1. Introduction

1.1 Purpose of this document

The purpose of the software requirement document is to systematically capture requirements for the project and the system "Real Estate 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.

1.2 Project Overview

One of the largest and leading Real Estate Company within the UK, serving millions of customers across the country offering real estate related activities like buy/sell different types of properties like Flats / Villas / Plots etc. The real estate company has historically been served by a large monolith system. This system has Customer information, Properties information, Agents information, Customer Requirements. This company is looking for a solution that will provide resilience & scalability for future growth. Following are the required features:

- Highly available
- Highly scalable
- Highly Performance
- Easily built and maintained
- Developed and deployed quickly

1.3 Scope

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

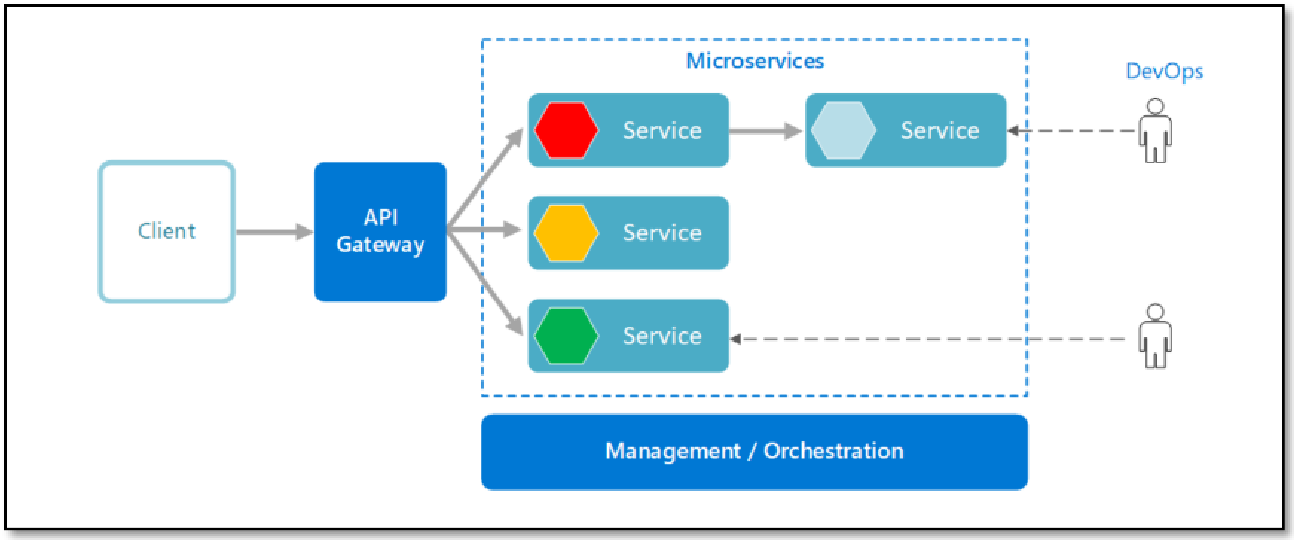
Req. No.	Req. Name	Req. Description
REQ_01	Customer Module	Customer Module is a Microservice that performs the following operations: • Create Customer • Get Customer Details
REQ_02	Property Module	Property Module is a Microservice that performs following operations: • Create New Property • Get All Properties • Get Properties By Type • Get Properties By Locality
REQ_03	Manager Module	Manager Module is a Microservice that performs the following operations: • Get All Executives • Add Executive • Get Executives By Area • Assign Executive to Customer
REQ_04	Authentication Module	Authentication Module is a Microservice that performs the following operations: • Login • Logout

REQ_05	UI Portal	Loads the UI and takes care of user sessions. Relies on all other Microservices for core functionality.

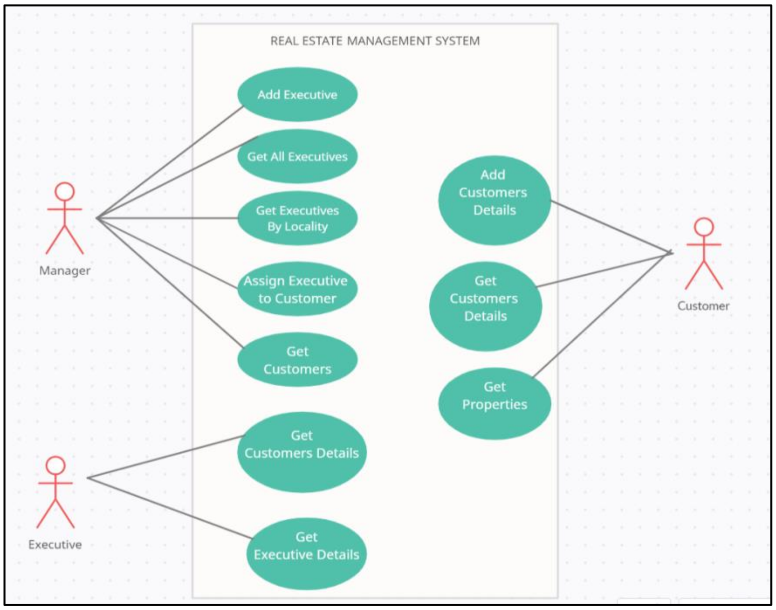
1.4 Hardware and Software Requirement

1. Hardware Requirement:
   a. Developer Desktop PC with 8GB RAM
2. Software Requirement (Java)
   a. Spring Tool Suite (STS) Or any Latest Eclipse
      i. Have PMD Plugin, EclEmma Code Coverage Plugin and AWS Code Commit Enabled
      ii. Configure Maven in Eclipse
   b. Maven
   c. Docker (Optional)
   d. Postman Client in Chrome
   e. AWS Cloud Account
3. Software Requirement (Dotnet)
   a. Visual studio 2017 enterprise edition
   b. SQL Server 2014
   c. Postman Client in Chrome
   d. Azure cloud access

