|  |  |
| --- | --- |
|  | **Cognizant Academy**  **International patient treatment management System**  **FSE – Business Aligned Project**  **Case Study Specification**  **Version 1.0** |
| |  |  |  |  | | --- | --- | --- | --- | |  | **Prepared By / Last Updated By** | **Reviewed By** | **Approved By** | | **Name** | Seshadri M R |  |  | | **Role** | Solution Designer |  |  | | **Signature** |  |  |  | | **Date** |  |  |  | |
|  |

Table of Contents

[1.0 Important Instructions 3](#_Toc46778402)

[2.0 Introduction 4](#_Toc46778403)

[2.1 Purpose of this document 4](#_Toc46778404)

[2.2 Project Overview 4](#_Toc46778405)

[2.3 Scope 4](#_Toc46778406)

[2.4 Hardware and Software Requirement 5](#_Toc46778407)

[2.5 System Architecture Diagram 6](#_Toc46778408)

[3.0 System Requirements 6](#_Toc46778409)

[**3.1.1** **Functional Requirements – IP treatment offerings** **Microservice** 6](#_Toc46778410)

[**3.1.2** **Functional Requirements – IPTreatment Microservice** 7](#_Toc46778411)

[**3.1.3** **Functional Requirements – InsuranceClaim** **Microservice** 9](#_Toc46778412)

[**3.1.4** **Functional Requirements – Authorization Microservice** 10](#_Toc46778413)

[**3.1.5** **Functional Requirements – IP treatment management portal** 10](#_Toc46778414)

[4.0 Cloud Deployment requirements 11](#_Toc46778415)

[5.0 Design Considerations 11](#_Toc46778416)

[6.0 Reference learning 11](#_Toc46778417)

[7.0 Change Log 12](#_Toc46778418)

# Important Instructions

1. Associate must adhere to the Design Considerations specific to each Technolgy 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 Unit testing and 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.
7. A READ ME has to be provided with steps to execute the submitted solution, the Launch URLs of the Microservices in cloud must be specified.

(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.
2. You are supposed to use an In-memory database or code level data as specified, for the Microservices that should be deployed in cloud. No Physical database is suggested for Microservice.

# Introduction

## Purpose of this document

The purpose of the software requirement document is to systematically capture requirements for the project and the system “Hospital 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 specificed wherever applicable, however the detailed design considerations have to be strictly adhered to during implementation.

## Project Overview

A leading hospital wants to bring up a web application to list and choose the treatment packages for International patients, register them, track their treatment to closure thru insurance claim.

## Scope

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

|  |  |  |
| --- | --- | --- |
| **Req. No.** | **Req. Name** | **Req. Description** |
| REQ\_01 | International patient treatment offerings module | This Module is a Middleware Microservice that performs following operations:   * Provides a list of treatment packages available at the hospital for international patients * Provides the list of specialists with their experience and contact detail * Enables the admin to update packages. * Enables the admin to add new specialist details * Enables the admin to delete a specific specialist. * Provides the list of specialists by their area of expertise. |
| REQ\_02 | International patient treatment module | This Module is a Middleware Microservice that performs the following operations:   * For a new international patient, based on the ailment, a time-table of consultations by specialists should be provided |
| REQ\_03 | Insurance claim module | This Module is a Middleware Microservice that performs the following operations:   * Admin should be able to choose the patient whose treatment is in-progress and can be marked as complete. * Gets the list of insurance agencies and its available limits and duration for claim processing. * Based on the choice of insurer from the admin, all the necessary detail from hospital should be furnished to the insurer to process the claim. |
| REQ\_04 | Authorization service | This microservice is used with anonymous access to Generate JWT |
| REQ\_05 | International patients treatment management portal | A Web Portal that allows a member to Login and allows to do following operations:   * Login * View the list of treatment offerings for International patients * Register the patient detail and get the treatment detail by choosing the relevant treatment package. Treatment status should be marked as ‘in-progress’ * View the list of registered patients to complete the treatment and proceed with insurance claim. View the list of insurers to choose the appropriate insurer. Provide the required patient detail to initiate the insurance claim. Upon successful insurance claim initiation, balance amount to be paid should be shown on the web application UI. |

Note: The project phase is for 2 weeks. The first week is to be developed on local machine and the second week deals with Cloud deployment.

The requirement details given below states in-memory database usage. **The first phase of the development which is done in the first week, SHOULD use the Database for related activities and NOT the in-memory database.**

The second phase of the development which is done in the second week, can use the in-memory database as mentioned in the requirement, with appropriate code modifications.

