Automated Premiums & Claims Management System

Contents

[1.0 Problem statement 2](#_Toc42538268)

[2.0 Skill Tower develop the project 2](#_Toc42538269)

[3.0 Use Case/Architecture Diagram 3](#_Toc42538270)

[4.0 User Stories 4](#_Toc42538271)

[5.0 Functional/Non Functional Requirement of the Problem Statement 5](#_Toc42538272)

[6.0 Expected Deliverables 12](#_Toc42538273)

[7.0 Milestone and duration 12](#_Toc42538274)

[8.0 Implementation Notes 12](#_Toc42538275)

[9.0 Evaluation rubrics 13](#_Toc42538276)

# Problem statement

The purpose of the requirements document is to systematically capture requirements for the project and the system “**Automated Premiums & Claims Management system**” to be developed. 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 Automated Premiums & Claims Management application in order to automate the process of managing the activities of Premium Collections and approving Claims.

The following section will cover aspects related to Automated Renewal System.

1. Premium Calculation & Process Payment.
2. File a Claim
3. Process Claims and Send Confirmation.

**Scope of the System**

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

* Premium Calculation & process Payment: Used to calculate the premiums based on the Opted plans and send the amount details to customer. Once customer makes the payment, the transaction details has to be stored and customer’s account had to be top uped.
* File a Claim: Based on the eligibility criteria, user will be allowed to file a claim with the necessary details.
* Process claims & send Confirmation: After processing the claim submitted by the customer, based on some predefined rules, claim will be processed and a confirmation mail will be sent to the customer via email.

# Skill Tower develop the project

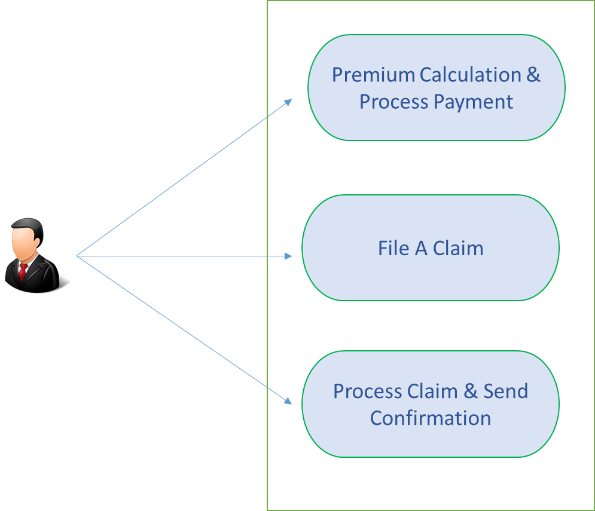
List the Technology based on your respective technology stack, that will be used to development the project.

Associate will choose any one of the technology stack and develop the application.

