Cognizant Academy

Audit Management System

Digital Honors 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

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. If backend has to be set up manually, appropriate DB scripts have to be provided along with the solution ZIP file.
6. Follow coding best practices while implementing the solution. Use appropriate design patterns wherever applicable.
7. You are supposed to use an In-memory database or code level + Cloud data as specified, for the Microservices that should be deployed in cloud.

2.0 Introduction

2.1 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 specified wherever applicable, however the detailed design considerations have to be strictly adhered to during implementation.

2.2 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.

2.3 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 severity module	Audit severity Module is a Middleware Microservice that performs the following operations: • Gets the audit response and analyzes the project execution status ○ Based on the standard set in this Microservice for number of acceptable findings, it compares the current project data, Determines the project execution status and the duration in which remedial action should be taken.
REQ_03	Authorization	This microservice is used with anonymous access to

	service	Generate JWT
REQ_04	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

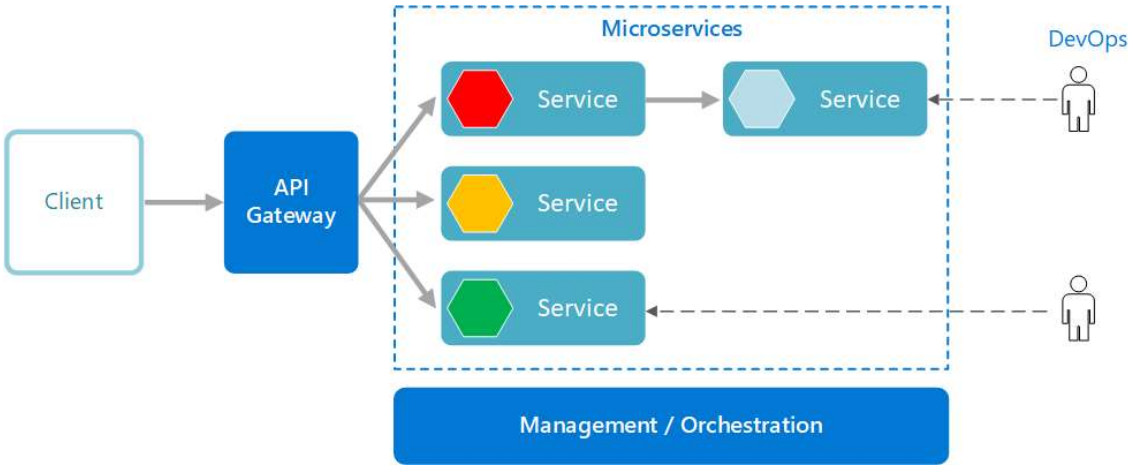
The requirement details given below states in-memory database or code level data usage. **On Cloud deployment, towards the end of the Cloud access and before the evaluation, this could be modified to use Cloud database.**

The front-end application to be done on Angular

2.4 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. Visual Studio Code latest version
    f. AWS Cloud access
3. Software Requirement (Dotnet)
    a. Visual studio 2017 enterprise edition
    b. SQL Server 2014
    c. Postman Client in Chrome
    d. Azure cloud access
    e. Visual Studio Code latest version

2.5 System Architecture Diagram



3.0 System Requirements

3.1.1 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 Angular 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?
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

3.1.2 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, follow the setting for the acceptable value of NO
  Audit type: Internal; Acceptable value of NO: 3

- 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.

Entities

AuditRequest

1. **ProjectName**
   <Project on which audit is conducted>
2. **ProjectManagerName**
   <Project manager name>
3. **ApplicationOwnerName**
   <Application owner name>
4. **AuditDetail**
   a. AuditType

b. AuditDate **c.** AuditQuestions – Question count with YES or NO answer **AuditResponse** 1. **AuditId** <A random number generated to identify the Audit> 2. **ProjectExecutionStatus** <The audit result on project execution> 3. **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 answer count 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.
Non-Functional Requirement: Only Authorized requests can access these REST End Points

3.1.3 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	

3.1.4 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.	

4.0 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 use SQL database 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 Angular app running in a local environment.

5.0 Design Considerations

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

DigitalHonors-Proje
ct-DesignConsidera

6.0 Reference learning

Please go through all of these k-point videos for

Microservices deployment into Azure Kubernetes Service.

AzureWithCICD-1

AzureWithCICD-2
AzureWithCICD-3
AzureWithCICD-4

Microservice deployment to AWS

AWS Learning Reference 1
AWS Learning Reference 2
AWS Learning Reference 3

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
Swagger (Optional)	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	https://stackoverflow.com/questions/10732644/best-practice-to-return-errors-

WebApi	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

7.0 Change Log

	Changes Made			
V1.0.0	Initial baseline created on <8-Sep-2021> by <Seshadri M R>			
	Section No.	**Changed By**	**Effective Date**	**Changes Effected**