Name :Suman Kumar  
Roll no : 20051666

Email: [sk3505357@gmail.com](mailto:sk3505357@gmail.com)

Github : <https://github.com/GAUTAMJBITS>

Problem Statement 02:

**End-to-End Testing of a Spring Boot Java Project with Cucumber**

Pre-requisite :

- Understand the Spring Boot Java project provided at the given GitHub link.

- Set up Cucumber in the Spring Boot environment.

- Write test scenarios and step definitions using Cucumber.

- Execute tests and generate a detailed test report

Technologies used :

● GitHub

● Spring Boot REST Service

● Java Development Kit (JDK)

● Spring Boot

● Cucumber

● Any Integrated Development Environment (IDE) like IntelliJ IDEA or Eclipse or VS CODE

● Maven or Gradle (depending on the project setup)

Let us understand about Technologies:

Step-by-Step Guide:

Step 1: Understand the Project:

1. Clone the Spring Boot Project:

⏎git clone <https://github.com/your-username/spring-boot-rest-service.git>

// username should be like this in my case:

//⏎git clone <https://github.com/your-username/spring-boot-rest-service.git>

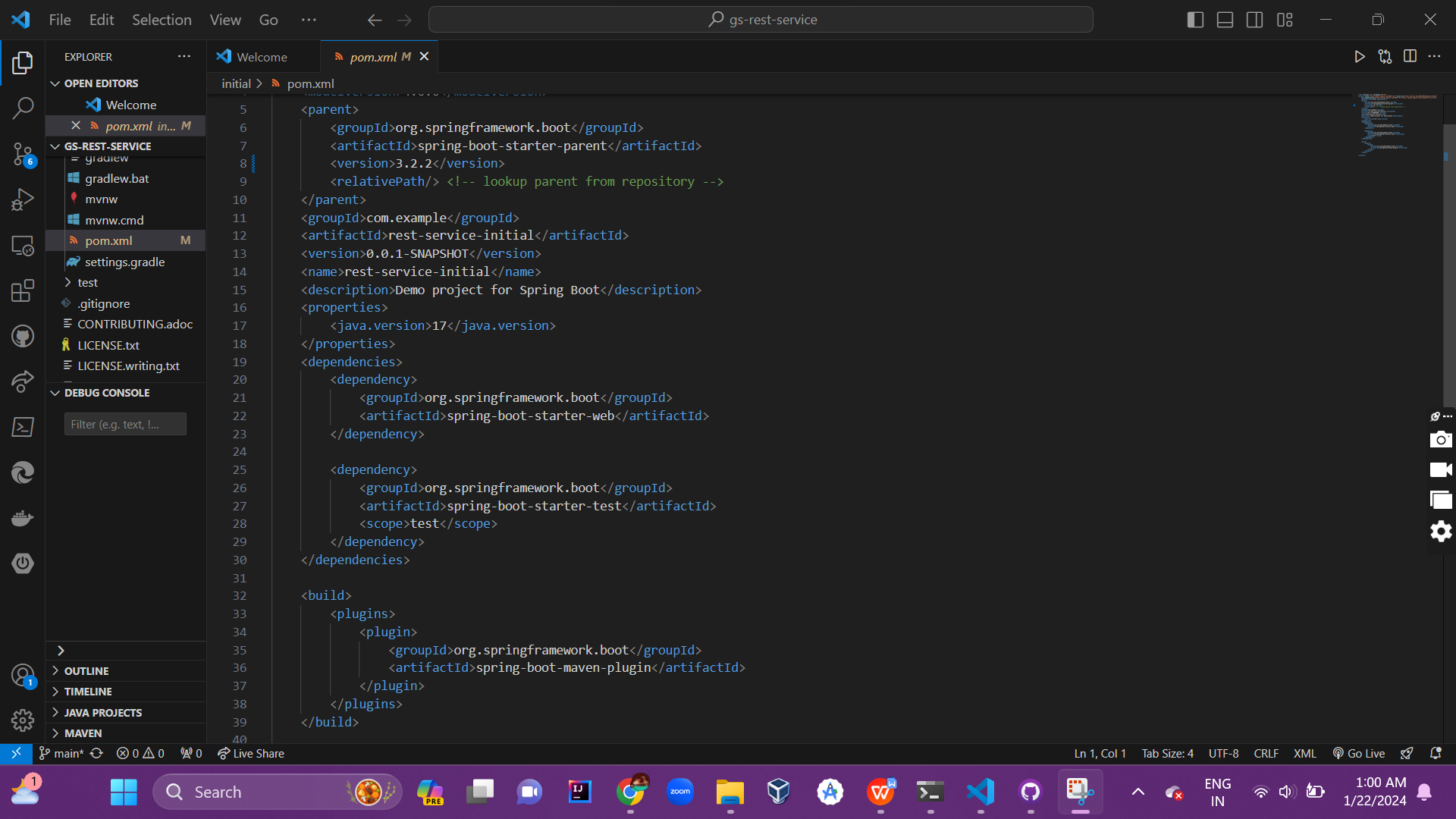
1. Explore the Project:

Navigate through the project structure. Understand key functionalities and components.

Step 2: Set Up Cucumber:

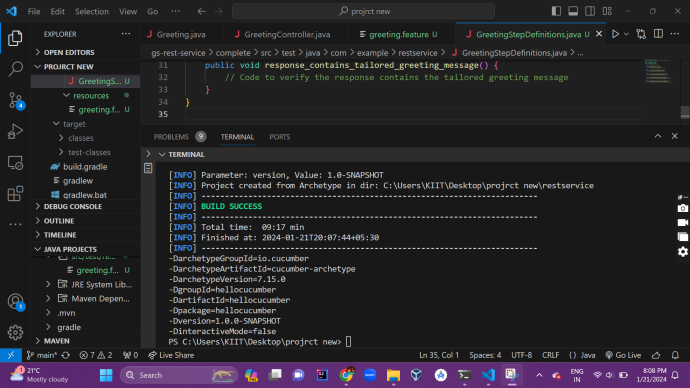
1. Add Cucumber Dependencies:

Open the project's build configuration file (pom.xml for Maven or build.gradle for Gradle). Add the Cucumber dependencies:



1. Set Up Cucumber Environment in IDE:

Install the Cucumber plugin in your IDE if not already installed



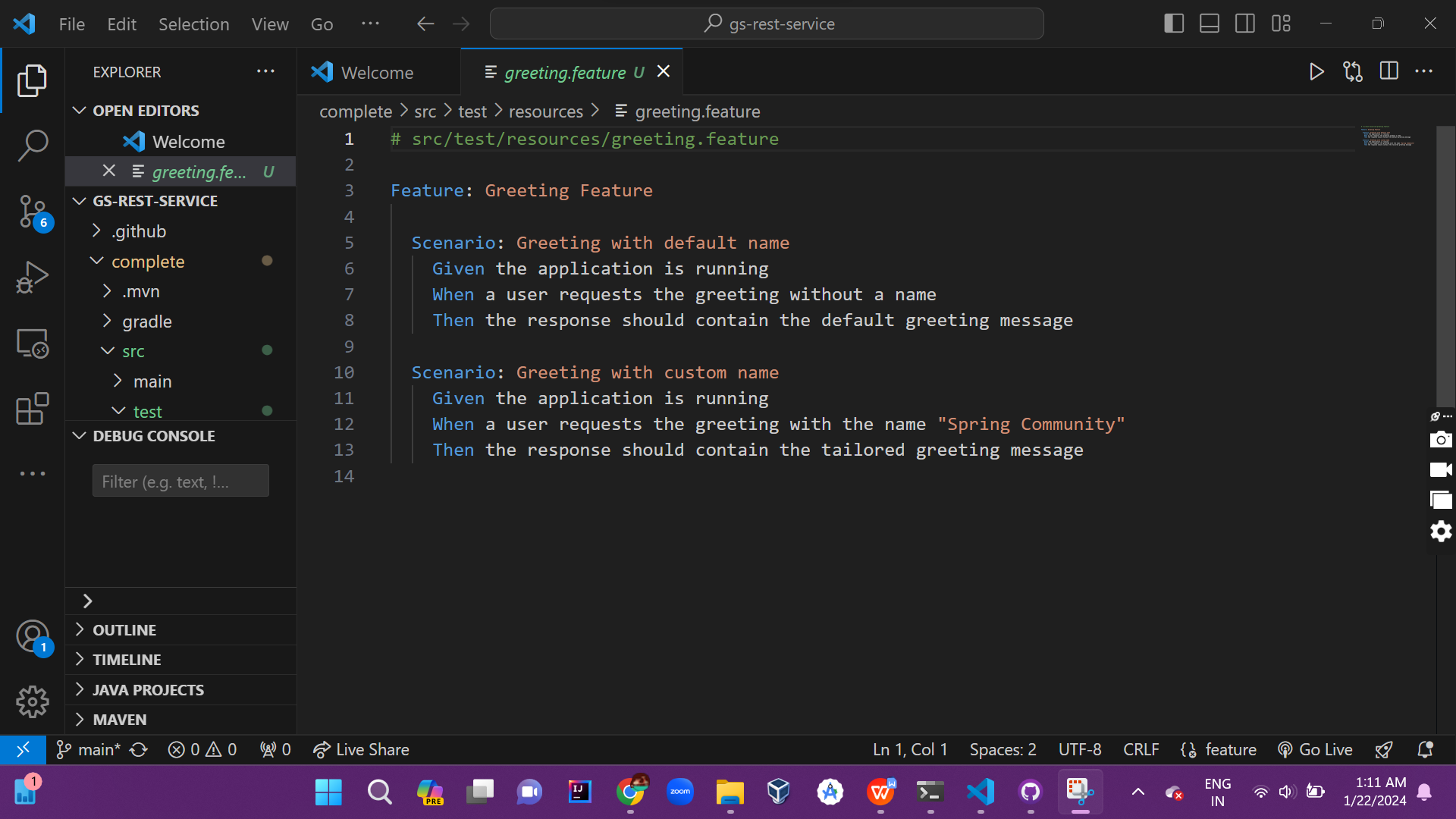
Step 3: Write Cucumber Test Scenarios:

1. Create Feature Files:

-In the src/test/resources directory, create feature files (e.g., sample.feature).

-Write test scenarios in Gherkin language, describing expected behavior.