1.5 System Architecture Diagram



1.6 Use case Diagram



2. System Requirements

1. Customer Microservice

Real Estate Management System	**Customer Microservice**
Functional Requirements The Customer microservice will perform the following tasks 1. Creating a customer profile based on the details provided by the customer 2. Displaying customer’s profile 3. Search for properties	
Entities	

1. **Customer**
   <Details of Customer: Name, Address, EmailId, Contact_Number, Requirement…>
2. **Requirement**
   <Details of Customer Requirement: PropertyType, Budget, Locality >

REST End Points Customer Microservice

- POST: /createCustomer (Input: Customer | Output: Status)
- GET: /getCustomerDetails (Input: Customer_Id | Output: Customer)
- GET: /getProperties (Output: List of properties)

Trigger – Customer creation will be done by the customer from the Real Estate Portal after inputing in his details

Steps and Actions

1. The Customer inputs Customer information within the Portal
2. createCustomer on the Customer Microservice will be called by passing the customer information.
3. Manager query on the customer to get information about the customer
   - Customer Details

Non-Functional Requirement:

- Only Authorized Members can access these REST End Points

Clients for this service:

Manager Service
UI Portal

2. Property Microservice

Real Estate Management System	Property Microservice

Functional Requirements

The Property microservice will perform the following tasks

1. Create New Property
2. Get All Properties
3. Get Properties By Type
4. Get Properties By Locality

Entities

1. Property
<Details of Property: PropertyType, Locality, Budget, ..>

REST End Points of Property Microservice

- POST: /createProperty (Input: Property| Output: Status)
- GET: /getAllProperties(Output: List of Properties)
- GET: /getAllPropertiesByType(Input: PropertyType | Output: List of Properties)
- GET: /getAllPropertiesByLocality(Input: Locality| Output: List of Properties)

Trigger – Property creation will be done by the manager from the Real Estate Portal after inputing the details

Steps and Actions

1. The Manager inputs Property information within the Portal to create new property
2. The Customer / Execute / Manager can get the properties details from the Portal.

Non-Functional Requirement:

- Only Authorized Members can access these REST End Points

Clients for this service:

Customer Service
UI Portal

3. Manager Microservice

Real Estate Management System	**Manager Microservice**

Functional Requirements

The Manager microservice will perform the following tasks

1. Get All Executives
2. Add Executive
3. Get Executives By Locality
4. Assign Executive to Customer
5. Get All Customers
6. Get Customer Details By Id

Entities

1. Executive

<Details of Executive: Name, ContactNumber, Locality, EmailId, ..>

REST End Points of Manager Microservice

- POST: /createExecutive (Input: Executive| Output: Status)
- GET: /getAllExecutives(Output: List of Executives)
- GET: /getAllExecutivesByLocality(Input: Locality | Output: List of Executives)
- PUT : / assignExecutive(Input:CustomerId | Output: Status)
- Get: / getAllCustomers(Output: List of Customers)
- Get: / getCustomersById(Input : CustomerId | Output : Customer)

Trigger – Executive creation and assigned executive will be done by the manager from the Real Estate Portal after inputing the details.

Steps and Actions

- The Manager inputs Executive information within the Portal to create new executive
- The Manager get the details of Executives
- The Manager assign the executive to customer

• Get customer details
Non-Functional Requirement: • Only Authorized Members can access these REST End Points
Clients for this service: **UI Portal**

4. Authentication Microservice

Real Estate Management System	**Authentication 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	

5. UI Portal – Real Estate Management Portal

Real Estate Management System	**Real Estate Management Portal**
Client Portal Requirements • Real Estate Management Portal must allow the Manager/Execute/Customer to Login. Once successfully logged in, the member do the following operations: • Manager can perform the following operations: 1. Create Properties 2. Create Executive 3. Get Property details 4. Get Customer Details 5. Assign Executive to Customer • Customer can perform the following operations: 1. Register his details on the portal 2. Search for properties • Executive can perform the following operations: 1. Get executive details 2. Get Customers Assigned to him • Each of the above operations will reach out to the middleware Microservices that are hosted in cloud.	

6. Swagger Documentation

Real Estate Management System	Swagger
Documentation Requirements • All the Microservices must be configured with Swagger for documentation	

3. 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 user sessions 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 Azure Kubernetes or AWS ECS Services.
- These services must be consumed from an MVC app running in a local environment.

4. Reference learning

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
Swagger (Optional)	https://www.c-sharpcorner.com/article/using-swagger-in-asp-net-core/
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