Handling the Test Data and Executing It in Multiple Environments for User Module of an Application

Project Setup and Configuration

1. Create a new Maven project in Eclipse.
2. Configure the project to use Java 1.8.
3. Set up the project structure with feature files, step definitions, and supporting Java classes.

Add Dependencies

1. Add TestNG dependencies to the project's “pom.xml” file.
2. Configure Maven to handle dependencies.

```
<dependencies>
  <dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <version>5.3.0</version>
    <scope>test</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured-common -->
  <dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured-common</artifactId>
    <version>5.3.0</version>
  </dependency>
  <!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured-all -->
  <dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured-all</artifactId>
    <version>5.3.0</version>
    <scope>test</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/io.rest-assured/json-path -->
  <dependency>
    <groupId>io.rest-assured</groupId>
```

```
    <artifactId>json-path</artifactId>
    <version>5.3.0</version>
    <scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.json/json -->
<dependency>
    <groupId>org.json</groupId>
    <artifactId>json</artifactId>
    <version>20180813</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.hamcrest/hamcrest -->
<dependency>
    <groupId>org.hamcrest</groupId>
    <artifactId>hamcrest</artifactId>
    <version>2.2</version>
    <scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.15.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.7.1</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>7.10.1</version>
```

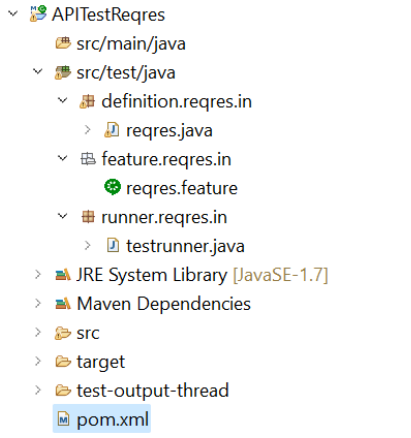
```
    </dependency>
    <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-junit</artifactId>
        <version>7.10.1</version>
        <scope>compile</scope>
    </dependency>
    <!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-core</artifactId>
        <version>7.10.1</version>
    </dependency>
    </dependency>
</dependencies>
```

Feature File Creation

1. Write Gherkin feature files for user registration, login, and user details.
2. Define scenarios with Given-When-Then steps to interact with the specified endpoints.