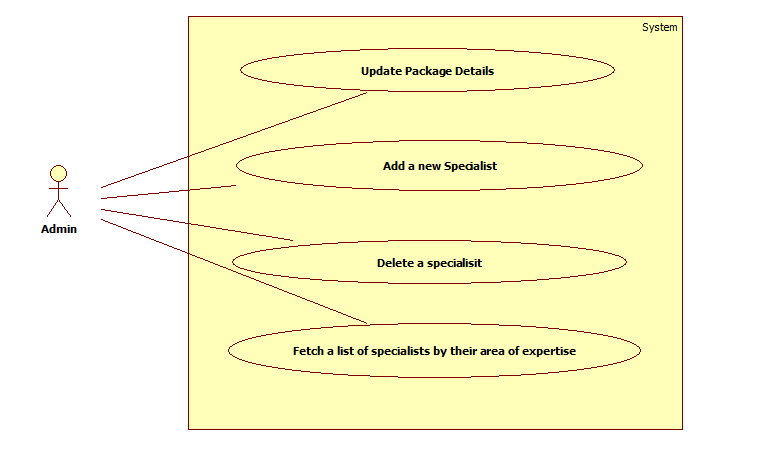
## Hardware and Software Requirement

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

## System Architecture Diagram



**Use cases- Enhancement Requirements in International patient treatment offering**s **module**



# System Requirements

### **Functional Requirements – IP treatment offerings** **Microservice**

|  |  |
| --- | --- |
| International patients treatment management system | IPTreatmentOffering Microservice |
| **Functional Requirements**  The intent of this Microservice is to provide the   * List of service packages for international patients * List of specialists along with their experience and contact detail * Enables the admin to update treatment packages available for international patients. * Enables the admin to add new specialist details for International patients. * Enables the admin to delete a specific specialist. * Provides the list of specialists by their area of expertise. | |
| **Entities**  **IPTreatmentPackages**   1. **AilmentCategory**   <Orthopaedics, Urology>   1. **PackageDetail**    1. **TreatmentPackageName**    2. **Test details**    3. **Cost**    4. **Treatment duration**   **SpecialistDetail**   1. **Name**   <Specialist name>   1. **AreaOfExpertise**   <Domain. Eg: Orthopaedics, Urology>   1. **ExperienceInYears** 2. **ContactNumber**   **REST End Points**  **InpatientServices Microservice**   * + GET: /IPTreatmentPackages (Input: None | Output: List of InpatientServicePackages)   + GET: / IPTreatmentPackageByName(Input: packageName | Output: InpatientServicePackage)   + GET: /Specialists   + GET: /specialistsByExpertsise( Input: AreaOfExpertise |Output: List of Specialists)   + POST: /addSpecialist(Input: SpecialistDetail | Output : POST status)   + DELETE:/ deleteSpecialist(Input: Specialist\_id | Output : Delete Status)   + PUT: /updatePackage(Input: packageId, IPTTreatmentPackages | Output: Update Status) | |
| **Trigger** – Should be invoked from the Inpatient service detail module | |
| **Steps and Actions**   1. Lists the In-patient services for it to be shown on the User interface  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Ailment** | **Package** | | | | | **Name** | **Test detail** | **Cost** | **Duration in weeks** | | Orthopaedics | Package 1 | OPT1, OPT2 | 2500 | 4 | | Orthopaedics | Package 2 | OPT3, OPT4 | 3000 | 6 | | Urology | Package 1 | UPT1, UPT2 | 4000 | 4 | | Urology | Package 2 | UPT3, UPT4 | 5000 | 6 |  1. Provides a list of specialists with their expertise and service charge detail. Have a pre-defined list of specialists, for each of the domain. Have pre-defined data of two junior and senior specialists for each of the domains. | |
| **Non-Functional Requirement:**   * Only Authorized requests can access these REST End Points | |

### **Functional Requirements – IP Treatment Microservice**

|  |  |
| --- | --- |
| International patients treatment management system | IP Treatment Microservice |
| **Functional Requirements**  This microservice should be invoked from the Web application. It allows the following operations:  The web application provides input to   * For a new in-patient service request, get the   + Patient detail – Name, Age   + Package name   + Date of treatment start * With this information, assign the specialist to the package with the logic that   + Package 1 - junior specialist of the domain   + Package 2 - senior specialist of the domain * A table with all these details should be returned to the Web application and the treatment status of the patient can be marked as in-progress | |
| **Entity**  **PatientDetail**   1. **Name** 2. **Age** 3. **Ailment** 4. **TreatmentPackageName** 5. **TreatmentCommencementDate**   **TreatmentPlan**   1. **PackageName** 2. **Test details** 3. **Cost** 4. **Specialist** 5. **TreatmentCommencementDate** 6. **TreatmentEndDate**   **REST End Points**  **InpatientService Microservice**   * 1. GET: /FormulateTreatmentTimetable (Input: PatientDetail | Output: TreatmentPlan) | |
| **Trigger** – Can be invoked from Patient treatment packages web portal | |
| **Steps and Actions**   * + This microservice will be invoked from the web portal with the JWT obtained from the Authentication service.   + The FormulateTimeTable is used to generate a time table for the admin user to explain to the patient.   + Patient detail along with the package and specialist detail should be saved in database. | |
| **Non-Functional Requirement:**  Only Authorized requests can access these REST End Points | |

