DEVOPS ASSESSMENT

Name:Priyadharshini Murugan

EmpId:11985

CICD –Problem Statement #7

A leading training institute in India is planning to develop their Self Learning Portal. The proposed solution has been planned to develop in a DevOps environment. There are many developers working on the application and they integrate code into a shared repository frequently .Each integration can then be verified by an automated build and automated tests and also deploying all code changes to a production environment after the build stage.

Module : Create Quiz Module

Scope:

You have been assigned the task of creating a .Net Core Application with a flow of creating new Quiz and making a CI / CD pipeline using tools such as Jenkins, Git, MSBuild ,SonarQube,NUnit and Docker.

Task # 1 : Create a a .Net Core Application with a REST endpoint to create a quiz to the system. Quiz information includes quiz id , quiz title , category , no of questions , max marks , total time.

Controller code :

Onlinequiztest Onlinequiztest.Controllers.QuizController

```
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using Onlinequiztest.Models;
using System.Reflection;

namespace Onlinequiztest.Controllers
{

    [ApiController]
    [Route("[controller]")]
    2 references | 0 changes | 0 authors, 0 changes
    public class QuizController : Controller
    {
        private readonly string _jsonFilePath = "data.json";

        [HttpGet]
        2 references | 0 changes | 0 authors, 0 changes
        public async Task<List<Quiz>> Get()
        {
            using (StreamReader file = System.IO.File.OpenText(_jsonFilePath))
            {
                JsonSerializer serializer = new JsonSerializer();
                return (List<Quiz>)serializer.Deserialize(file, typeof(List<Quiz>));
            }
        }

        [HttpPost]
        1 reference | 0 changes | 0 authors, 0 changes
        public async Task<IActionResult> Post([FromBody] Quiz model)
        {
            var data = await Get() ?? new List<Quiz>();
            data.Add(model); // Add new data

            using (StreamWriter file = System.IO.File.CreateText(_jsonFilePath))
            {
                JsonSerializer serializer = new JsonSerializer();
                serializer.Serialize(file, data);
            }
            return Ok();
        }
    }
}
```

Model code:

```
Onlinequiztest                                  Onlinec
namespace Onlinequiztest.Models
{
    7 references | 0 changes | 0 authors, 0 changes
    public class Quiz
    {
        1 reference | 0 changes | 0 authors, 0 changes
        public int Id { get; set; }
        1 reference | 0 changes | 0 authors, 0 changes
        public string? Title{ get; set; }
        1 reference | 0 changes | 0 authors, 0 changes
        public string? Category { get; set; }

        1 reference | 0 changes | 0 authors, 0 changes
        public int Numberofquestions { get; set; }

        1 reference | 0 changes | 0 authors, 0 changes
        public int Maximummark { get; set; }

        1 reference | 0 changes | 0 authors, 0 changes
        public int Totalmarks { get; set; }
    }
}
```

Program.cs code:

