Postman:

Test Case 1: Add New User

POST - https://thinking-tester-contact-list.herokuapp.com/users

Body :

```
{
"firstName": "Dilip",
"lastName": "Kuldip",
"email": "{{UserEmail}}",
"password": "Heroku@123"
}
```

Pre request :

```
let dtime =  new Date().getTime();
let newEmail = "dilipkuldip2012" + dtime + "@gmail.com";

pm.environment.set("UserEmail", newEmail);
```

Post respone:

```
let res = pm.response;

pm.test("User created successfully and status code is 201", ()
=> {
   pm.expect(res.code).to.eql(201);
})

pm.test("User created successfully and status message is
Created", () => {
    pm.expect(res.status).to.eql("Created")
})

pm.environment.set("HerokuTokenID", res.json().token)
```

Response Validation :

PASSED User created successfully and status code is 201
PASSED User created successfully and status message is Created

Test Case 2: Get user Profile

GET - https://thinking-tester-contact-list.herokuapp.com/users/me

Bearer Token - {{HerokuTokenID}}

Post Response :

```
let res = pm.response;

pm.test("User created successfully and status code is 200", ()
=> {
    pm.expect(res.code).to.eql(200);
})

pm.test("User details displayed and status message is OK", () =>
{
     pm.expect(res.status).to.eql("OK")
})
```

Response Validation :

PASSED User created successfully and status code is 200
PASSED User details displayed and status message is OK

Test Case 3: Update User

PATCH - https://thinking-tester-contact-list.herokuapp.com/users/me

Bearer Token - {{HerokuTokenID}}

Post Response :

```
pm.test("Updated user details with status code 200", () => {
    pm.expect(pm.response.code).to.eql(200);
})

pm.test("Updated user details with status code OK", () => {
     pm.expect(pm.response.status).to.eql("OK")
})
```

Response Validation :

PASSED Updated user details with status code 200
PASSED Updated user details with status code OK

Test Case 4: Log In User

POST - https://thinking-tester-contact-list.herokuapp.com/users/login

Body :

```
{
"email": "{{UserEmail}}",
"password": "Heroku@1234"
}
```

Post Response :

```
let res = pm.response.json();

pm.test("Loggedin successfully with status code 200", () => {
   pm.expect(pm.response.code).to.eql(200);
})

pm.test("Loggedin successfully with status code OK", () => {
    pm.expect(pm.response.status).to.eql("OK")
})

pm.environment.set("LoginUserTokenID",res.token)
```

Response Validation :

PASSED Loggedin successfully with status code 200
PASSED Loggedin successfully with status code OK

Test Case 5: Add Contact

POST - https://thinking-tester-contact-list.herokuapp.com/contacts

Bearer Token - {{LoginUserTokenID}}

Body :

```
{
"firstName": "John",
"lastName": "Doe",
"birthdate": "1970-01-01",
"email": "jdoe@fake.com",
"phone": "8005555555",
"street1": "1 Main St.",
"street2": "Apartment A",
"city": "Anytown",
"stateProvince": "KS",
"postalCode": "12345",
"country": "USA"
}
```

Post Response :

```
pm.test("Contact added successfully with status code 201", () =>
{
   pm.expect(pm.response.code).to.eql(201);
})

pm.test("Contact added successfully with status code Created",
() => {
    pm.expect(pm.response.status).to.eql("Created")
})
```

Response Validation :

PASSED Contact added successfully with status code 201
PASSED Contact added successfully with status code Created

Test Case 6: Get Contact List

GET - https://thinking-tester-contact-list.herokuapp.com/contacts

Bearer Token - {{LoginUserTokenID}}

Post Response :

```
let res = pm.response.json()
pm.test("Contact list disaplyed with status code 200", () => {
   pm.expect(pm.response.code).to.eql(200);
})

pm.test("Contact list displayed with status message OK", () => {
    pm.expect(pm.response.status).to.eql("OK")
})

pm.environment.set("Contact ID", res[0]._id )
```

Response Validation :

PASSED Contact list displayed with status code 200
PASSED Contact list displayed with status message OK

Test Case 7: Get Contact

GET - https://thinking-tester-contact-list.herokuapp.com/contacts/{{Contact ID}}

Bearer Token - {{LoginUserTokenID}}

Post Response :

```
pm.test("Contact disaplyed with status code 200", () => {
   pm.expect(pm.response.code).to.eql(200);
})

pm.test("Contact displayed with status message OK", () => {
    pm.expect(pm.response.status).to.eql("OK")
})
```

Response Validation :

PASSED Contact disaplyed with status code 200
PASSED Contact displayed with status message OK

Test Case 8: Update Contact

PUT - **https://thinking-tester-contact-list.herokuapp.com/contacts/{{Contact ID}}**

Bearer Token - {{LoginUserTokenID}}

Body :

```
{
"firstName": "Amy",
"lastName": "Miller",
"birthdate": "1992-02-02",
"email": "amiller@fake.com",
"phone": "8005554242",
"street1": "13 School St.",
"street2": "Apt. 5",
"city": "Washington",
"stateProvince": "QC",
"postalCode": "A1A1A1",
"country": "Canada"
}
```

Post Response :

```
let res = pm.response.json();

pm.test("Contact details updated successfully with status code
200", () => {
   pm.expect(pm.response.code).to.eql(200);
})

pm.test("Contact details updated successfully with status
message OK", () => {
    pm.expect(pm.response.status).to.eql("OK")
})

pm.test("Validating Email is amiller@fake.com", () =>{
    pm.expect(res.email).to.eql("amiller@fake.com")
})
```

Response Validation :

