on Angular</u>

2.4 Hardware and Software Requirement

1. Hardware Requirement:

    a. Developer Desktop PC with 8GB RAM

2. Software Requirement (Java)
    a. Spring Tool Suite (STS) Or any Latest Eclipse
        i. Have PMD Plugin, EclEmma Code Coverage Plugin and AWS Code Commit Enabled
        ii. Configure Maven in Eclipse
    b. Maven
    c. Docker (Optional)
    d. Postman Client in Chrome
    e. Visual Studio Code latest version
3. Software Requirement (Dotnet)

    a. Visual studio 2017 enterprise edition

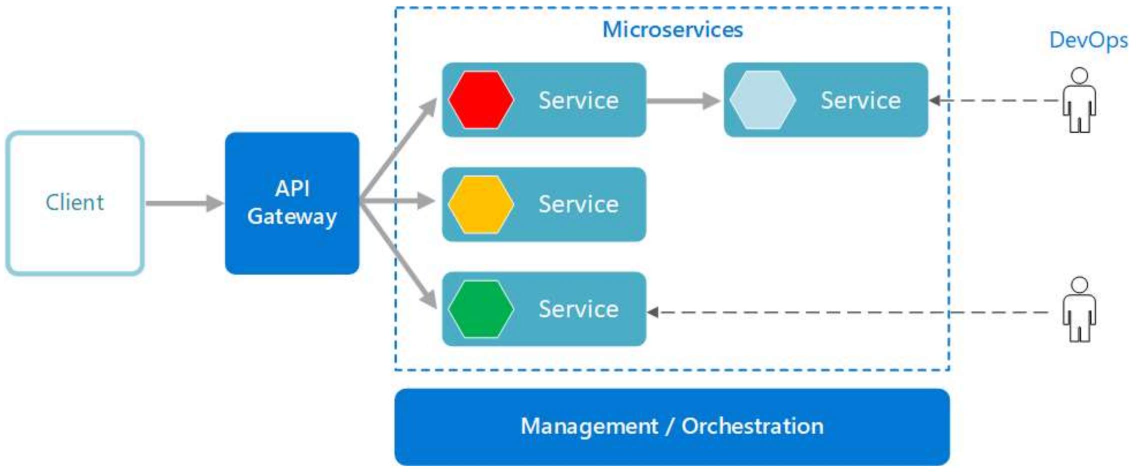
    b. SQL Server 2014

    c. Postman Client in Chrome

    d. Azure cloud access

    e. Visual Studio Code latest version

2.5 System Architecture Diagram



3.0 System Requirements

3.1.1 Functional Requirements – Component processing Microservice

Return Order Management System	**ComponentProcessing Microservice**
Functional Requirements The intent of this Microservice is to determine the Component processing detail. It interacts with packaging and delivery microservice to get the consolidated cost for the processing. Post Authorization using JWT, based upon the type of component – Integral or Accessory, repair or replacement workflow respectively, should determine the processing details.	
Entities **ProcessRequest** 1. **Name** <User name> 2. **ContactNumber** <User contact number> 3. **DefectiveComponentDetail** a. ComponentType – Integral / Accessory b. ComponentName c. Quantity <Details of defective component> **ProcessResponse**	

1. **RequestId**
   <A random number generated to identify the process detail>
2. **ProcessingCharge**
   <Total charge for the processing>
3. **PackagingAndDeliveryCharge**
   <Packaging and delivery charge>
4. **DateOfDelivery**
   <Date on which the product would be delivered to the user>

REST End Points

ComponentProcessing Microservice

- GET: /ProcessDetail (Input: ProcessRequest | Output: ProcessResponse)
- POST: /CompleteProcessing (Input: RequestId, CreditCardNumber, CreditLimit, ProcessingCharge) | Output: string response of the success of operation)

Trigger – Should be invoked from Return order Portal (local Angular app)