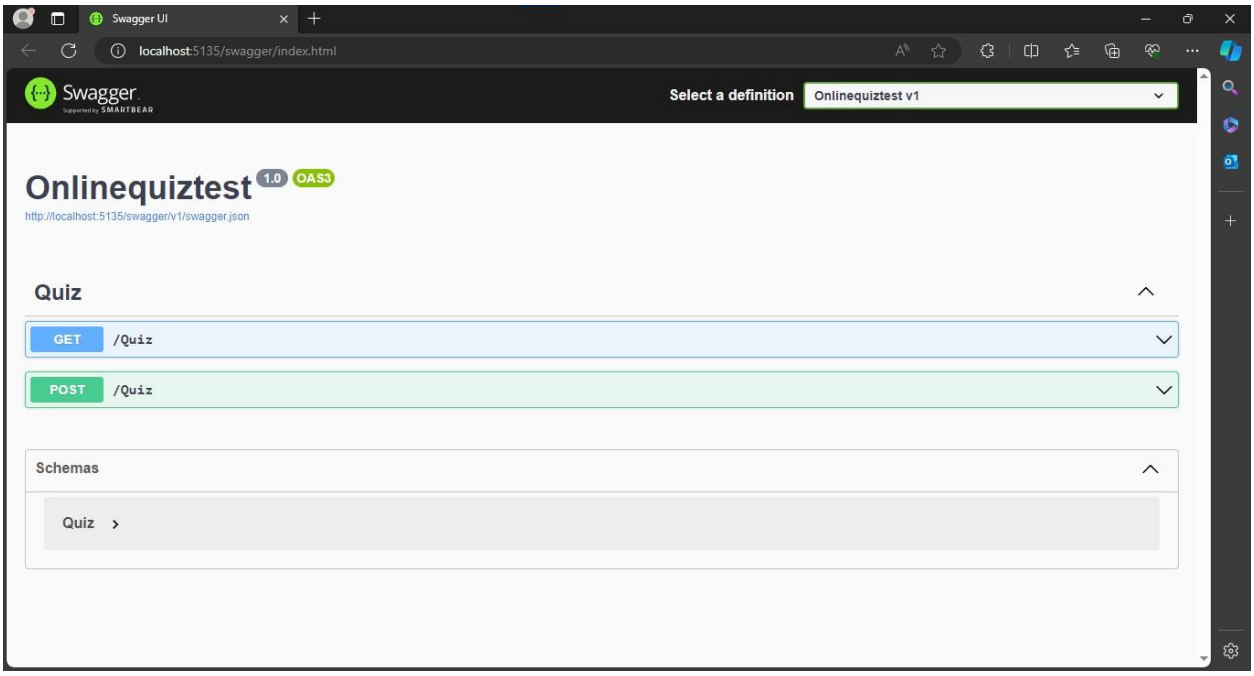
```
Onlinequiztest
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseAuthorization();

app.MapControllers();
app.UseHttpsRedirection();

app.Run();
```

Task #2: Create a NUnit TestClass and TestMethod to verify the DAO class method.

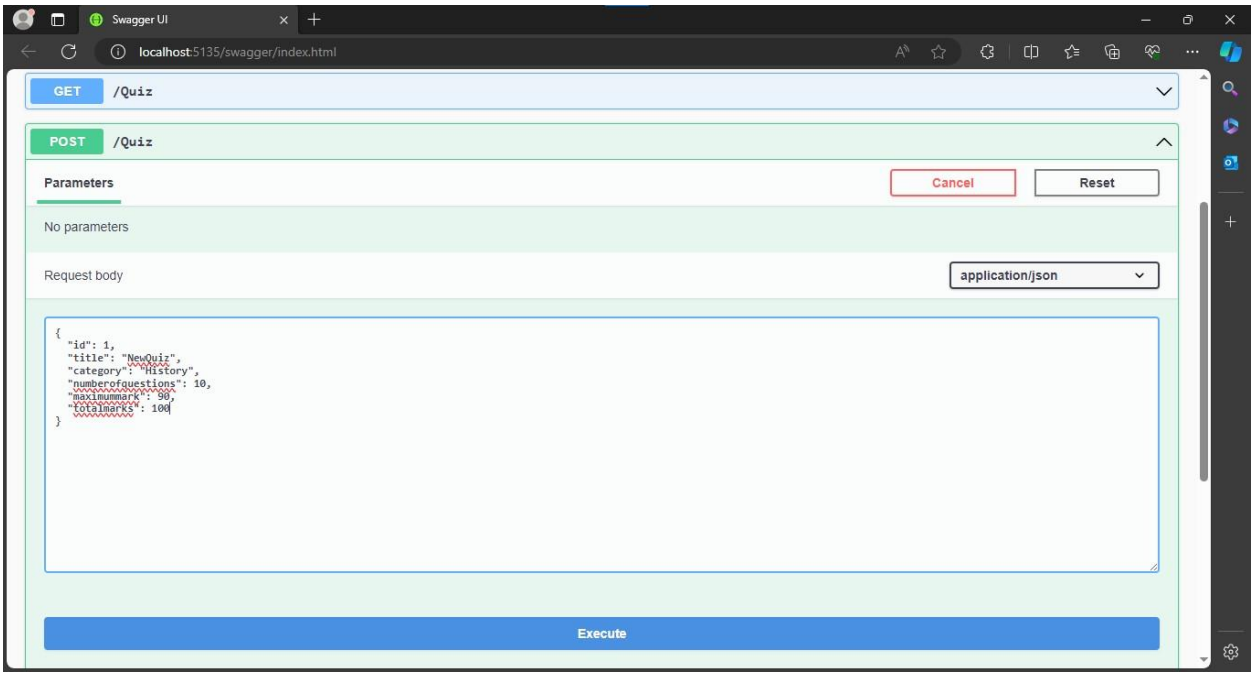
NUnit test code:

```
Quiztesting                                   Onlinequiztest.Tests.Tests
1   using NUnit.Framework;
2   using Onlinequiztest.Controllers;
3   using Onlinequiztest.Models;
4   using Microsoft.AspNetCore.Mvc;
5   using System.Collections.Generic;
6   using System.Threading.Tasks;
7   using System.Reflection;
8
9   namespace Onlinequiztest.Tests
10  {
11      [TestFixture]
        0 references | 0 changes | 0 authors, 0 changes
12      public class Tests
13      {
14              private QuizController _controller;
15
16              [SetUp]
                0 references | 0 changes | 0 authors, 0 changes
17              public void Setup()
18
19              {
20                  _controller = new QuizController();
21              }
22              [Test]
23
                0 references | 0 changes | 0 authors, 0 changes
24              public async Task Get_ReturnsListOfModels()
25
26              {
27                  // Act
28
29                  var result = await _controller.Get();
30                  // Assert
31
32                  Assert.IsNotNull(result);
33
34                  Assert.IsInstanceOf<List<Quiz>>(result);
35
36              }
37
38
39
```