1. Example Feature File:



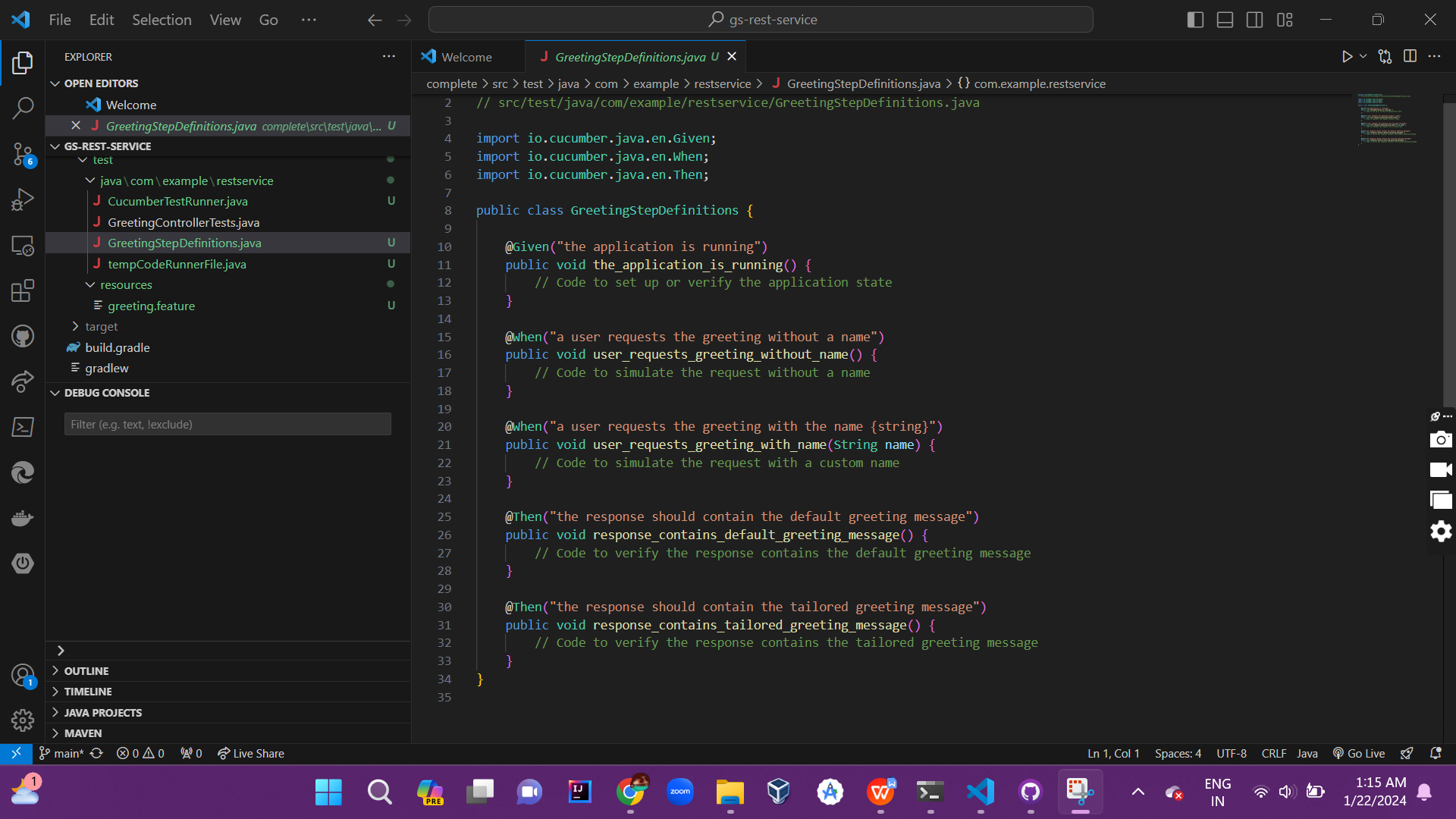
Step 4: Implement Step Definitions

1. Create Step Definition Classes:

- In the src/test/java directory, create step definition classes.

- Implement steps corresponding to scenarios in feature files.

