Pension Plan Quote Management

**Contents**

**1. Problem statement 2**

**2. Skills to develop the project 3**

**3. Architecture Diagram for the Problem Statement 3**

**4. User Stories 3**

**5. Expected Deliverables 4**

**6. Milestone and duration 5**

**7. Implementation Notes 6**

**8. Evaluation rubrics 7**

1. **Problem statement**

The objective of the document is to provide associate with a practice case study to develop .Net Core based .NET application, implement the Azure DevOps with CI/CD pipeline using Azure Cloud platform, and deploy using Docker container.

A Pension insurance company wants to manage the quote information. If investment amount, retirement age and pensions plan is provided then it will give the information on money received on maturity date. The insurance company wants manage this information in their data store to follow-up the client further for actual investment. The insurance company wants to provide this service as a REST API and host in Azure Platform.

**Information gathered while creating a quote.**

1. Client Name (30 characters)
2. Client Sex (Male/Female).
3. Date Of Birth (Past Date)
4. Quote Date (System Date)
5. Email (email format)
6. Mobile Number(10 digit numbers)
7. Investment amount (decimal only)
8. Retirement Age ( 60 >= && <=75)
9. Pension Plan (Pension Silver / Pension Gold / Pension Platinum)
10. Maturity Amount (Calculated)

**Maturity Amount Calculation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Plan** | **Min. Investment Amount** | **Min. Retirement Age** | **Factor** |
| Pension Silver | 100000 | 65 | 0.02 |
| Pension Gold | 300000 | 63 | 0.04 |
| Pension Platinum | 500000 | 60 | 0.06 |

Maturity Amount = Investment Amount \* (1 + Factor) \* (Retirement Age – Current Age) / 100

The following section will cover aspects related to Pension Plan Management System.

1. Retrieve Quote list
2. Create Quote
3. Edit Quote
4. Delete Quote

Scope of the System

The scope of the system is explained through service provided by the REST API as follows

* Retrieve Quote List – used to retrieve list of existing quote in the Pension Plan System.
* Create Quote – will allow user to create quote in the system with given client information.
* Edit Quote - will allow user to edit the already existing Quote in the system with new plan or investment amount or retirement age changes.
* Delete Quote – will allow user to delete the quote, which already exist in the system.

1. **Skills to develop the project**

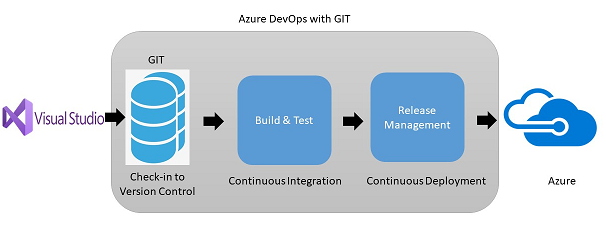
Associate will implement skills from .NET Core and Azure DevOps Cloud platform to develop the REST API services.

Below are the skill details.

|  |  |
| --- | --- |
| **Tower Name** | **Topics** |
| Backed - .NET | .Net Core, C#, NUnit  EF core  ASP.NET Web API Core, JSON |
| Cloud – Azure | Docker, Containerization, Azure Container services, Azure Pipelines, Azure Devops |

1. **Architecture Diagram for the Problem Statement**

Development Environment Architecture



1. **User Stories**

Capture the user stories of the application.

|  |  |  |
| --- | --- | --- |
| **User Story #** | **User Story Name** | **User Story** |
| US\_01 | Retrieve Quote List | As a client, I want to get the list of Quote available in the system.    Acceptance criteria:  When there is an existing Quote then return them all.    Return all the fields mentioned above including QuoteId, QuoteDate |
| US\_02 | Create Quote | As a Client, I want to create new Quote in the system.  Acceptance criteria:  Validation has to be done on following fields as per criteria given above.  Client Name  Client Sex  Date Of Birth  Email  Mobile Number  Investment amount  Retirement Age  Pension Plan  IF any the validation is field, it has to return list of errors to client.  Quote Date should always be taken as system date.  Maturity Amount has to be calculated according to the above formula |
| US\_03 | Edit Quote | As a Client I should be able to edit the quote, which is already in the system.  Acceptance criteria: Quote Id has to be validated if it is not found in the system then Quote {id} is not exists message should be shown.  Quote Date should be revised to current system date.  Maturity Amount has to be recalculated. |
| US\_04 | Delete Quote | As a Client I should be able to delete a quote which is already in the system.  Acceptance criteria:  Quote Id has to be validated if it is not found in the system then Quote {id} is not exists message should be shown.    The record to be permanently deleted from the system. |
|  |  |  |
|  |  |  |

1. **Expected Deliverables**

Capture the deliverables to be submitted by associate after completing the development.

* Asp.net mvc core web api application
* Source code development in C#
* Hosted in GIT
* Unit Tested
* Azure DevOps:
* Continuous Integration Pipeline
* Release management pipeline
* Hosted the app using Docker Container
* Sql Database in Azure Cloud
* Docker
* Docker file needed to create Docker image.
* Scripts
* Database schema script
* Code Quality metrics
* SonarQube Report

1. **Milestone and duration**

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

|  |  |
| --- | --- |
| Milestone | Topic |
| Milestone -1 | Developing the API with Test Driven Development using NUnit Framework, with Coding Standings followed in the Account. Database script for schema creation. EF Core will be used for persistence.  Application should be runnable in the local machine with local database. Postman tool will be used to demonstrate the consumption of the service. |
| Milestone -2 | Implement Build and Release pipeline using Azure DevOps |
| Milestone-3 | Deployment using Docker container. |

1. **Implementation Notes**

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

|  |  |
| --- | --- |
| Backend -.NET | * Use Rest APIs to develop the services * Use browser / POST Man to invoke APIs * Use EF Core * 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 * Database connections and web service URLs should be Configurable. * Implement Unit Test Project for testing the API * Follow coding standards * Code Quality need to be measured by SonarQube |
| Cloud – Azure DevOps | Milestone-2   * Creating an Azure DevOps Organization * Creating an Azure DevOps Project * Building an Azure DevOps Build Pipeline * Linking a GIT to the Build Pipeline * Manually Running the Azure Build Pipeline * Building an Azure DevOps Release Pipeline * Adding Artifacts to the Azure DevOps Release Pipeline * Creating the Azure DevOps Release |
| Docker | Milestone – 3   * Creating a Docker Image * Manually Deploying a Release * Inspecting the Deployed Azure Web API |

1. **Evaluation rubrics**

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

|  |  |
| --- | --- |
| Rest API | * Associate must have used REST API for exposing resources * Associate must have used HTTP GET/PUT/POST/DELETE request method designators for the business methods which is to be exposed * Associates must have used appropriate RETURN CODES based on the service outcome |
| .NET Core/C# | * Associate should have used appropriate Base class Libraries, Control Statements and Operators, File Handling and I/O Operations for implementing the functionalities.. * Should use Dependency Injection profiled by .NET Core |
| Unit Testing | * Test cases covers the functionality of API with custom inputs * Moq the database persistence. * Good test Coverage |
| Common | * Coding Standards * Code Quality Metrics will be measured by SonarQube |
| DevOps on Cloud | * DevOps pipeline for API 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 * Checked-in code should meet 75%+ code coverage in unit testing |