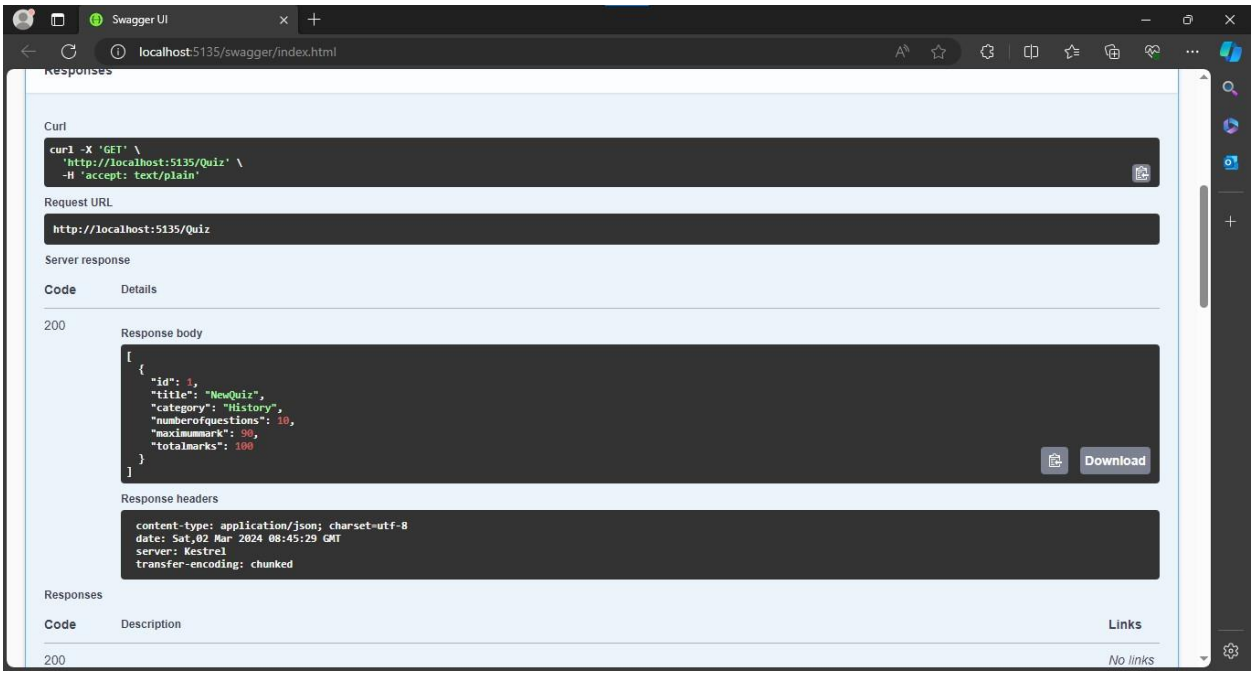
Task #3: Use SWAGGER and test the application