1. Example Step Definition:

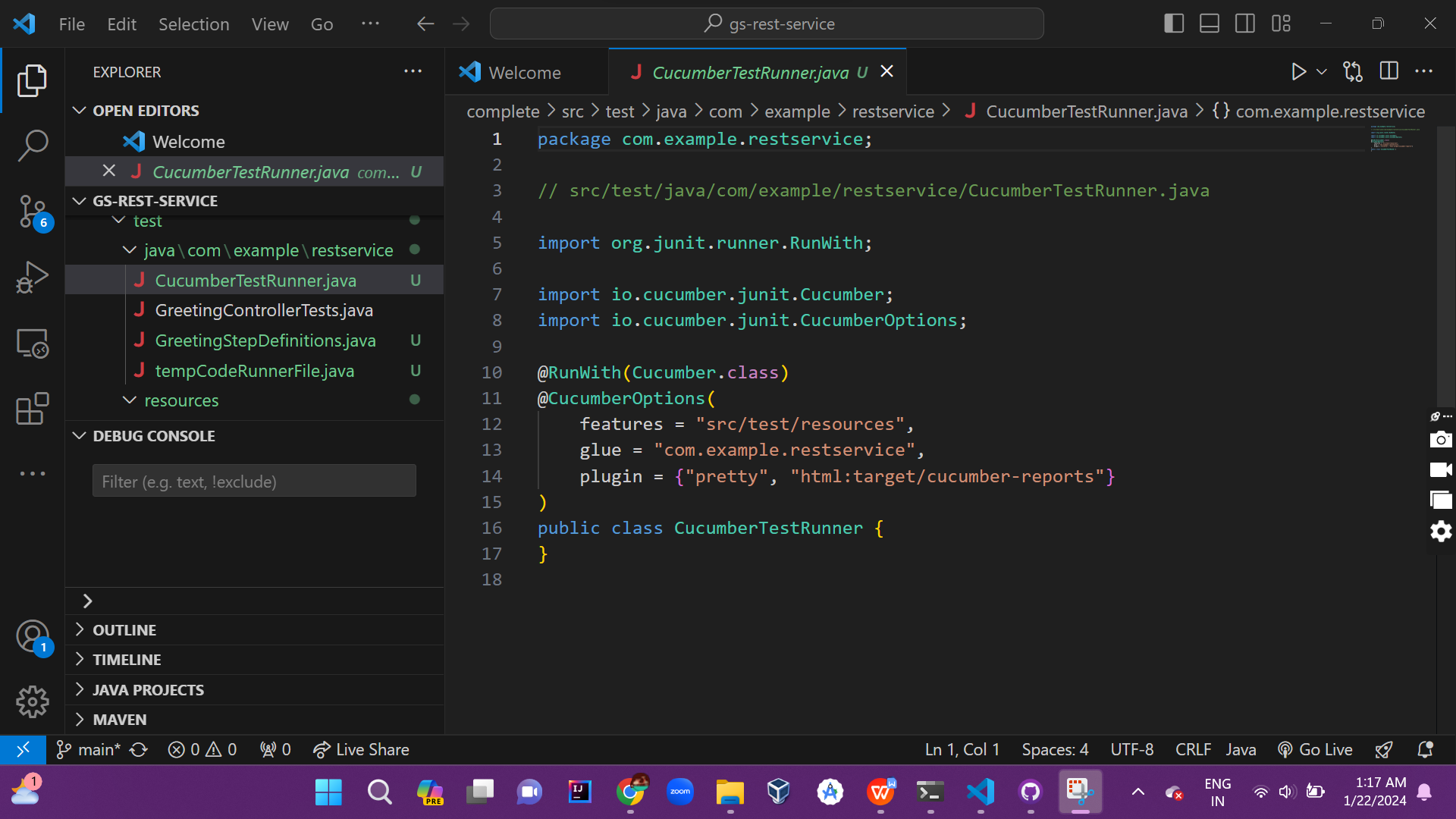


Step 5: Configure Cucumber Options:

1. Configure Cucumber Options:

- Create a runner class (e.g., TestRunner.java) or use a cucumber.properties file.

- Specify features, glue, plugins, etc.

