Cognizant Academy

Pension Management System

FSE – Business Aligned 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 Technology 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. 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)
8. Follow coding best practices while implementing the solution. Use appropriate design patterns wherever applicable.
9. You are supposed to use an In-memory database or code level data as specified, for the Microservices that should be deployed in cloud. No Physical database is suggested for Microservice.

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 “Pension 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

State government aims to automate a portion of the Pension detail provisioning. This project covers pensioner detail provision, calculate provision, initiate pension disbursement.

2.3 Scope

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

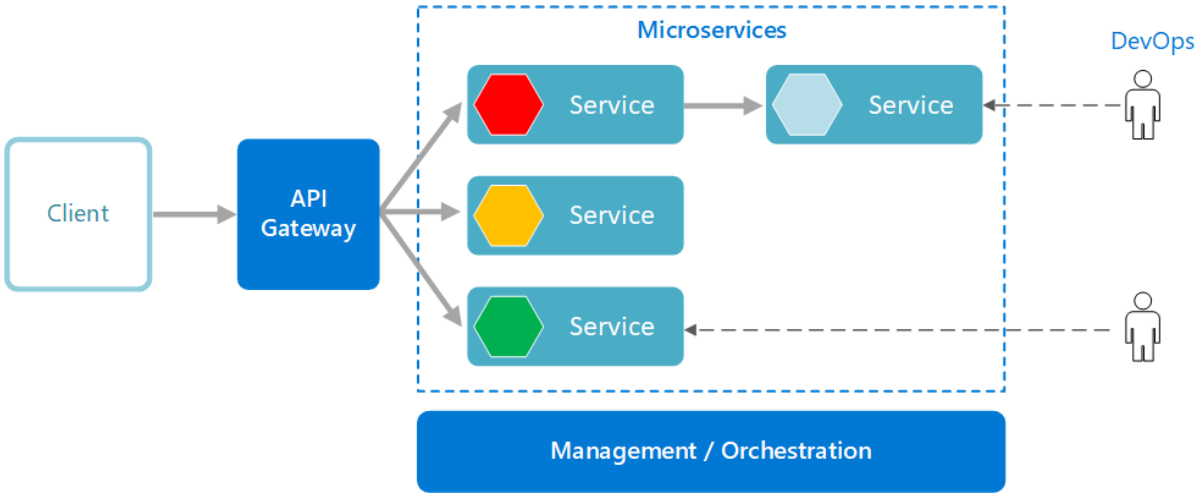
Req. No.	Req. Name	Req. Description
REQ_01	Process Pension module	This module is a Middleware Microservice that performs following operations: • Determines if it’s a self or family pension. Calculate the pension amount post data authentication, and display on the web application user interface • On confirmation from the admin to process the pension disbursement, Pension disbursement process should be initiated. List of error codes from Pension disbursement module should be maintained in this module. Based on the response appropriate message should be returned • This module should receive input from the web application
REQ_02	Pensioner detail module	This module is a Middleware Microservice that performs the following operations: • Provides information about the registered pensioner detail i.e., Pensioner name, PAN, bank name, bank account number, bank type – private or public

REQ_03	Pension disbursement module	This module is a Middleware Microservice that performs the following operations: • Gets the pension amount along with pensioner aadhaar details. This internally gets the bank detail from the Pensioner detail Microservice. Checks if the bank service charge is credited along with the pension amount or not. If the data is correct, then success code is returned, else appropriate error code is returned • This should be invoked from the ProcessPension microservice
REQ_04	Authorization service	This microservice is used with anonymous access to Generate JWT
REQ_05	Pension Management portal	A Web Portal that allows a member to Login and allows to do following operations: • Login • Load the pensioner detail • Invoke the Process pension module • Post validation of the pensioner detail, pension amount should be displayed on the UI. On approval, pension amount disbursal happens thru ProcessPension microservice • Store the pensioner detail, pension amount and the bank transaction detail in database

2.4 Hardware and Software Requirement

1. Hardware Requirement:
    a. Developer Desktop PC with 8GB RAM
2. Software Requirement (Dotnet)
    a. Visual studio 2017 enterprise edition
    b. SQL Server 2014
    c. Postman Client in Chrome
    d. Azure cloud access

2.5 System Architecture Diagram



3.0 System Requirements

3.1.1 Functional Requirements – Process Pension Microservice

Pension Management System	ProcessPension Microservice
Functional Requirements ProcessPension Microservice should be invoked from the web application. It allows the following operations: • It takes in the pensioner detail like the name, aadhaar number, pan detail, self or family or both type of pension • Verifies if the pensioner detail is accurate by getting the data from PensionerDetail Microservice or not. If not, validation message “Invalid pensioner detail provided, please provide valid detail.”. If valid, then pension calculation is done and the pension detail is returned to the Web application to be displayed on the UI	
Entity **PensionerInput** 1. **Name** <Pensioner name> 2. **Date of birth**	

<Pensioner date of birth>

3. **PAN**
   <Permanent account number>
4. **Aadhaar number**
5. **Self or Family pension**
   <Option to be chosen>

PensionDetail

1. **Name**
   <Pensioner name>
2. **Date of birth**
   <Pensioner date of birth>
3. **PAN**
   <Permanent account number>
4. **Self or Family pension**
   <Is the pension earned for self or family pension>
5. **Pension amount**
   <Total pension amount>

