|  |  |
| --- | --- |
|  | **Cognizant Academy**  **Audit 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](#_Toc46477278)

[2.0 Introduction 4](#_Toc46477279)

[2.1 Purpose of this document 4](#_Toc46477280)

[2.2 Project Overview 4](#_Toc46477281)

[2.3 Scope 4](#_Toc46477282)

[2.4 Hardware and Software Requirement 5](#_Toc46477283)

[2.5 System Architecture Diagram 6](#_Toc46477284)

[3.0 System Requirements 6](#_Toc46477285)

[**3.1.1** **Functional Requirements – Audit checklist** **Microservice** 6](#_Toc46477286)

[**3.1.2** **Functional Requirements – Audit benchmark Microservice** 7](#_Toc46477287)

[**3.1.3** **Functional Requirements – AuditSeverity** **Microservice** 8](#_Toc46477288)

[**3.1.4** **Functional Requirements – Authorization Microservice** 9](#_Toc46477289)

[**3.1.5** **Functional Requirements – Audit management portal** 10](#_Toc46477290)

[4.0 Cloud Deployment requirements 10](#_Toc46477291)

[5.0 Design Considerations 10](#_Toc46477292)

[6.0 Reference learning 11](#_Toc46477293)

[7.0 Change Log 11](#_Toc46477294)

# 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 “Audit 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 Supply chain Management Organization wants to automate the Audit processing, to make the management scalable and ensure clarity and ease of tracking.

## Scope

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

|  |  |  |
| --- | --- | --- |
| **Req. No.** | **Req. Name** | **Req. Description** |
| REQ\_01 | Audit checklist module | Audit checklist Module is a Middleware Microservice that performs following operations:   * Provides a list of YES/NO type of questions for the audit based on the audit type * This will be consumed by the User interface the display the questions on the portal |
| REQ\_02 | Audit benchmark module | Audit benchmark Module is a Middleware Microservice that performs the following operations:   * Provides the acceptable number of answers with NO as the answer for various audit types |
| REQ\_03 | Audit severity module | Audit severity Module is a Middleware Microservice that performs the following operations:   * Gets the audit response and analyzes the project execution status   + Gets the Audit benchmark detail from Microservice, compares the current project data. Determines the project execution status and the duration in which remedial action should be taken. |
| REQ\_04 | Authorization service | This microservice is used with anonymous access to Generate JWT |
| REQ\_05 | Audit management portal | A Web Portal that allows a member to Login and allows to do following operations:   * Login * Choose audit type and view audit questions * Provide response and view the project execution status * Store the Audit date, Audit type, project execution status and remediation duration in database |

## 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



# System Requirements

### **Functional Requirements – Audit checklist** **Microservice**

|  |  |
| --- | --- |
| Audit Management System | AuditChecklist Microservice |
| **Functional Requirements**  The intent of this Microservice is to provide the list of questions for Audit checklist. Post Authorization using JWT, the questions will be used to display the questions on the Web UI | |
| **Entities**  **REST End Points**  **AuditChecklist Microservice**   * + GET: /AuditCheckListQuestions (Input: AuditType | Output: List of questions) | |
| **Trigger** – Should be invoked from Audit management Portal (local MVC app) | |
| **Steps and Actions**   1. Audit management Portal should be the front-end application where audit related detail will be provided to the project manager to check the execution status. An instance of the AuditRequest object should be created to fill the request detail. 2. The portal should invoke the Authentication Microservice to get the JWT. 3. On receiving the token, the web portal should invoke the AuditChecklist Microservice GET action method with the Audit type. JWT should be added to the request header for authorization. 4. The microservice should get the audit type and return the checklist questions    * Question list      + Internal        - Have all Change requests followed SDLC before PROD move?        - Have all Change requests been approved by the application owner?        - Are all artifacts like CR document, Unit test cases available?        - Is the SIT and UAT sign-off available?        - Is data deletion from the system done with application owner approval?      + SOX        - Have all Change requests followed SDLC before PROD move?        - Have all Change requests been approved by the application owner?        - For a major change, was there a database backup taken before and after PROD move?        - Has the application owner approval obtained while adding a user to the system?        - Is data deletion from the system done with application owner approval? 5. The Web application should use the list of questions to display and capture the response. | |
| **Non-Functional Requirement:**   * Only Authorized requests can access these REST End Points | |