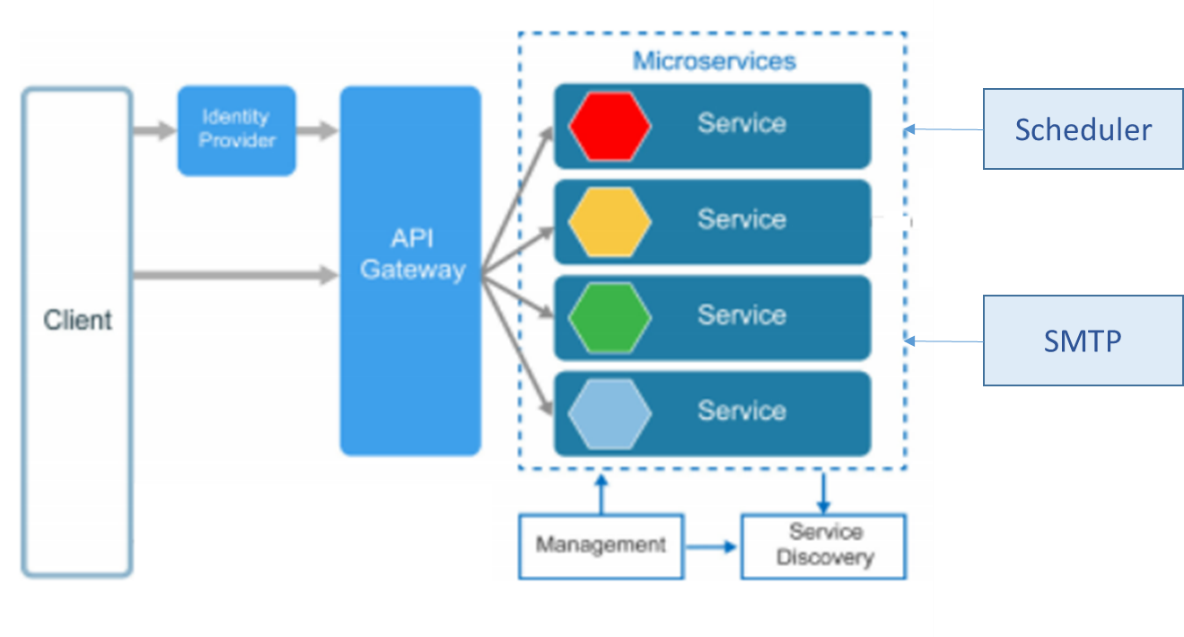
|  |  |
| --- | --- |
| Tower Name | Skill Names |
| Backend - Java | Core Java 8, Junit  Spring Boot  Spring Cloud, JSON  Microservice  Data Structures |
| Backed - .NET | .Net Core, C#, NUnit  EF core  ASP.NET Web API Core, JSON  Microservices  Ocelot API Gateway or any equivalent  Data Structures |
| Backend – MEAN | Data Structures  Node JS  Express JS  Rest API, JSON  Microservice  Mocha/Chai  API Gateway |
| Backend - Python | Data Structures  Python  Rest API  Microservices  JSON  API Gateway |

# Use Case/Architecture Diagram

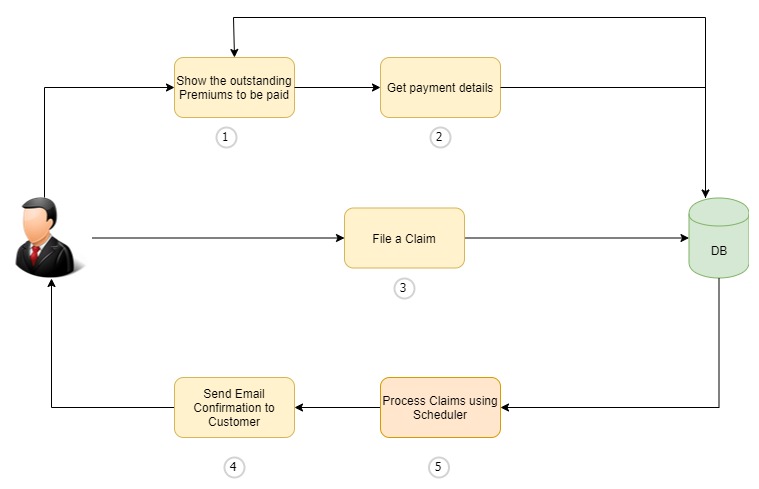
**Use Case Diagram:**



**Sample Architecture Diagram:**



**Flow Diagram:**



**Explanation:**

* Design a Solid Database with Customer Details, Policy Detail, Premium & Claim Details, Transaction details.
* Design an App with Micro Service architecture along with service discovery & API Gateway etc.
* Items marked as 1,2,3 & 4 are API services that needs to be created.
* Scheduler to process Claims & an SMTP implementation to send out emails.

