Auditing System

Contents

[1.0 Problem statement 2](#_Toc43141625)

[2.0 Skills to develop the project 2](#_Toc43141626)

[3.0 Architecture Diagram for the Problem Statement 3](#_Toc43141627)

[4.0 User Stories 4](#_Toc43141628)

[5.0 Expected Deliverables 5](#_Toc43141629)

[6.0 Milestone and duration 5](#_Toc43141630)

[7.0 Implementation Notes 6](#_Toc43141631)

[8.0 Evaluation rubrics 7](#_Toc43141632)

# Problem statement

X is the top auditing firm, was facing issues in their existing system, auditors often need to interact with their clients for getting more details on audit or receiving some missing documents, but this workflow of the auditing processes was not structured. Also, lot of manual intervention is needed here hence auditing timelines were overshot.

Auditing firm would like develop an Audit Workflow System which provides solution to above mentioned problems. The application should be Cloud Native Architecture with Microservice. Both functional and non-functional requirements are captured in this document. It also serves as the input for the project scoping.

**About the System**

Client wants to develop an Audit Workflow System (AWS) application in order to manage the audit workflow. This will be used by Auditors for communicating with their clients, also send/receive information and documents. This system will help to reduce manual intervention in this process.

The following section will cover aspects related to Bank Management System.

1. New User Login/Registration
2. Create Audit Record/s
3. Create Request (with/without documents)
4. Send Request
5. Receive or Client service respond to request

**Scope of the System**

The scope of the system is explained through its modules as follows

* New user registration – Register new user - Auditor and Client user
* New Client portfolio setup – Used by auditors to register the details of clients the system. The system stores the necessary details of the client and. Each client identify with unique id.
* Create Audit Record/s - Will be used by auditors to create new audit entries for client. It can target to audit specific business portfolio.
* Create and Send Requests – Auditors requesting more information or missing documents from their clients. Minimum one client side POC assigned to each audit portfolio.
* Send Response – Client service responds with details/ attach documents.

# Skills to develop the project

Associate will implement skills from Backend and Azure Cloud platform to develop the application.

Below are the skill details. Choose the Backend based on the technology stack.

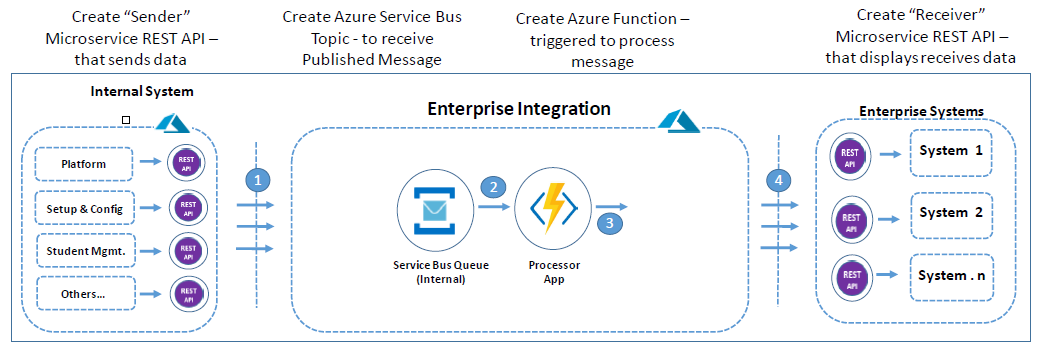
|  |  |
| --- | --- |
| **Tower Name** | **Topics** |
| Backend - .NET | Data Structures  .Net Core, C#  EF core  xUnit  ASP.NET Web API Core, JSON  Microservices |
| Cloud - Azure | Server less architecture (e.g. Azure Functions)  Azure App Services,  Azure Service Bus,  Azure DevOps, Azure SQLDB |

# Architecture Diagram for the Problem Statement

**Use Case Diagram**

**Flow Diagram** 

**Sample Architecture Diagram(TBU)**



# User Stories

Wireframes are attached and can be customized.

|  |  |  |
| --- | --- | --- |
| **User Story #** | **User Story Name** | **User Story** |
| US\_01 | Auditor Login | As an Auditor or client, I should be able to register my details in the system so that I can login into the system.(Two separate portals for auditor and client)  **Acceptance criteria:**  Both auditor and client should be able register the details in the system and it should be saved in the database.  Customer should be able to login with a User Id and Password that exists in database. On clicking logout the session should be invalidated and login page must be displayed    Capture the details like Name, Username, Password, Address, Email Address  \*\*\*\*\*DO NOT STORE PASSWORD IN PLAIN TEXT\*\*\*\* |
| US\_02 | Create Audit portfolio | As a Auditor user, I should be able to create audit portfolio for new or existing client  **Acceptance criteria**  Auditor should able to create audit portfolio for client with details like Audit Name, Description, Client Code, Audit Report Release date  Auditor can select existing client or create new client if required (You can have predefined clients as well)  Each portfolio or audit record should have Client Code  Multiple Portfolio records can exist for client., each identified by Audit Id  All details should get saved in database |
| US\_03 | View/Send Request | As an Auditor user, I should be able to create requests against audit portfolio.  **Acceptance criteria**  Auditors can create request with title, description, comments  Auditors can set assignees for request (Minimum one Client user needed), can assigned to multiple users  Save Draft – Save request data into database  Submit Request – Submit request to receive for client user.  Error Handling – Display appropriate error message to user  Drafted/Sent/Processing/Received/Sent Response/Complete |
| US\_04 | Add Documents to request | As an Auditor user, I should be able to add document with requests against audit portfolio.  **Acceptance criteria**  User should able to upload one more documents  Uploaded documents should get displayed and user able to view it  Send request should send request with documents.  Error Handling  **Use Blob Storage to store documents** |
| US\_05 | View/Send Responses | As a Customer, I should be able to login so that I can view request and send responses  **Acceptance criteria:**  Open existing request to add response details , comments and save  Each response identified by unique id  View/Download request documents  Attach Response Documents (**Optional Requirement**)  Save Draft – Save request data into database  Submit Request – Submit request to receive for auditor.  Error Handling – Display appropriate error message to user |

