**Name: Priyanshu Sarraf**

**Roll: 2005183  
E-mail: priyanshusarraf23@gmail.com**

**Setting up cucumber and writing feature file for testing:**

We utilise a Gherkin file to define a testable application feature. The file begins with the Feature keyword, followed by the feature name on the same line and an optional description that may span numerous lines below.

**Step 1: Cucumber-JVM Implementation:**

In order to make use of Cucumber-JVM in a Maven project, the following dependency needs to be included in the POM:

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>6.8.0</version>

<scope>test</scope>

</dependency>

To facilitate JUnit testing with Cucumber, we need to have one more dependency:

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>6.8.0</version>

</dependency>

**Step 2: Create a Feature File**

I’ve created a new directory src/test/resources and created a new file greeting.feature which has the features in gherkin:

Feature: Greeting API

Scenario: Test greeting with default name

Given the greeting endpoint is available

When I request the greeting with name "World"

Then the response should contain "Hello, World!"

Scenario: Get a tailored greeting

Given the greeting endpoint is available

When I request the greeting with name "Spring Community"

Then the response should contain "Hello, Spring Community!"

**Step 3: Write Step Definitions**

Created a step definition class (e.g., GreetingStepDefinitions.java) under the com.example.restservice.steps package:

@RunWith(Cucumber.class)

public class GreetingStepDefinitions {

@Autowired

private GreetingController greetingController;

private String response;

@Given("the greeting endpoint is available")

public void theGreetingEndpointIsAvailable() {

}

@When("I request the greeting with name {string}")

public void iRequestTheGreetingWithName(String name) {

Greeting greeting = greetingController.greeting(name);

response = greeting.getContent();

}

@Then("the response should contain {string}")

public void theResponseShouldContain(String expected) {

assert(response.contains(expected));

}

}

**Step 4: Run Cucumber Tests**

Created a Cucumber runner class (e.g., GreetingCucumberRunner.java):

package com.example.restservice;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

import org.junit.runner.RunWith;

@RunWith(Cucumber.class)

@CucumberOptions(

features = "src/test/resources/features",

glue = "com.example.restservice.steps"

)

public class GreetingCucumberRunner {

}

**Step 5: Review Test Reports**

The generated test reports are stored in target folder:

-------------------------------------------------------------------------------

Test set: com.example.restservice.GreetingControllerTests

-------------------------------------------------------------------------------

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.663 s -- in com.example.restservice.GreetingControllerTests

The generated report shows that there are no errors in the file.

**Areas for improvement:**

* Implement global exception handling to provide consistent error responses.
* If your application serves requests from different origins, configure Cross-Origin Resource Sharing (CORS) appropriately.
* Dockerize our Spring Boot application for easy deployment and scalability.
* Set up CI/CD pipelines for automated testing and deployment.
* Consider implementing API versioning to manage changes in a backward-compatible manner.