### **Functional Requirements – Audit benchmark Microservice**

|  |  |
| --- | --- |
| Audit Management System | AuditBenchmark Microservice |
| **Functional Requirements**  AuditSeverity Microservice interacts with AuditBenchmark Microservice. AuditBenchmark Microservice allows the following operations:  This Microservice should provide the acceptable benchmark value for every audit type. It should return a Dictionary of values with AuditType and Acceptable benchmark value of number of questions whose answers can be NO  Audit type: Internal; Acceptable value of NO: 3  Audit type: SOX; Acceptable value of NO: 1 | |
| **Entity**  **AuditBenchmark**   1. **AuditType**   <Type of audit>   1. **BenchmarkNoAnswers**   <Acceptable Number of questions whose answer is NO>  **REST End Points**  **Claims Microservice**   * + GET: /AuditBenchmark (Input: None | Output: List of AuditBenchmark) | |
| **Trigger** – Can be invoked from AuditSeverity Microservice | |
| **Steps and Actions**   * + This microservice will have only 1 REST endpoint to return the benchmark value of audit types | |
| **Non-Functional Requirement:** | |

### **Functional Requirements – AuditSeverity** **Microservice**

|  |  |
| --- | --- |
| Audit Management System | AuditSeverity Microservice |
| **Functional Requirements**  AuditSeverity Microservice should be invoked from Audit management portal. Post authorization of request, it allows the following operations:   * Based on the Audit request input, the AuditBenchmark Microservice should be invoked to analyze the count of questions whose answer can be NO * Determine the project execution status and arrive at the remediation duration detail   + If the value is within the acceptable limit, then no action need to be action, else action should be taken in taken within a specific span of time. The logic is listed below     - Audit type – Internal; Count of NO <= acceptable value ; Audit result – GREEN; Remedial action duration: No action needed     - Audit type – Internal; Count of NO > acceptable value ; Audit result – RED; Remedial action duration: Action to be taken in 2 weeks     - Audit type – SOX; Count of NO <= acceptable value ; Audit result – GREEN; Remedial action duration: No action needed     - Audit type – SOX; Count of NO > acceptable value ; Audit result – RED; Remedial action duration: Action to be taken in 1 week | |
| **Entities**  **AuditRequest**   1. **ProjectName**   <Project on which audit is conducted>   1. **ProjectManagerName**   <Project manager name>   1. **ApplicationOwnerName**   <Application owner name>   1. **AuditDetail**    1. AuditType – Internal / SOX    2. AuditDate    3. AuditQuestions – List of questions   <Details of Audit>  **AuditResponse**   1. **AuditId**   <A random number generated to identify the Audit>   1. **ProjectExecutionStatus**   <The audit result on project execution>   1. **RemedialActionDuration**   <Duration by which the remedial action should be taken>  **REST End Points**  **AuditSeverity Microservice**   * + POST: /ProjectExecutionStatus (Input: AuditRequest | Output: AuditResponse) | |
| **Trigger** – Can be invoked from Audit management portal | |
| **Steps and Actions**   * + The portal should invoke the Authentication Microservice to get the JWT.   + The answers to the audit checklist questions along with the basic project information will be filled in the AuditRequest object. This will be sent as input to the AuditSeverity Microservice   + AuditSeverity microservice should interact with AuditBenchmark service   + The response from AuditSeverity Microservice along with the basic project information will be stored in the database thru the Web application | |
| **Non-Functional Requirement:**  Only Authorized requests can access these REST End Points | |

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

|  |  |
| --- | --- |
| Audit 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 – Audit management portal**

|  |  |
| --- | --- |
| Audit Management System | Audit management Portal |
| **Client Portal Requirements**   * Audit management Portal must allow a member to Login. Once successfully logged in, the member do the following operations:   + Choose the audit type to view the list of audit checklist questions   + Let the project manager provide answers to the questions   + Invoke the AuditSeverity Microservice to determine the project execution status   + Display the result on the Web UI * The audit request detail along with the project execution status and remedial action duration should be saved to the database * 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 .Net core 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> | | | |
|  |  | | | |
| **Section No.** | **Changed By** | **Effective Date** | **Changes Effected** |
|  |  |  |  |