# Expected Deliverables

The following deliverables are expected as outcomes

* Application Code base
* API for each service
* Readme document on the complete application
  + Setup of the application
  + High level steps used to convert to server less architecture
  + How to run the application
  + Any inference
  + Snapshot of any implementation
* Reports:
  + Code Assessment Report
  + Functional Test Report
  + Vulnerability Assessment Report
  + Performance Test Report
  + Code Profiling Report

# Milestone and duration

As per project requirement, modification can be done in the below table.

|  |  |
| --- | --- |
| Milestone | Topic |
| Milestone -1 | Design and develop microservice application using backend technology |
| Milestone -2 | Convert the existing application to server less architecture, API Gateways and implement DevOps on Cloud |

# Implementation Notes

As per the project requirement modification can be done in the below table.

|  |  |
| --- | --- |
| Backend – .NET | Milestone-1   * Use Rest APIs to develop the services * Use Microservice Architecture * Use Domain Driven Design * Implement repository pattern * Azure SQL DB to be used to store data * Use Swagger definitions * Use browser / POST Man to invoke APIs * Implement Redis-Cache * Upload file to Azure Blob * User access security microservice to allow/disallow CRUD operations * Any error message or exception should be logged (and help in refactor) * Unit test the application * API versioning is done to manage changes in the API without affecting the client that are using the existing API * Raising Pull Requests, closing them are highly encouraged * All implementation should publish Code Quality Metrics using SonarCloud/SonarQube * Technical Debt – lower-the-better * Code Smell – lower-the-better * Cyclomatic Complexity - lower-the-better * Code Coverage – higher-the-better * Secure coding practices * Follow coding standards * Message input/output format should be in JSON (Read the values from the property/input files, wherever applicable). Input/output format can be designed as per the discretion of the participant * Web service URLs should be configurable. * Implement External Configurable Solution |
| Cloud - Azure | Milestone-2   * Application should follow Serverless architecture on public cloud. Maximize use of cloud provider’s PaaS capabilities to minimize IT infrastructure management effort. * Usage of API Gateway as the entry point for service calls originating from clients (web or mobile). The API Gateway should also cater to authentication and handle SSL termination. * Use cloud provider’s DevOps tools to setup a CI/CD pipeline to automate your releases. API versioning best practices should be followed. * Publish of web app to azure app service * Implement functionality of FileUpload service which uploads a file from local to Azure Blob storage * Usage of Redis cache in the get operations to improve the performance of the App * Usage of logger to log critical actions in the app * Implement simple azure functions that gets triggered on message arrival on service bus or event trigger through grid |

# Evaluation rubrics

As per the project requirement any addition can be done in the below table.

|  |  |
| --- | --- |
| Microservices | * Follow the below basic structure   + API - Controllers   + Domain - Model, Events, Business Services Integration   + Services – API Implementation   + Infrastructure Project * Associate must have designed/developed Microservices as per the requirement * Each of the Microservices need to comprise below functionality, which need to be developed * Entity & Model classes, including appropriate relationship (like One-One, Many-One, etc) between Entity Classes. (Entity and Model classes have been developed in the Previous Phase) * In case specific Entity or Model classes are required across multiple Microservices, it is recommended to maintain separate copy of Entity or Model classes for each Microservices. * Microservices should interact with corresponding databases it owns. * Microservice need to interact with other Microservice * Usage of Postman to test the Microservices by directly passing requests to each REST end Point, of each Microservice * Circuit Breaker, Service Registry, Service Discovery should be implemented |
| Rest API | * Associate must have used REST API for exposing resources * Associate must have used HTTP GET/PUT/POST request method designators for the business methods which is to be exposed * Associate must have customized the request and response formats according to the requirement * Associates must have used appropriate RETURN CODES based on the service outcome * Associates must have extracted query/form/header parameters from the input * Associate must have built a custom response based on the input * Use Swagger UI and test each public method in the service |
| C# | Associate should have used appropriate Base class Libraries, Control Statements and Operators, File Handling and I/O Operations for implementing the functionalities. |
| Unit Testing | * Test cases covers the functionality of API with custom inputs * Good test Coverage |
| Common | * Code Smell * Technical Debt * Secured Coding * Coding Standards |
| Server less implementation | * Configuration of Azure service bus (via Topic or Queue) * Azure functions that gets triggered on message arrival on service bus or event trigger through grid. * Azure SQL DB used to store data * Implement functionality of FileUpload service which uploads a file from local to Azure Blob storage * Publish of web app to azure app service * Usage of Redis cache in the get operations to improve the performance of the App * Usage of logger to log critical actions in the app * Implement simple azure functions that gets triggered on message arrival on service bus or event trigger through grid |
| DevOps on Cloud | * DevOps pipeline for each microservices which uses cloud PaaS services to trigger a CI/CD pipeline when code is checked-in to GIT * The check-in process should trigger unit tests with mocked dependencies * Unit tests should not alter persistent data * DevOps dashboard should show status of CI/CD pipeline * DevOps pipeline should support manual approval for rollout, gradual traffic shifting and rollback to earlier version * Checked-in code should meet 75%+ code coverage in unit testing |