ProcessPensionInput

1. **Aadhaar number**
2. **Pension amount**
   <Total pension amount>

REST End Points

Claims Microservice

a. GET: /PensionDetail (Input: pensionerInput | Output: pensionDetail)

b. POST: /ProcessPension(Input: processPensionInput| Output: processingCode)

Trigger – Should be invoked from Pension management portal (local MVC app)

Steps and Actions

- This microservice should have 2 REST endpoints
- The GET endpoint should calculate the PensionInput, invoke the Pensioner detail microservice and get the salary detail. Pension amount calculation detail is as follows
  - **Self pension:** 80% of the last salary earned + allowances
  - **Family pension:** 50% of the last salary earned + allowances
- The POST endpoint should invoke the PensionerDetail Pension disbursement microservice Along with the bank service charge. The microservice can have pre-defined list of banks and service charge as follows
  - Public banks – INR 500
  - Private banks – INR 550
- The microservice should have pre-defined process codes and its message

- 10 – Pension disbursement Success
- 21 – Pension amount calculated is wrong, Please redo the calculation.

- On receiving the response from Pension disbursement microservice, if the process code is
  - 10 - the corresponding success message is returned to the web application.
  - 21 – try the service HIT for 3 times. If the same code is returned, return the message to the web application

Non-Functional Requirement:

- Only Authorized requests can access these REST End Points

3.1.2 Functional Requirements – Pensioner detail Microservice

Pension Management System	PensionerDetail Microservice

Functional Requirements

The intent of this Microservice is to provide the Pensioner detail based on Aadhaar number. Post Authorization using JWT, pensioner detail like the name, PAN detail, Bank name and bank account number

Entities

PensionerDetail

1. **Name**
   <Pensioner name>
2. **Date of birth**
   <Pensioner date of birth>
3. **PAN**
   <Permanent account number>
4. **SalaryEarned**
   <Last earned salary by the pensioner>
5. **Allowances**
   <Sum of all the allowances>
6. **Self or Family pension**
   <Is the pension classification self or family pension>
7. **Bank detail**
   a. **Bank name**
   b. **Account number**
   c. **Public or Private bank**

   <Bank detail>

REST End Points

PensionerDetail Microservice

- GET: /PensionerDetailByAadhaar (Input: aadhaarNumber | Output:

pensionerDetail)
Trigger – Should be invoked from ProcessPension microservice
Steps and Actions 1. This Microservice is to fetch the pensioner detail by the Aadhaar number. This should be consumed by Process pension microservice. 2. Flat file(CSV file with pre-defined data) should be created as part of the Microservice. This file has to contain data for 20 Pensioners. This has to be read and loaded into List for ALL the operations of the microservice.
Non-Functional Requirement: • Only Authorized requests can access these REST End Points

3.1.3 Functional Requirements – Pension disbursement Microservice

Pension Management System	PensionDisbursement Microservice
Functional Requirements PensionDisbursement Microservice should be invoked from the ProcessPension microservice. It allows the following operations: ○ Receives input for Pension disbursement details from the ProcessPension microservice. ○ This invokes the Pensioner detail Microservice to get the Bank details of the pensioner ○ Based on the bank type, checks if the bank service charge is sent through the corresponding field also check if the value of pension calculated is correct or not. This microservice should also the have pre-defined list of banks and service charge as follows, used for verification ▪ Public banks – INR 500 ▪ Private banks – INR 550 ○ Check if all the relevant data is there in the input. If so, Success code should be returned, else the appropriate error message should be returned. ▪ 10 - Success ▪ 21 – Pension amount calculated is wrong, Please redo the calculation.	
Entities **ProcessPensionInput** 1. **Aadhaar number** 2. **Pension amount**	

<Total pension amount>

3. **Bank service charge**

ProcessPensionResponse

1. **ProcessPensionStatusCode**

<Integer representing processing status>

REST End Points

PensionDisbursement Microservice

POST: /DisbursePension (Input: ProcessPensionInput | Output: ProcessPensionResponse)

Trigger: Should be invoked from ProcessPension microservice

Steps and Actions:

Non-Functional Requirement:

Only Authorized requests can access these REST End Points

3.1.4 Functional Requirements – Authorization Microservice

Pension 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	

3.1.5 Functional Requirements – Pension Management portal

Pension Management System	Pension Management Portal
Client Portal Requirements ○ Pension Management Portal must allow a member to Login. Once successfully logged in, the member do the following operations: ○ Provide the pensioner detail ○ Invoke the ProcessPension microservice to get the pension detail ○ UI should receive validation message if the pensioner detail provided as input has invalid data	

- Display the processed pension detail on the UI
- The pensioner and pension detail should be saved to the database
- Each of the above operations should reach out to 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 MVC app running in a local environment.

5.0 Design Considerations

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

CDE-Project-Design Considerations.pptx

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

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 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 .Net core 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
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

7.0 Change Log

<table>
<tr><th></th><th colspan="4">Changes Made</th></tr>
<tr><td>V1.0.0</td><td colspan="4">Initial baseline created on <25-Jul-2020> by <Seshadri M R></td></tr>
<tr><td rowspan="3"></td><td colspan="4"></td></tr>
<tr><td>Section No.</td><td>Changed By</td><td>Effective Date</td><td>Changes Effected</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>