Contents

[1.0 Important Instructions 2](#_Toc131151846)

[2.0 Introduction 3](#_Toc131151847)

[2.1 Purpose of this document 3](#_Toc131151848)

[2.2 Project Overview 3](#_Toc131151849)

[2.3 Scope 4](#_Toc131151850)

[2.4 Hardware and Software Requirement 5](#_Toc131151851)

[2.5 System Architecture Diagram 5](#_Toc131151852)

[3.0 Functional Requirements and High Level Design 6](#_Toc131151853)

[3.1 Use Case Diagram 6](#_Toc131151854)

[3.2 Individual Components of the System 7](#_Toc131151855)

[**3.2.1** **Consumer Microservice** 7](#_Toc131151856)

[**3.2.2** **Policy Microservice** 9](#_Toc131151857)

[3.2.3 **Quotes Microservice** 10](#_Toc131151858)

[**3.2.4** **Authorization Microservice** 11](#_Toc131151859)

[3.2.5 **Swagger** 11](#_Toc131151860)

[**3.2.6** **Insureity Portal (Angular)** 12](#_Toc131151861)

[4.0 Reference learning 12](#_Toc131151862)

[5.0 Change Log 13](#_Toc131151863)

# 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 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

(Importantly, the READ ME should contain the steps to execute DB scripts, the LAUNCH URL of the application)

1. 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 “Policy Administration 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

A leading Insurance Management Organization offers Business insurance that protects businesses from losses due to events that may occur during the normal course of business. There are many types of [insurance](https://www.investopedia.com/terms/i/insurance.asp) it provides for businesses including coverage for property damage, legal liability and employee-related risks.

Companies evaluate their insurance needs based on potential risks, which can vary depending on the type of environment in which the company operates. Business insurance refers broadly to a class of insurance coverage intended for purchase by businesses rather than individuals.

Quotes will get generated based on the consumer business and if the consumer is fine with the quote, then the policy will be issued. A quote is an estimate of premium for the insurance coverage you selected and information you entered.

In this project we focus only on the Business Property Insurance that covers building, equipment, signage, inventory, and furniture in the event of a fire, storm or theft.

The Insurance Organization operates over Insurance Agents to handle the Policy Subscription. An insurance agent is a licensed professional who sells an insurance company’s products to consumers for a commission. An agent helps consumers select the right insurance to buy, but represents the Insurance Organization in the transaction.

Insureity is the Portal that allows any Agents to Login and create Policies for the consumers.

This Portal in turn interacts with the Middleware system of the Organization which will allow the Agents to create policies based on quotes.

## Scope

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

|  |  |  |
| --- | --- | --- |
| **Req. No.** | **Req. Name** | **Req. Description** |
| REQ\_01 | Consumer Module | Consumer Module is a Middleware Microservice that performs following operations:   * Create Consumer Business * Update Consumer Business * View Consumer Business * Create Business Property * Update Business Property * View Business Property |
| REQ\_02 | Policy Module | Policy Module is a Middleware Microservice that performs the following operations:   * Create Policy * View Policy * Issue Policy |
| REQ\_03 | Quotes Module | Quotes Module is a Middleware Microservice that performs the following operations:   * Get Quotes |
| REQ\_04 | Insureity - Agent Portal | An Web Portal that allows an Agent to Login and allows to do following operations:   * Login * Create Policy * Issue Policy |

## Hardware and Software Requirement

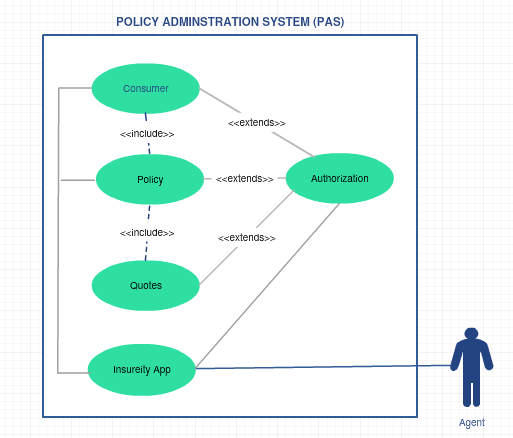
1. Hardware Requirement:
   1. Developer Desktop PC with 8GB RAM
2. Software Requirement
   1. Visual studio 2017 enterprise edition
   2. SQL Server 2014
   3. Postman Client in Chrome

