Member Registration Portal

Contents

[1.0 Problem statement 2](#_Toc65488937)

[2.0 Architecture Diagram for the Problem Statement 2](#_Toc65488938)

[3.0 Use case details 3](#_Toc65488939)

[4.0 Functional/Non-Functional Requirement of the Problem Statement 4](#_Toc65488940)

[5.0 Milestone 7](#_Toc65488941)

[6.0 Skills to develop the project 8](#_Toc65488942)

[7.0 Implementation Notes 8](#_Toc65488943)

[8.0 Evaluation rubrics 9](#_Toc65488944)

# Problem statement

The purpose of the requirements document is to systematically capture requirements for the project and the system “**Member Registration Portal**” to be developed. The application should be Cloud Native Architecture with Microservices. Both functional and non-functional requirements are captured in this document. It also serves as the input for the project scoping.

**About the System**

The client would like to develop an independent application Member Registration Portal (MRP) application in order to automate the process of registering a member, edit his details and submit his claims

**Scope of the System**

The scope of the system is explained through its modules as follows

* Member Registration – used by member to register the details of self-information into the system. The system stores the details of the member in the system and able to edit it.
* Submit claims – The user should be able to submit the claim for him and his dependent

# Architecture Diagram for the Problem Statement

**Use case Diagram**

**US\_01 Member Registration**

Member Registration

Member

Save to DB

**US\_02 Submit Claims**

Enter claim details

Registered Member

Save to DB

**US\_03 Update Member details**

Update the member/dependent details

Registered Member

Save to DB

# Use case details

|  |  |  |
| --- | --- | --- |
| **User Story #** | **User Story Name** | **User Story** |
| US\_01 | Member Registration | As a Member, I should be able to register my details in the system.  Acceptance criteria:  Member should be able register the details in the system and it should be saved in the database.  Capture the details like Name, Address, State, Country, Email Address, PAN, Contact No, DOB. |
| US\_02 | Update the member details | As a Member, I should be able to update the member details, dependent details |
| US\_03 | Submit claims | Member should be able to submit his claim details in the db. |

# Functional/Non-Functional Requirement of the Problem Statement

US -01 Member Registration

Business Rules & Validations

1. Member id should be generated automatically during the time of registration and should be shown in the success message.
2. The customer name should contain only alphabets and space.
3. All fields are mandatory.
4. Contact number should be 10 digits.
5. Email id should contain @ and. symbols.
6. Member id should be in the format of ‘R-XXX’.XXX should be random numeric of 3 digits.
7. PAN number must be alpha numeric and no special characters are allowed, no space is allowed. PAN number must be 12.
8. Based on the DOB, age will be calculated.
9. Activation date can be calculated based on the registration date.
10. DOB(Date of Birth) should not be less than system date.
11. Age should be greater than 18.
12. Claim Number should be generated automatically when the user submits a claim and should be a numeric of 10 digits
13. The member should be able to add only 2 dependents and store the dependent name, DOB.

**Validations**

* All fields are mandatory.
* Based on the country, state must be populated in the dropdown automatically.
* DOB(Date of Birth) should not be less than system date
* Age should be greater than 18.
* Claim number should be generated automatically and should be a numeric of 10 digits
* Registration Date should not be lesser than system date.

Submit claims

|  |  |
| --- | --- |
| US\_02 | Submit claims |
| Description  The member should be able to submit his claims | |
| Input Parameters  Below are the input parameters.  Member name, DOB, date of admission, date of discharge, Provider name, total bill amount. | |
| Business Rules & Validations   * All fields are mandatory. * Member name should contains only characters and space * Member can submit claim for dependent also. * On entering the member/dependent name, the member/dependent DOB should be automatically populated and it should be a non-editable field. * Date of admission should be on or before date of discharge * Bill amount should be numeric only. * Claim Number should be generated automatically when the user submits a claim and should be a numeric of 10 digits | |

Member Update

|  |  |
| --- | --- |
| US\_03 | Member details Update |
| Description  Member should be able to update his details | |
| Input Parameter   * + Member should be able to update only his mail id, PAN number, state, address, contact number.   + Email id should be validated.   + PAN number should not contain any special characters.   + When a member has dependents their name, DOB should be prepopulated. DOB of the dependents can be edited. | |
| Business Rules & Validations   * Email id should contain @ and . Symbols * Contact number should be 10 digit number. * PAN number should not contain any special characters. * On entering the member/dependent name , the member/dependent details should be auto populated | |

**Service Requirements**

**US\_01 Member Registration**

Once the user enters the details, they should be sent to the POST method and saved in the db.

Mandatory fields should be validated as mentioned in the rules above and 400 exception response should be sent with the missing field details.

