|  | **Pension Management System**  **Case Study Specification**  **Node JS specific** |
| --- | --- |
|  |
|  |

**Table of Contents**

[**1.0**](#_heading=h.3znysh7) **Important Instructions 2**

[**2.0**](#_heading=h.2et92p0) **Introduction 4**

[**2.1**](#_heading=h.tyjcwt) **Purpose of this document 4**

[**2.2**](#_heading=h.1t3h5sf) **Project Overview 4**

[**2.3**](#_heading=h.4d34og8) **Scope 4**

[**2.4**](#_heading=h.2s8eyo1) **Hardware and Software Requirement 5**

[**2.5**](#_heading=h.17dp8vu) **System Architecture Diagram 6**

[**3.0**](#_heading=h.3rdcrjn) **System Requirements 6**

[**3.1.1**](#_heading=h.26in1rg) **Functional Requirements – Process Pension Microservice** 6

[**3.1.2**](#_heading=h.lnxbz9) **Functional Requirements – Pensioner detail Microservice** 7

[**3.1.3**](#_heading=h.35nkun2) **Functional Requirements – Authorization Microservice** 8

[**3.1.4**](#_heading=h.1ksv4uv) **Functional Requirements – Pension Management portal** 8

# Important Instructions

1. Associates must adhere to the Design Considerations specific to each Technology Track.
2. Associate must not submit projects 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 the 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.

# Introduction

## 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.

## Project Overview

State government aims to automate a portion of the Pension detail provisioning. This project covers pensioner detail provision, calculation provision and view for further processing.

## Scope

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

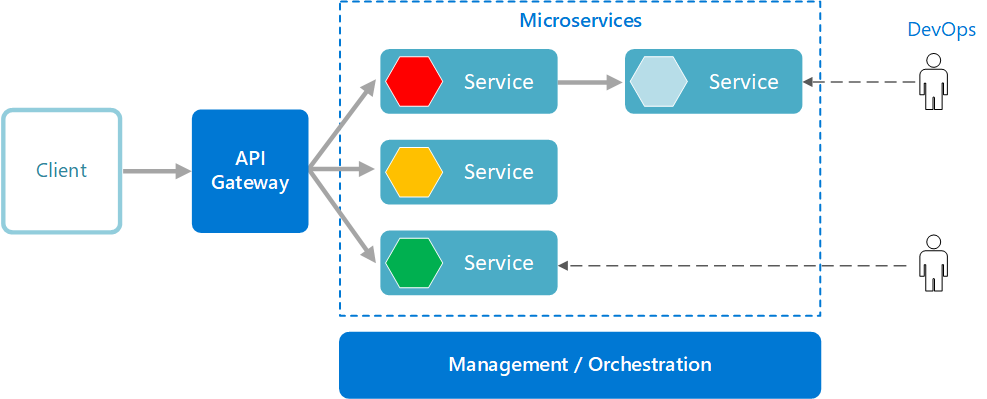
| **Req. No.** | **Req. Name** | **Req. Description** |
| --- | --- | --- |
| REQ\_01 | Process Pension module | This module is a Middleware Microservice that performs following operations:   * Determines if it’s a self or family pension. Calculate the pension amount and bank service charge post data authentication, and display on the web application user interface * This module should receive input from the web application |
| REQ\_02 | 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 | Authorization service | This microservice is used with anonymous access to 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 |

**The front-end application to be done on Angular/React/Vue.js**

## Hardware and Software Requirement

1. Hardware Requirement:
   1. Developer Desktop PC with 16 GB RAM
2. Software Requirement (Node JS)
   1. VS code Latest
   2. Node JS LTS / Express JS LTS
   3. GitBash CLI
   4. Docker (Optional)
   5. Postman Client

## System Architecture Diagram



# System Requirements

### **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 Aadhaar number and determines the Pension amount and bank service charge * 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**  **ProcessPensionInput**   1. **Aadhaar number**   **PensionDetail**   1. **PensionAmount** 2. **BankServiceCharge**   **REST End Points**  **Claims Microservice**  POST: /ProcessPension(Input: processPensionInput| Output: PensionDetail) | |
| **Trigger** – Should be invoked from Pension management portal | |
| **Steps and Actions**   * + This microservice should have 1 REST endpoint   + The POST endpoint should calculate the Pension for the person through 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 detail microservice has the bank detail. Process pension microservice can 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 Endpoints | |

### **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>   1. **Date of birth**   <Pensioner date of birth>   1. **PAN**   <Permanent account number>   1. **SalaryEarned**   <Last earned salary by the pensioner>   1. **Allowances**   <Sum of all the allowances>   1. **Self or Family pension**   <Is the pension classification self or family pension>   1. **Bank detail**    1. **Bank name**    2. **Account number**    3. **Public or Private bank**   <Bank detail>  **REST Endpoints**  **PensionerDetail** **Microservice**   * + GET: /Pensioner Details By Aadhaar (Input: aadhaarNumber | Output: pensionerDetail) | |
| **Trigger** – Should be invoked from ProcessPension microservice | |
| **Steps and Actions**   1. This Microservice is to fetch the pensioner details by the Aadhaar number. This should be consumed by Process pension microservice. 2. Flat file(CSV file with predefined 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 Endpoints | |

### **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 | |

### **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. | |

# 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.

**Design Consideration:**

1. There must be minimum of 2 to 3 Functional Microservices, must be deployed in AWS Cloud
2. Must have Parallel/Asynchronous Design
3. Configure Security using JSON Web Tokens
4. Service-Service communication must happen through service discovery.
5. Service must perform CRUD operations in a database
6. All the services must be independently deployable, must have its own database
7. Use Lombok Logging (Use Console Appender)
8. Use Junit, Mockito, Jacoco Code Coverage (Code coverage must be > 80%)
9. Use SONAR for Static code analysis (there must not be any high severity issues)
10. Access your application from local environment using Feign Client, in an MVC application and consume data
11. Use Swagger for Documentation purposes

**Other References:**

| Node JS Microservice | https://fauna.com/blog/how-to-build-microservices-with-node-js | |
| --- | --- | --- |
| Node JS JWT token | https://www.digitalocean.com/community/tutorials/nodejs-jwt-expressjs | |
| Setting up vs code | https://code.visualstudio.com/learn/educators/nodejs | |
| Testing in node js | https://www.testim.io/blog/node-js-unit-testing-get-started-quickly-with-examples/ | |
| Build | https://medium.com/swlh/step-by-step-building-your-first-node-js-project-45489f15aaa1 | |
| Swagger | https://www.section.io/engineering-education/documenting-node-js-rest-api-using-swagger/ | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  |
|  |  |
|  |  |