### **Functional Requirements – InsuranceClaim** **Microservice**

|  |  |
| --- | --- |
| International patients treatment management system | InsuranceClaim Microservice |
| **Functional Requirements**  The intent of this Microservice is to provide a list of Insurers with their detail, for the admin user to choose based on the patient’s choice. Upon selection of the insurer, a claim initiation request is sent to the insurer with the balance amount to be paid by the patient to be displayed on the Web UI. | |
| **Entities**  **InsurerDetail**   1. **InsurerName** 2. **InsurerPackageName** 3. **InsuranceAmountLimit**   <Total amount for the package>   1. **DisbursementDuration**   <Insurance amount disbursement duration>  **InitiateClaim**   1. **PatientName** 2. **Ailment** 3. **TreatmentPackageName** 4. **InsurerName**   **REST End Points**  **AuditSeverity Microservice**   * + GET: /GetAllInsurerDetail (Input: None | Output: List of Insurer detail objects)   + GET: /GetInsurerByPackageName (Input: PackageName | Output: Insurer detail object)   + POST: /InitiateClaim (Input: InitiateClaim | Output: Balance amount to be paid) | |
| **Trigger** – Can be invoked from Web portal | |
| **Steps and Actions**   * + To mark a service as complete, Insurer detail should be chosen to make the payment   + This microservice should be invoked to get all the insurers with the details. The insurer details can be a set of pre-defined detail of Insurers.   + From the list, the admin chooses one of the insurer initiate claim.   + The microservice gets the chosen insurer, determines the balance amount to be paid, returns the balance amount to be paid to be shown on the UI. This completes the claim initiation. The chosen insurer, InsurerPackageName should be mapped to the patient detail in the database. | |
| **Non-Functional Requirement:**  Only Authorized requests can access these REST End Points | |

### **Functional Requirements – Authorization Microservice**

|  |  |
| --- | --- |
| International patients treatment 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 – IP treatment management portal**

|  |  |
| --- | --- |
| International patients treatment management system | IP treatment management Portal |
| **Client Portal Requirements**   * IP Treatment management Portal must allow the admin to Login. Once successfully logged in, the member do the following operations:   + View the list of service offerings as package of treatments for International patients.   + Option to choose the package for a patient and view a timetable of treatment plan.   + View the list of registered patients.   + Choose any of the patients to mark the treatment completion. On doing that, admin should choose the insurer and package for claim. To proceed with that, the InsuranceClaim microservice should be invoked to initiate the claim. Upon getting back the balance amount to be paid, the insurer mapping to patient should be done and all the details should be saved in the database.   + Update Treatment package   + Add a new Specialist   + Delete a specialist   + Get a List of Specialists by their specialization. * Each of the above operations will 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 MVC app running in a local environment.

# Design Considerations

Java and Dotnet specific design considerations are attached here. These design specifications, technology features have to be strictly adhered to.



# Reference learning

Please go through all of these k-point videos for

Microservices deployment into Azure Kubernetes Service.

|  |
| --- |
| [AzureWithCICD-1](https://cognizant.kpoint.com/app/video/gcc-19532393-d4e0-4fd9-8a0c-80ecbdb349d3) |
| [AzureWithCICD-2](https://cognizant.kpoint.com/app/video/gcc-6633a958-ab72-4c69-b926-fe832e4b56a1) |
| [AzureWithCICD-3](https://cognizant.kpoint.com/app/video/gcc-553eb186-c1cf-448e-96fc-a96fe37b2e6a) |
| [AzureWithCICD-4](https://cognizant.kpoint.com/app/video/gcc-fad7d4af-d651-4501-99c6-2785190670c2) |

**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 | Initial baseline created on <24-Jul-2020> by <Seshadri M R> | | | |
| V2,0.0 |  | | | |
| **Section No.** | **Changed By** | **Effective Date** | **Changes Effected** |
| 2.3  2.5  3.1.1  3.1.5 | Reni Varghese | 03-March-2021 | Enhancement specified to the IPTreatmentOffering Microserivce to provide the following additional functionalities.   * Enables the admin to update packages. * Enables the admin to add new specialist details * Enables the admin to delete a specific specialist. * Provides the list of specialists by their area of expertise.   The Management portal also enhanced to incorporate the above-mentioned changes. |