## System Architecture Diagram



# Functional Requirements and High Level Design

## Use Case Diagram



## Individual Components of the System

### **Consumer Microservice**

|  |  |
| --- | --- |
| Policy Administration System | Consumer Microservice |
| **Functional Requirements**  An Agent through Insureity App can interact with Consumer Service to create new consumer / update consumer details. Consumer is the customer here who wishes to subscribe to a Business Property Insurance with the Insurance Company. In this project we are restricting to Property Insurance that means the land/building in which the business is operated will be covered.  The Consumer Microservice has to interact with Policy Microservice to create a policy and as well as to view the policy status  Post Authorization, the Consumer Microservice will perform the following functionalities:   * Create Consumer Business * Create Business Property * View Consumer Business * View Business Property * Update Consumer Business * Update Business Property | |
| **Entities**   1. **Consumer**   <Details of Consumer, Personal Details, Business Overview, Validity of Consumer, Agent Details who subscribes the Consumer>   1. **Business**   <Details of Business including, Business Type, Annual Turnover, Total Employees, etc.>   1. **Property**   <Business ID, Building Sq.ft, Building Type (Owned/Rental), Building Storeys Building Age>   1. **Property Master**   <Has Property types, rules permissible for Insurance>   1. **Business Master**   <Has Business types, rules permissible for Insurance>  **REST End Points**  **Consumer Microservice**   * + POST: /createConsumerBusiness (Input: Consumer Details like name, business type, consumer DOB, email, PAN #, Business Turnover, Total Employees, Agent Details etc.. | Output: Status)   + POST: /updateConsumerBusiness (Input: Consumer ID, Consumer Details like name, business type, consumer DOB, email, PAN #, Business Turnover, Total Employees etc.. | Output: Status)   + POST: /createBusinessProperty (Input: Consumer ID, Property Type, Property Age, Storeys etc. | Output: Status)   + POST: /updateBusinessProperty (Input: Consumer ID, Property ID, Property Type, Property Age, Storeys etc. | Output: Status)   + GET: / viewConsumerBusiness (Input: Consumer ID) | Output: Consumer Details like name, business type, consumer DOB, email, PAN #, Business Turnover, Total Employees etc.)   + GET: / viewConsumerProperty (Input: Consumer ID, Property ID) | Output: Property ID, Property Type, Property Age, Storeys etc.) | |
| **Trigger** – Can be invoked from Insureity Portal App | |
| **Steps and Actions**   1. Authorization has to be performed 2. Basic Input Validations have to be performed. Every Consumer Business must be in the Permissible Business Category and Property must be in permissible Property Category    * Example, Business Owner must be operating business in the Property for minimum of 2 years for a Theft Cover 3. Business Value and Property Values will be calculated based on the rules configured in Master Entities. Rules are given below:    * Business value will be given an index from 0 to 10, 0 being lowest and 10 being highest based on the annual turnover vs. capital invested    * Property Value will be given an index from 0 to 10 based on its depreciation value like below:     Where Cost of the asset is the purchase price of the asset,  Salvage value is the value of asset at the end of its useful life  Useful Life of the Asset represents the number of years in which the asset is used by the Business Consumer | |
| **Non-Functional Requirement:**   * Only Authorized Member can access these REST End Points | |