# User Stories

|  |  |  |
| --- | --- | --- |
| **User Story #** | **User Story Name** | **User Story** |
| US\_01 | Premium Calculation & Process Payment | As a customer I should be able to View the outstanding premium payments due for this month for various policies that I have taken.  Also, upon selecting the policy and premium that I wish to make the payment for, I should be able to successfully make the payment.  **Acceptance criteria:**  Retrieve the Policy details from the database and show the customers the list of premiums due for that Month.  Upon Selecting and making the payment, store the transaction details in the transaction table and update the Premium table with the paid amount. |
| US\_02 | File a Claim | As a Customer, I should be able to file a claim against a particular policy.  **Acceptance criteria:**  Customers should be able to file a claim against a policy by providing the necessary details like Claim amount, reason, date, Additional Comments & Proof Document. |
| US\_03 | Process Claims & Send Confirmation | As a Customer, I should be able to receive an update via Email once the claim is processed.  **Acceptance criteria:**  There must be a batch process established to process the outstanding claims based on certain rules and then send an update to the customer via the registered email address. |

# Expected Deliverables

The following deliverables are expected as outcomes

* Application Code base
* API for each service
* Readme document on the complete application
  + Setup of the application
  + High level steps used to convert to server less architecture
  + How to run the application
  + Any inference
  + Snapshot of any implementation
* Reports:
  + Functional Test Report
  + Code Profiling Report
  + Other Reports as applicable

# Milestone and duration

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

|  |  |
| --- | --- |
| Milestone | Topic |
| Milestone -1 | Developing the API/Microservice, Coding Standings, 12 Factor Principles |
| Milestone -2 | Service Discovery, Registry, Circuit Breaker, API gateway, Testing the API, Code Quality Metrics |

# Implementation Notes

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

|  |  |
| --- | --- |
| Milestone -1 | * Use Rest APIs to develop the services * Use Microservice Architecture * Use Domain Driven Design * Implement repository pattern * Use Swagger definitions * Use browser / POST Man to invoke APIs * Use ORM to work with database, under repository pattern * Use browser / POST Man to invoke APIs * Implement API Versioning * Implement API Gateway * User access security microservice to allow/disallow CRUD operations * 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 * Any error message or exception should be logged to the user should be user-readable (and not technical) * Database connections and web service URLs should be configurable. * Web service URLs should be configurable. * Implement External Configurable Solution * Implement 12 Factor principles |
| Milestone -2 | * Implement Unit Test Project for testing the API * Implement Service Discovery, Registry, Circuit Breaker   All implementation should publish Code Quality Metrics   * Technical Debt –lower-the-better * Code Smell –lower-the-better * Code Coverage –higher-the-better * Secure coding practices * Follow coding standards |

# Evaluation rubrics

|  |  |
| --- | --- |
| Microservices | * Follow the below basic structure   + API - Controllers   + Domain - Model, Events, Business Services Integration   + Services – API Implementation   + Infrastructure Project * Associate must have designed/developed individual Microservices for each functionality. * Each of the Microservices need to comprise below functionality, which need to be developed   a. REST Controllers  b. Services  c. Entity & Model classes, including appropriate relationship (like One-One, Many-One, etc…) between Entity Classes. (Entity and Model classes have been developed in the Previous Phase)  d. In case specific Entity or Model classes are required across multiple Microservices, it is recommended to maintain separate copy of Entity or Model classes for each Microservices.  e. Microservices should interact with corresponding DB tables or Databases it owns.  f. Microservice need to interact with other Microservice  h. Usage of Postman to test the Microservices by directly passing requests to each REST end Point, of each Microservice  i. Unit Testing code should be developed using NUnit/Xunit and perform Unit Testing  j. Circuit Breaker, Service Registry, Service Discovery should be implemented   * Use Swagger UI and test each public method in the service * Implementation of Repository pattern |
| Rest API | * Associate must have used REST API for exposing resources * Associate must have used HTTP GET/PUT/POST request method designators for the business methods which is to be exposed * Associate must have customized the request and response formats according to the requirement * Associates must have used appropriate RETURN CODES based on the service outcome * Associates must have extracted query/form/header parameters from the input * Associate must have built a custom response based on the input |
| Java/C#/Node/Python | * Associate should have used appropriate Base class Libraries, Control Statements and Operators, File Handling and I/O Operations for implementing the functionalities. |
| Unit Testing | * Test cases covers the functionality of API with custom inputs * Good test Coverage |
| Common | * Code Smell * Technical Debt * Secured Coding * Coding Standards |