SFJBS

Pension Management System

Case Study Specification

Version 1.0

	Prepared By / Last Updated By	Reviewed By	Approved By
Name			
Role			
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 BackEnd Developer Project, you must focus on ALL layers of the application development.
4. Unit Testing is Mandatory, and we expect a code coverage of 90+%. 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/Regular 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 “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 specified 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 and view for further processing.

2.3 Scope

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

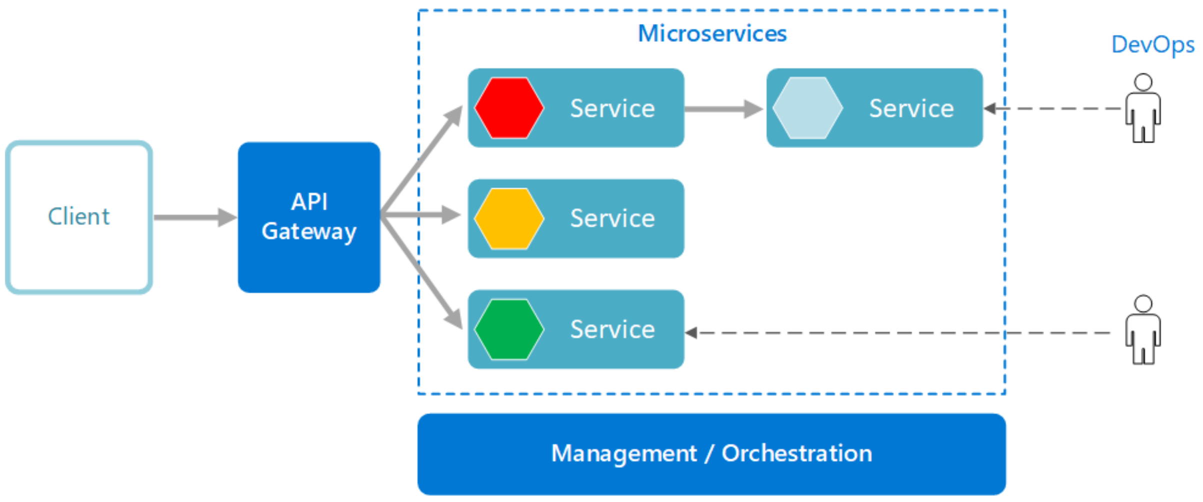
Req. No.	**Req. Name**	**Req. Description**
REQ_01	Process Pension module	This module is a Middleware Microservice that performs following operations: • Determines if it’s a self or family pension. Calculate the pension amount and bank service charge post data authentication, and display on the web application user interface • This module should receive input from the web application
REQ_02	Pensionerdetail 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	Authorization	This microservice is used with anonymous access to

	service	Generate JWT
REQ_04	Pension Management portal	A Web Portal that allows a member to Login and allows to do following operations: • Login • Load the pensioner detail through the Pensioner detail module • Post validation of the pensioner detail, pension amount should be displayed on the UI. This happens on invocation of ProcessPension module • 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 (Java)
   a. Spring Tool Suite (STS) Or any Latest Eclipse
   b. Have PMD Plugin, EclEmma Code Coverage Plugin
   c. Configure Maven in Eclipse
   d. Maven
   e. Docker (Optional)
   f. Postman Client in Chrome
   g. git

2.5 System Architecture Diagram



3.0 System Requirements

3.1.1 Functional Requirements – Process PensionMicroservice

Pension Management System	Process Pension Microservice
Functional Requirements Process Pension Microserviceshould be invoked from the web application. It allows the following operations: • It takes in Aadhaar number and determines the Pension amount and bank service charge • Verifies if the pensioner detail is accurate by getting the data from	

PensionerDetailMicroservice 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

ProcessPensionInput

1. **Aadhaar number**

PensionDetail

1. **PensionAmount**
2. **BankServiceCharge**

REST End Points

ClaimsMicroservice

POST: /ProcessPension(Input: processPensionInput| Output: PensionDetail)

Trigger – Should be invoked from Pension management portal

Steps and Actions

- This microserviceshould have 1 REST endpoint
- The POST endpoint should calculate the Pension for the person throught the Aadhaar number. It should 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 Pensioner detailmicroservicehas the bank detail. Process pension microservicecan have pre-defined list of banks and service charge as follows
  - Public banks – INR 500
  - Private banks – INR 550
- The PensionDetail object is returned to the web portal to display the data.

Non-Functional Requirement:

- Only Authorized requests can access these REST End Points

3.1.2 Functional Requirements – Pensioner detail Microservice

Pension Management	PensionerDetailMicroservice

System	

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

PensionerDetailMicroservice

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

Trigger – Should be invoked from ProcessPensionmicroservice

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 – 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 Endpoint Rest Endpoint 1.POST: /aunthicate: Username password need to generate jwt token and hit the other endpoints 2.POST: /registration For pensioner	

4.0 Reference learning

Other References:

Java 8 Parallel Programm	https://dzone.com/articles/parallel-and-asynchronous-programming-in-java-8

ing	
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

5.0 Change Log

	Changes Made			
V1.0.0	Initial baseline created on <16-Sep-2021> by			
	Section No.	**Changed By**	**Effective Date**	**Changes Effected**