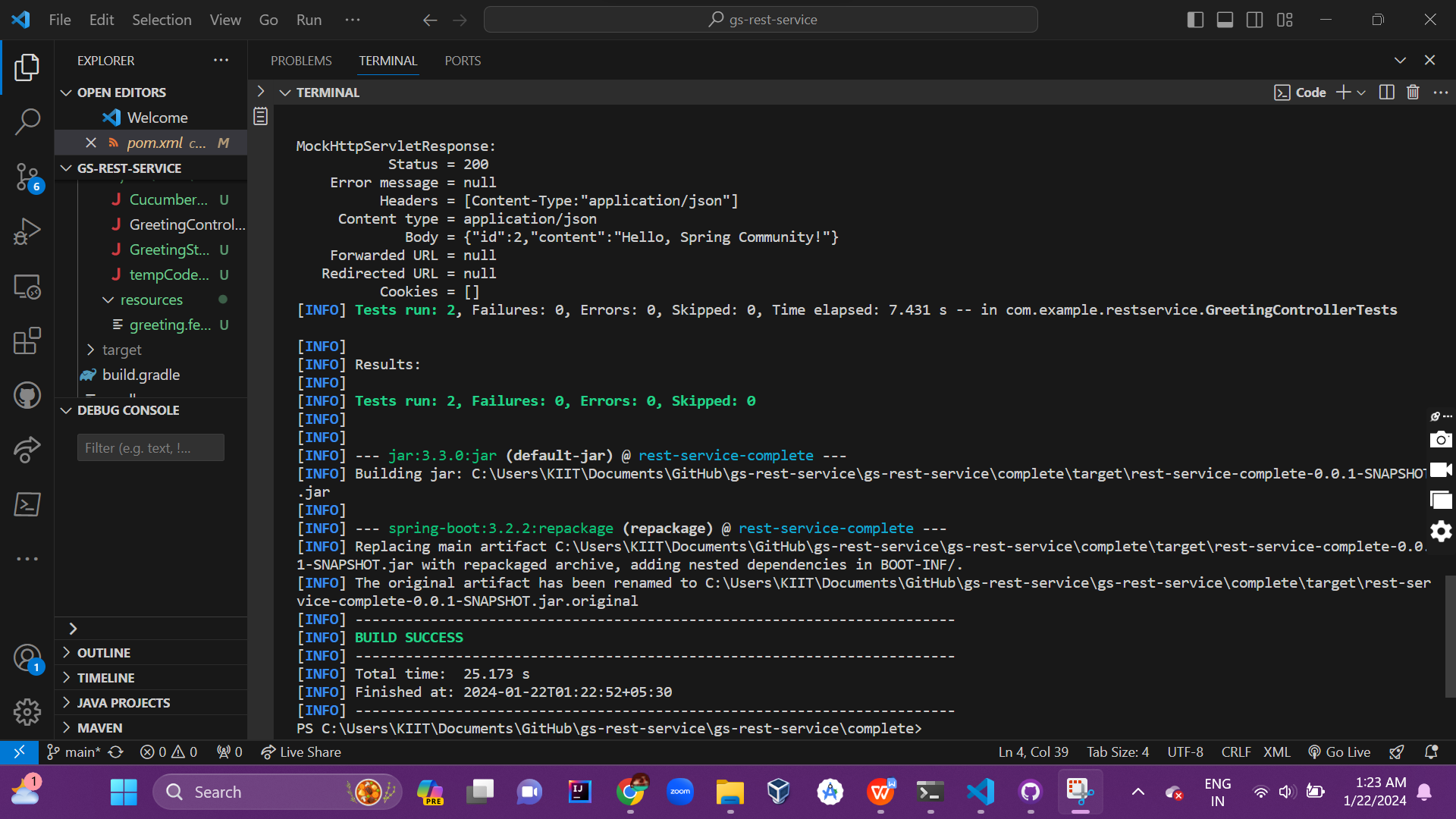


Step 6: Execute Tests and Generating Reports:

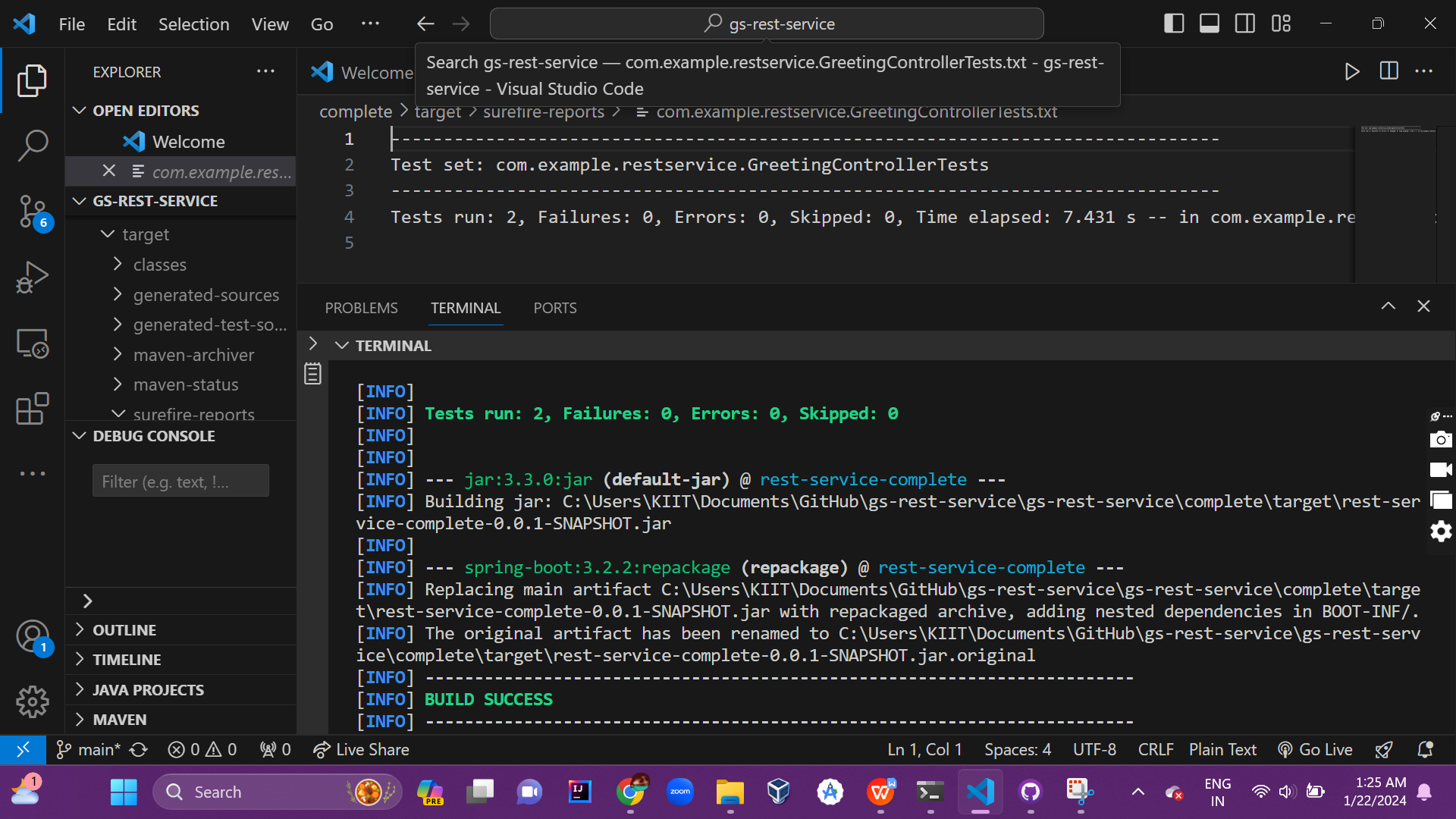
1. Run Tests:

- Execute the test runner class (TestRunner.java) to run Cucumber tests.

-- command -> mvn test or mvn clean test



Step 7: Review Test Reports:



Accordingly here are some areas of improvemment and they are :

1. Insufficient Test Data Management:

Issue: Tests dependent on specific data, leading to dependencies and test fragility.

Suggestions: Use test data factories to create and manage test data. Implement data setup and teardown mechanisms. Consider using database snapshots for data consistency

1. Test Failures:

Issue: Tests failing due to application defects or changes in functionality.

Suggestions: Review failed scenarios and identify the root cause of failures. Work closely with developers to address defects. Update affected test scenarios to reflect changes in application behavior.

1. Non-Deterministic Tests:

Issue: Tests producing inconsistent results or failing intermittently.

Suggestions: Identify and eliminate dependencies between tests. Ensure proper cleanup and reset of test data after each scenario. Implement wait mechanisms to handle asynchronous operations.

1. Reporting Insufficiencies:

Issue: Reports lack detail, making it challenging to interpret results.

Suggestions: Enhance Cucumber reports with additional plugins or reporting tools. Include screenshots or additional context in case of failures. Provide meaningful information in test failure messages.

1. Unrealistic Test Data:

Issue: Tests using unrealistic or static data that does not represent real-world scenarios.

Suggestions: Use a combination of realistic and edge-case data. Incorporate data variations

To simulate diverse user interactions. Consider data-driven testing for

a broader range of scenarios.

Conclusion :

This comprehensive guide outlines the steps to perform end-to-end testing on a Spring Boot Java project using Cucumber. Following these steps ensures a structured and efficient testing process, allowing for thorough analysis of application behavior.