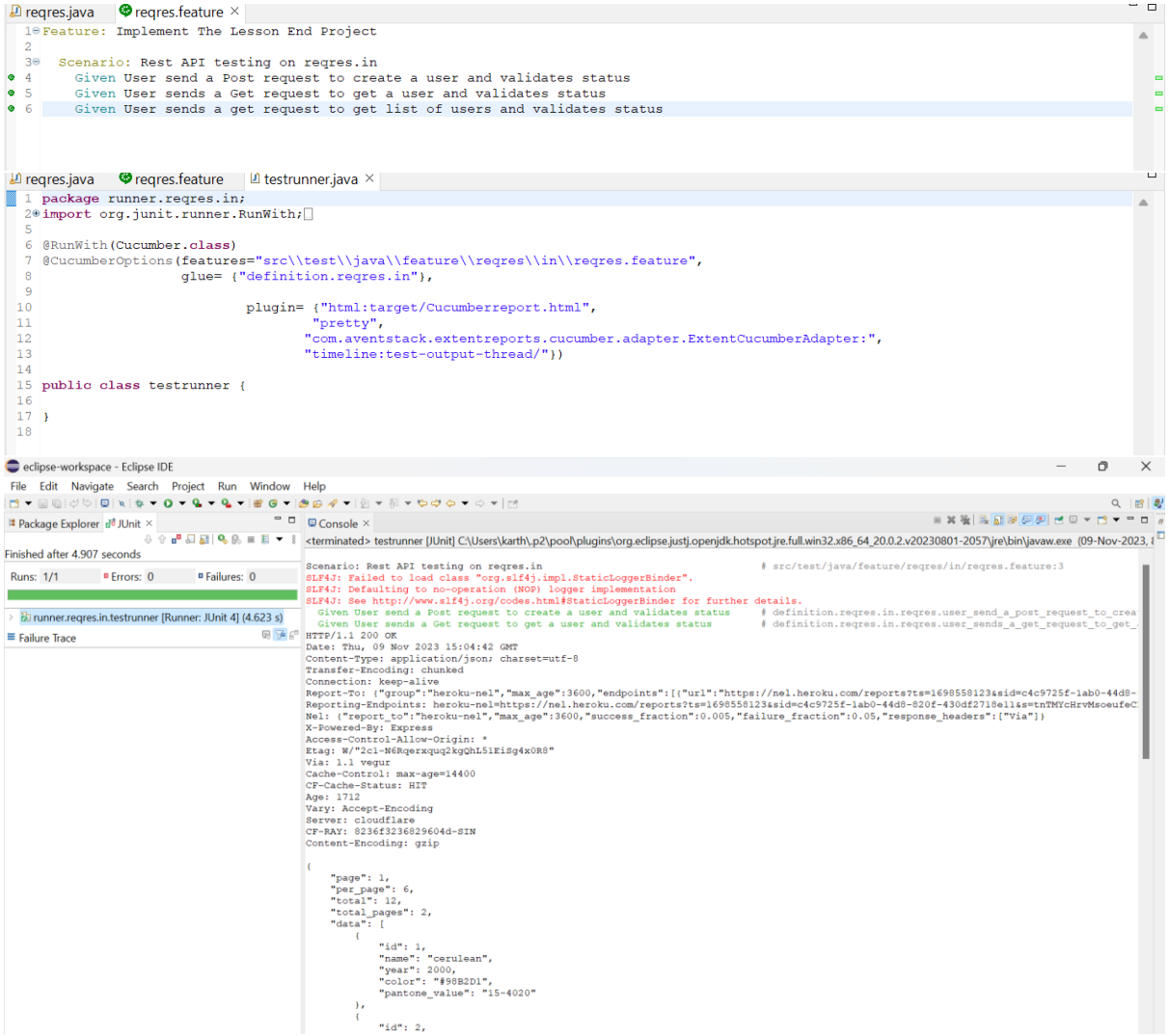
Step Definitions

- Create step definition classes in Java corresponding to the steps in your feature files.Implement Java methods to interact with the web services and perform data validation



reqres.java

```java
package definition.reqres.in;
import java.io.ObjectInputFilter.Config;
public class reqres {
    @Given("User send a Post request to create a user and validates status")
    public void user_send_a_post_request_to_create_a_user_and_validates_status() {

        JSONObject body = new JSONObject();
        body.put("name", "meghna");
        body.put("job", "developer");

        RestAssured.given()
        .baseUri("https://reqres.in")
        .basePath("/api/users")
        .contentType(ContentType.JSON)
        .body(body.toString())
        .when().post()
        .then().statusCode(201).log().ifError(); // log if there an error


    }


      @Given("User sends a Get request to get a user and validates status")
      public void user_sends_a_get_request_to_get_a_user_and_validates_status() {

              JSONObject body = new JSONObject();
              body.put("email", "meghna@gmail.com");
              body.put("password", "pas123");
              RestAssured.given() .baseUri("https://reqres.in")
              .contentType(ContentType.JSON) .body(body.toString())
              .when().post("/api/register")
              .then().statusCode(400);


       }


    @Given("User sends a get request to get list of users and validates status")
    public void user_sends_a_get_request_to_get_list_of_users_and_validates_status() {

        RestAssured.given()
        .baseUri("https://reqres.in")
        .basePath("/api/unknown")
        .when().get()
        .then().statusCode(200).log().all();
    }

}
```



GitHub repository : https://github.com/Karthick-Office/Project01.git

Handling the Test Data and Executing It in Multiple Environments for User Module of an Application Using REST Assured is present under the “REST Assured Practice Project” folder in the My GitHub repository