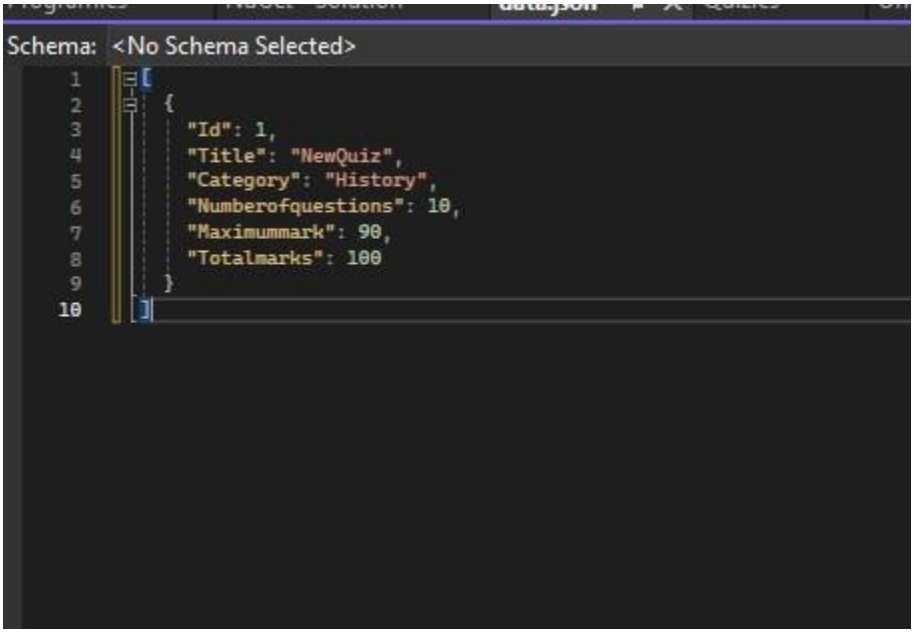
Inserting value through post method :



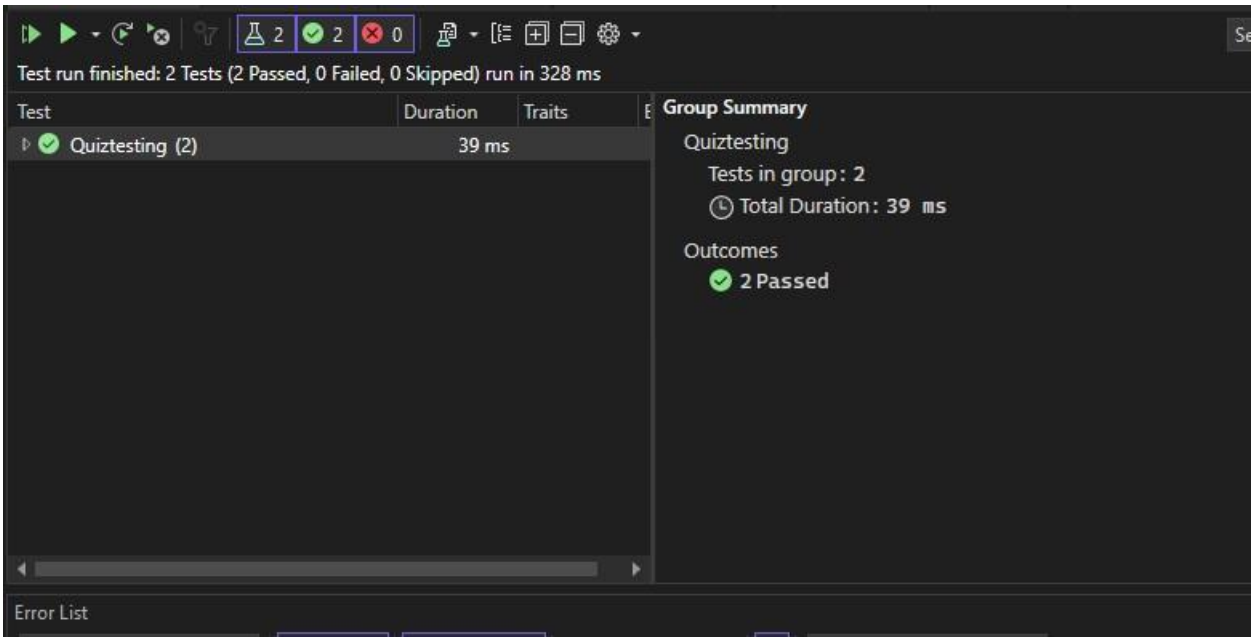
Using Get method to view :