Steps and Actions

1. Return order Portal should be the front-end application where a user provides the detail of the defective component for servicing. An instance of the ProcessRequest object should be created to fill the request detail.
2. The portal should invoke the Authentication Microservice to get the JWT.
3. On receiving the token, the web portal should invoke the ComponentProcessing Microservice GET action method with the ProcessRequest object. JWT should be added to the request header for authorization.
4. The microservice design should have an Interface with a method declaration for Processing the service
   - There should be two Classes implementing the Interface. It should contain
     - Repair for Integral part workflow – This workflow should cater to the logic for Integral part servicing. The default processing duration should be 5 days. Default processing charge is INR 500
     - Replacement for Accessory part workflow – This workflow should cater to the logic for accessory part servicing. The default processing duration should be 2 days. Default processing charge is INR 300
5. The Process response should be displayed to the user.
6. Upon confirmation from the user to proceed, the CompleteProcessing action method should be invoked to complete the processing. Payment processing is out of scope of this requirement.
7. The complete processing detail(process request and response) should be saved in the database.

Non-Functional Requirement:

- Only Authorized requests can access these REST End Points

3.1.2 Functional Requirements – PackagingAndDelivery Microservice

Return Order Management System	PackagingAndDelivery Microservice

Functional Requirements

ComponentProcessing Microservice interacts with PackagingAndDelivery Microservice. PackagingAndDelivery Microservice allows the following operations:

The microservice should contain a list of packaging and delivery cost detail

Packaging items

- Integral item – INR 100
- Accessory – INR 50
- Protective sheath – INR 50

Delivery items

- Delivery charge for Integral item – INR 200
- Delivery charge for Accessory – INR 100

The microservice should get the component type and count to determine the packaging and delivery charge

REST End Points

Claims Microservice

- GET: /GetPackagingDeliveryCharge (Input: ComponentType, Count | Output: PacakagingAndDeliveryCharge)

Trigger – Can be invoked from ComponentProcessing Microservice

Steps and Actions

- This microservice should have only 1 REST endpoint to calculate the packaging and delivery charge
- The microservice has the detail on the charges as pre-defined values
- Based on the input, the packaging and delivery charge should be calculated and returned to the caller

Non-Functional Requirement:

3.1.3 Functional Requirements – Authorization Microservice

Return Order Management System	Authorization Microservice
Security Requirements ○ Create JWT ○ Have the token expired after specific amount of time say 30 minutes ○ Has anonymous access to get the token detail	

3.1.4 Functional Requirements – Return order portal

Return Order Management System	Return order Portal
Client Portal Requirements ○ Return Portal must allow a member to Login. Once successfully logged in, the member do the following operations: ○ Provide detail for Component processing ○ View processing detail and charge ○ Upon user confirmation, processing should complete displaying 'Processed successfully' message along with the processing detail. ○ The request detail should be saved in the database along with the complete process request and response detail ○ Each of the above operations should invoke the middleware Microservices that are hosted in cloud.	

4.0 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 data in the application 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 AWS/Azure Kubernetes Services.
- These services must be consumed from an Angular app running in a local environment.

5.0 Design Considerations

These design specifications, technology features have to be strictly adhered to.

DigitalHonors-Proje
ct-DesignConsiderat

6.0 Reference learning

Please go through all of these k-point videos for

Microservices deployment into Azure Kubernetes Service.

AzureWithCICD-1
AzureWithCICD-2
AzureWithCICD-3
AzureWithCICD-4

Microservice deployment to AWS

AWS Learning Reference 1
AWS Learning Reference 2
AWS Learning Reference 3

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
Swagger (Optional)	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	https://javabydeveloper.com/lombok-slf4j-examples/

Logging	
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

7.0 Change Log

	Changes Made			
V1.0.0	Initial baseline created on <8-Sep-2021> by <Seshadri M R>			
	Section No.	Changed By	Effective Date	Changes Effected