### **Policy Microservice**

|  |  |
| --- | --- |
| Policy Administration System | Policy Microservice |
| **Functional Requirements**  Policy Microservice interacts with Consumer Microservice and Quotes Microservice, but it gets invoked from Insureity Portal (angular app). Post authorization of request, Policy Microservice allows the following operations:   * Creates Policy based on Business and Property Value from Consumer Service. For these values, the service will validate the permissible Policies of type “Property Insurance” and for the coverage the Agent has quoted, and get the quotes from Quotes service. * Issue Policy will issue a policy to the consumer based on the payment and acceptance status from consumer, for the quoted insurance cover. * View Policy allows to view the Policy Details | |
| **Entities**   1. **Policy Master**   <Types of permissible Policies for the type of business and the type of business value>  Example:   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | ID | Property-Type | Consumer Type | Assured Sum | Tenure | Business Value | Property Value | Base Location | Type | | P01 | Building | Owner | 2,00,00,000 | 3 years | 8 | 5 | Chennai | Replacement | | P02 | Factory Equipment | Owner | 4,00,000 | 1 year | 9 | 10 | Chennai | Replacement | | P03 | Property in Transit | Owner | 2,00,000 | 1 week | 7 | 8 | Pune | Pay Back |  1. **Consumer Policy**   <Policies created for every consumer>  **REST End Points**  **Policy Microservice**   * + POST: /createPolicy (Input: Consumer Details, Business Details, Accepted Quotes, Agent Details | Output: Policy Status, Description)   + POST: /issuePolicy (Input: Policy\_ID, Consumer ID, Business ID, Payment Details, Acceptance Status) | Output: Issue Status, Status Description)   + GET: / viewPolicy (Input: Consumer ID, Policy ID) | Output: Policy Details)   + GET: /getQuotes (Input: Business Value, Property Value | Quotes details) | |
| **Trigger** – Will be invoked from Insureity Portal. | |
| **Steps and Actions**  Create Policy   * For the Consumer and Business Details, get the concrete details from Consumer Microservice * For the Business Value find appropriate policy based on policy rules in Policy Master Entity * Request Quotes from Quote Service for the Business Value * Create Policy with Status “Initiated” * Assume that Agent details / Consumer Business Details will not be verified in the scope of this Policy Microservice.   Issue Policy:   * Invoked from Insureity App when Agent clicks on Issue Policy in the Portal, with Payment Status and Policy Acceptance Status by Business * Policy Status will get updated as “Issued”, Update the Quote on which the Policy was approved * Policy Effective Date, Covered Sum, Duration gets updated. * Assume that Payment of Policy Premium happens outside the portal and Agent   View Policy:   * Policy Details will be returned for the given Business and Consumer ID | |

### **Quotes Microservice**

|  |  |
| --- | --- |
| Policy Administration System | Quotes Microservice |
| **Functional Requirements**  Quotes Microservice will be invoked only from Policy Service. Post authorization of request; based on the business value and property value, for the opted policy quotes will be calculated. The service exposes the following operation:   * Get Quotes – for the given consumer business and property value, policy type, policy covered sum etc. | |
| **Entities**   1. **Quotes Master**   Holds the Quote values for the different policies, for different ranges of Business Value and Property Value  Ex.: If Business value is 0-2 AND Property Value is 0-2, and Property Type is “Equipment” then quotes can be 80,000 INR | If Business value is 3-5 AND Property Value is 3-5 and Property Type is “Equipment” then quotes can be 50,000 INR  Note: Always the quotes are for the insurance period given in the Policy Table  **REST End Points**  **Quotes Microservice**   * + GET: /getQuotesForPolicy (Input: Policy, with all value ranges and type | Output Quotes) | |
| **Trigger** – Can be invoked from Policy Portal | |
| **Steps and Actions**   * + The policy directly hits the Quotes Table and returns quotes   + If no quotes are available, then an appropriate response will be returned with description “No Quotes, Contact Insurance Provider”. | |

### **Authorization Microservice**

|  |  |
| --- | --- |
| Policy Administration System | Authorization Microservice |
| **Security Requirements**   * Service to Service communication has to happen using JWT * Pass End User Context across Microservices * Have the token expired after specific amount of time say 15 minutes. | |

### **Swagger**

|  |  |
| --- | --- |
| Policy Administration System | Swagger |
| **Documentation Requirements (Java)**   * All the Microservices must be configured with Swagger for documentation * Register the swagger resources in the Swagger Microservice and enable them as REST end points | |

### **Insureity Portal (Angular)**

|  |  |
| --- | --- |
| Policy Administration System | Insureity Portal |
| **Insureity Portal Requirements**   * Portal must allow the Agents to Login. Once successfully logged in, the member do the following operations:   + Create / Edit / View Consumer Businesses and Properties   + Create Policy   + Issue Policy – Based on Payment Information and Acceptance by Customer, policy will be issued. However Payment and Acceptance is outside the system boundary. Authorized Agents will get the information and that will be fed into the Portal by the agents   + View Policy | |

# Reference learning

Please go through all of these k-point videos for Microservices deployment into AWS.

**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](https://dzone.com/articles/microservices-communication-feign-as-rest-client) |
| Swagger (Optional) | [https://dzone.com/articles/centralized-documentation-in-Microservice-spring-b](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 .Netcore 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> |

# Change Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Changes Made | | | |
| V1.0.0 |  | | | |
|  |  | | | |
| **Section No.** | **Changed By** | **Effective Date** | **Changes Effected** |
|  |  |  |  |