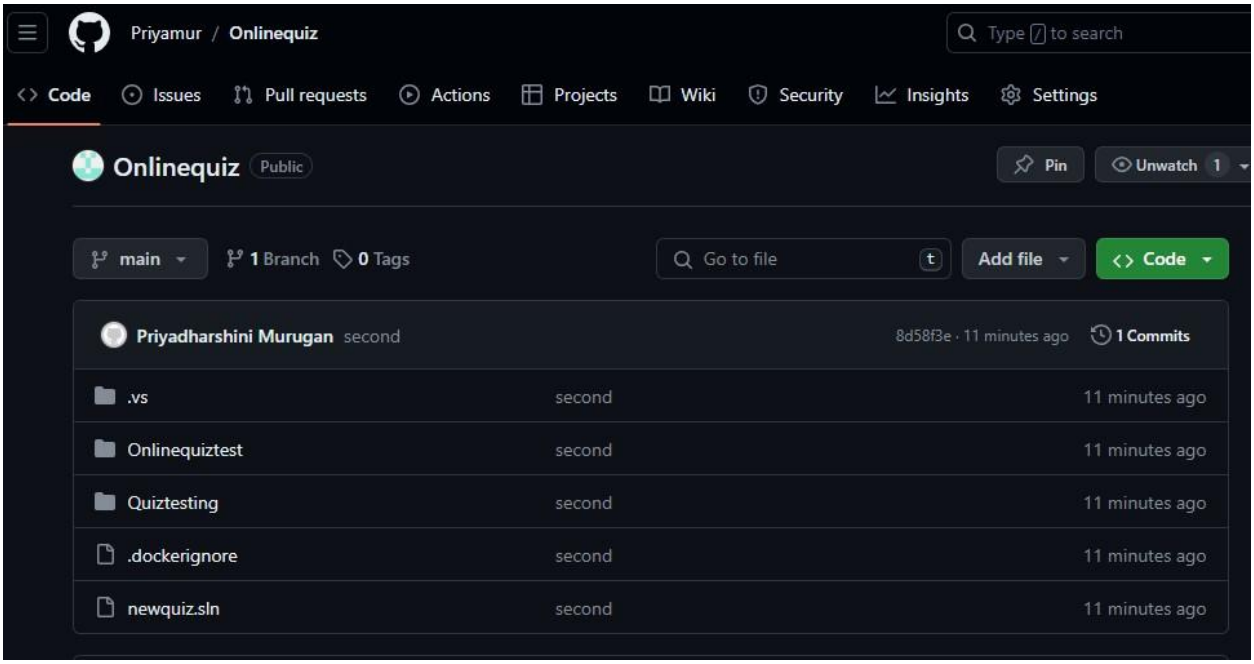
Output in Json file:



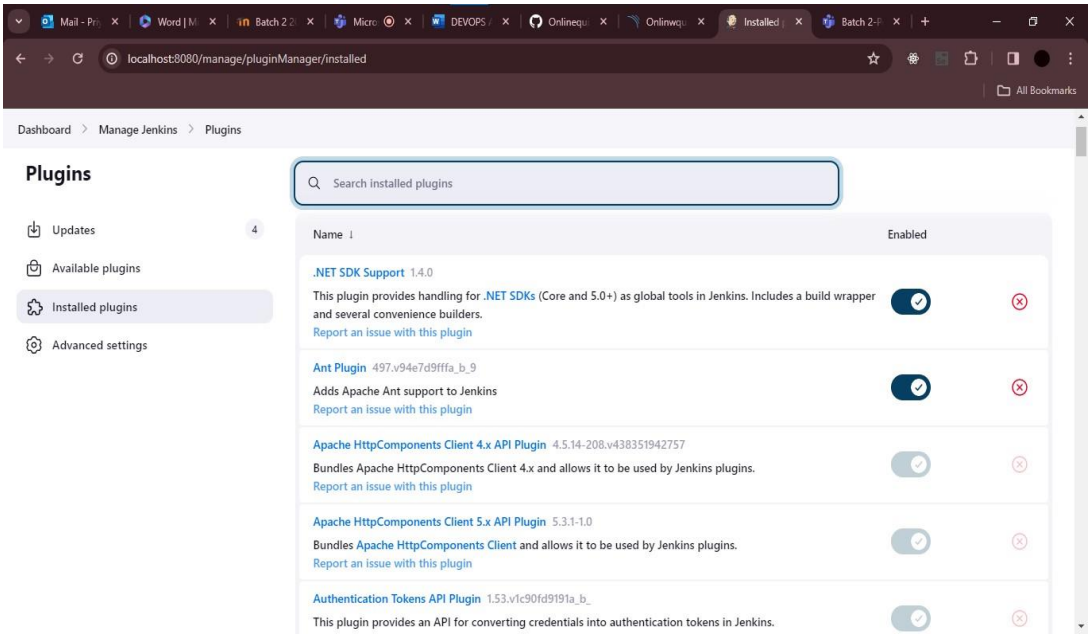
Nunit test case output:



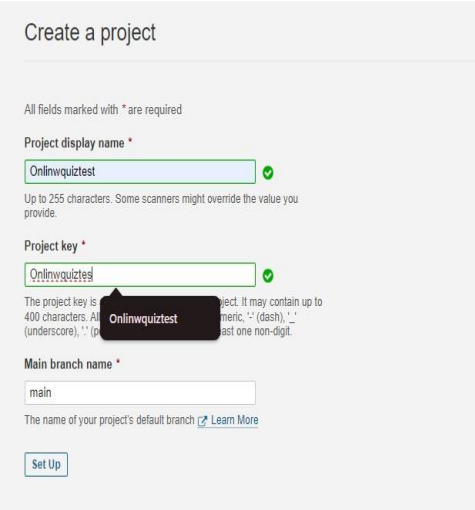
Task #4: Add the application to the Git Repository (Use the necessary commands)

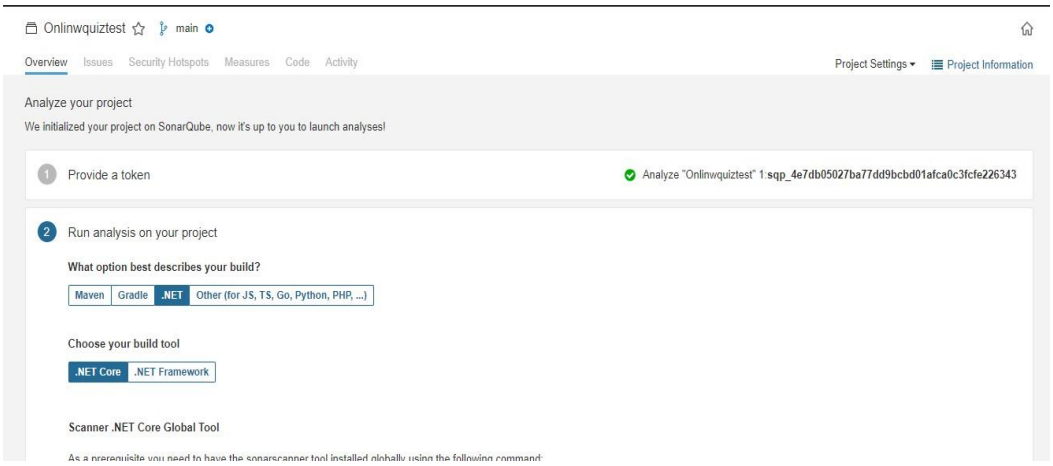


Task #5 : Configure the Jenkins tool with required plugins and paths.



Task #6 : Start the Sonar server and configure the project





Task #8 : Create a Dockerfile in your application and add necessary steps and commit the changes.