When the details are saved successfully, the service should response 200 ok along with success message.

If there are any exceptions while connecting/saving to DB, the service should throw corresponding error with error status as 500.

**US\_02 Submit claims**

To retrieve the member/dependent details on entering the name field , a GET method should be implemented to fetch the details.

Once the user enters the details, they should be sent to the POST method and saved in the db.

Mandatory fields should be validated as mentioned in the rules above and 400 exception response should be sent with the missing field details.

Claim Number should be generated automatically when the user submits a claim and should be a numeric of 10 digit

When the details are saved successfully, the service should response 200 ok along with success message.

If there are any exceptions while connecting/saving to DB, the service should throw corresponding error with error status as 500.

**US\_03 Member Update**

To retrieve the member/dependent details on entering the name field , a GET method should be implemented to fetch the details.

Once the user enters the details, they should be sent to the PUT method and saved in the db.

Mandatory fields should be validated as mentioned in the rules above and 400 exception response should be sent with the missing field details. When the details are saved successfully, the service should response 200 ok along with success message.

If there are, any exceptions while connecting/saving to DB. The service should throw corresponding error with error status as 500.

**Expected Deliverables**

The following deliverables are expected as outcomes:

* Application Code base
* Readme document on the complete application
  + Setup of the application
  + How to run the application
  + Any inference
  + Screenshot of UI results
* Reports:
  + Unit/Functional Test Report

# Milestone

The milestone for the project use is given below

|  |  |  |
| --- | --- | --- |
| Milestone | Duration (in weeks) | Topic |
| Milestone - 1 |  | Design and develop the UI for the application |
| Milestone -2 |  | Develop the required APIs for the application  C:\Users\366598\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\C9B327B2.tmp  Or Develop all the APIs with NodeJS. |
| Milestone -3 |  | Integrate service layer with UI component, Dockerize the application and Push your Docker images to an Amazon ECR repository |

# Skills to develop the project

List the Technology based on your respective technology stack, that will be used to development the project.

|  |  |
| --- | --- |
| Skill Stack | Java - AWS |
| Front end | Angular /React  Bootstrap/ CSS  JavaScript/ JQuery  Typescript  Karma/ Cypress/ Jest |
| Service End | Spring Boot, Spring MVC, JDK, Maven/ SonarQube/ Junit or NodeJS/Express JS |
| Database | MongoDB |
| AWS | AWS code commit  Amazon ECR repository |

# Implementation Notes

As per the project requirement modification can be done in the below table.

|  |  |
| --- | --- |
| Milestone -1 | * Implement user-stories using any one of the UI frameworks [Angular/React] * Design application with Minimum Backend or Mock backend as the main focus **for milestone-1** in on frontend skills * Implement Forms, databinding, validations * Use appropriate unit test framework. |
| Milestone -2 | **SpringBoot:**  Create Spring Boot REST Microservices to perform SAVE Operation using POST method.   * Use Microservices Architecture * Follow coding standards * Follow Standard project structure * Message input/output format should be in JSON (Read the values from the property/input files, wherever applicable). Input/output format can be designed as per the discretion of the participant * Database connections and web service URLs should be configurable. * Use browser / POST Man to invoke APIs * Run SonarQube for code quality. * Implement Junit for unit testing.   **NodeJS:**   * Create a NodeJS project code * Develop a POST API method * Use NodeJS/Express project structure. * Follow coding standards * Message input/output format should be in JSON (Read the values from the property/input files, wherever applicable). Input/output format can be designed as per the discretion of the participant * Database connections should be configurable. * Use browser / POST Man to invoke APIs * Write test cases and Implement unit testing. |
| Milestone -3 | * Integrate service layer with UI component. * Dockerize the application * Push your Docker images to an Amazon ECR repository |

# 8.0 Evaluation rubrics

|  |  |
| --- | --- |
| Angular | * Associate must have used Angular Components, Modules, Databinding, data validation, CLI commands. * Associate must have used Forms and Forms validation * Associate must have used Directives * Associate must have developed Reusable Components * Associate must have followed coding standards |
| REACT | * Associate must have used Component, Databinding, data validation, CLI commands. * Associate must have used Forms and Forms validation * Associate must have defined React state * Associate must have followed coding standards |
| Microservices | * REST controller * Follow controller ->service->Dao model * Entity and Model classes * Appropriate logging statements * Exception handling |
| Docker | * Dockerize the application * Build docker containers * Push your Docker images to an Amazon ECR repository with the docker push command |
| NodeJS/ExpressJS | * NodeJS, Helmet and Express JS should be used in the * API’s application * Advantage of using NodeJS, Projections in Mongo Queries over other frameworks. |
| AWS | * Code is committed in AWS * Push your container images to Amazon Elastic Container Registry |