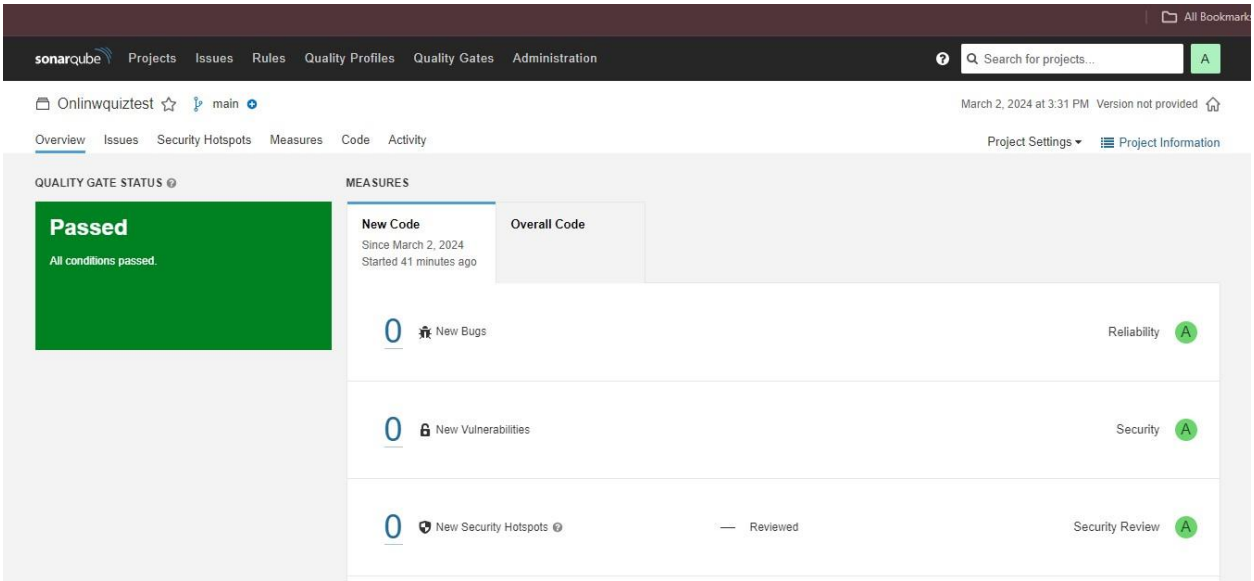
```
FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base
USER app
WORKDIR /app
EXPOSE 8080
EXPOSE 8081

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
ARG BUILD_CONFIGURATION=Release
WORKDIR /src
COPY ["Onlinequiztest.csproj", "."]
RUN dotnet restore "./././Onlinequiztest.csproj"
COPY . .
WORKDIR "/src/."
RUN dotnet build "./Onlinequiztest.csproj" -c $BUILD_CONFIGURATION -o /app/build

FROM build AS publish
ARG BUILD_CONFIGURATION=Release
RUN dotnet publish "./Onlinequiztest.csproj" -c $BUILD_CONFIGURATION -o /app/publish /p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "Onlinequiztest.dll"]
```

Task #9 : Create a Jenkins pipeline job and the pipeline script to get the application from Git, build the application ,run unit tests ,run code quality tests and deploy the application in docker.

Script ?

```
node
{
    stage('1. GIT Checkout'){
        git 'https://github.com/Priyamur/JenkinsAssessment.git'
    }
    stage('2. Build the project'){

       bat 'dotnet build'
    }
      stage('3. Test the project'){

       bat 'dotnet test'
    }
     stage('4. Code quality test'){

       bat 'dotnet sonarscanner begin /k:"Onlinwquiztest" /d:sonar.host.url="http://localhost:9000"  /d:sonar.login="sq
    }
     stage('5. Code quality test'){

       bat 'dotnet build'
    }
      stage('6. publish project'){

      bat 'dotnet sonarscanner end /d:sonar.login="sqp_dd86e4d712900982d6040efefdadf7d49dbdf391"'
    }
    dir('Jason4ni')
```

Save Apply



Output of Jenkins :

Dashboard > JsonPipeline >

- Status
- Changes
- Build Now
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

Build History trend

Filter builds... /

JsonPipeline

Add description

Disable Project

Stage View

	1. GIT Checkout	2. Build the Project	3. Test the Project	4. Code Quality Test	5. Code Quality Build	7. Dockerisation
Average stage times: (Average full run time: ~42s)	3s	4s	6s	2s	12s	12s
#11 Mar 02 15:46 No Changes	3s	4s	6s	2s	12s	12s

Permalinks