PASSED Contact details updated successfully with status code 200
PASSED Contact details updated successfully with status message OK
PASSED Validating Email is amiller@fake.com

Test Case 9: Update Contact

PATCH-https://thinking-tester-contact-list.herokuapp.com/contacts/{{Contact ID}}

Bearer Token - {{LoginUserTokenID}}

Body:

```
{
"firstName": "Anna"
}
```

Post Response :

```
let res = pm.response.json();

pm.test("Firstname updated successfully with status code 200",
() => {
   pm.expect(pm.response.code).to.eql(200);
})

pm.test("Firstname updated successfully with status message OK",
() => {
    pm.expect(pm.response.status).to.eql("OK")
})

pm.test("Validating FirstName is Anna", () =>{
    pm.expect(res.firstName).to.eql("Anna")
})
```

Response Validation :

PASSED Firstname updated successfully with status code 200
PASSED Firstname updated successfully with status message OK
PASSED Validating FirstName is Anna

Test Case 10: Logout User

POST - https://thinking-tester-contact-list.herokuapp.com/users/logout

Bearer Token - {{LoginUserTokenID}}

Post Response :

```
pm.test("Logged out successfully with status code 200", () => {
    pm.expect(pm.response.code).to.eql(200);
})

pm.test("Logged out successfully with status message OK", () =>
{
     pm.expect(pm.response.status).to.eql("OK")
})
```

Response Validation :

PASSED Logged out successfully with status code 200
PASSED Logged out successfully with status message OK

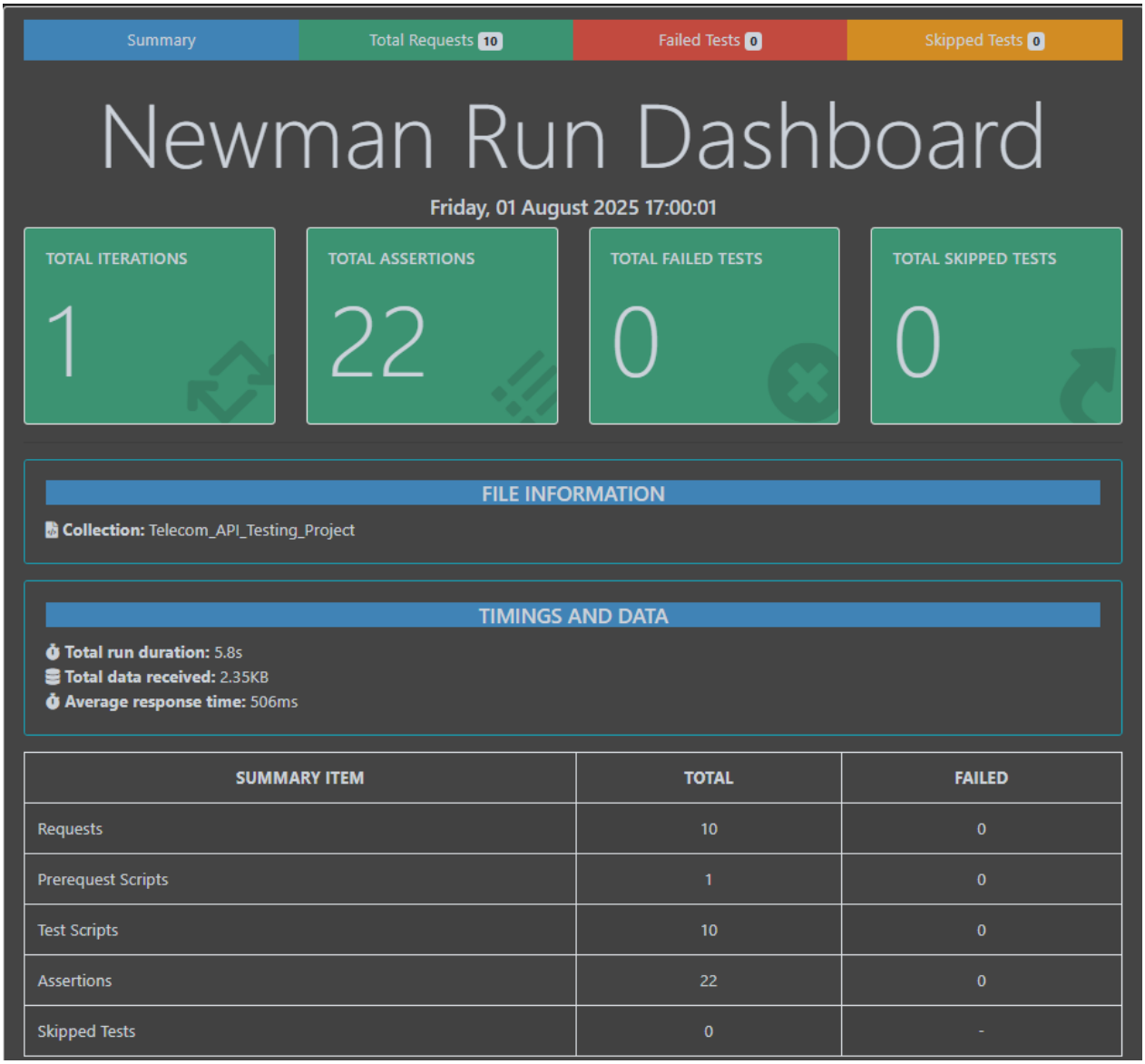
Environment -

Telecom_APIProject.postman_environment.json

Collection -

Telecom_API_Testing_Project.postman_collection.json

Result -



RestAssured:

ExtentTestListener.java:

```
package com.telecom.extentreportlistener;

import org.testng.ISuite;
import org.testng.ISuiteListener;
import org.testng.ITestContext;
```

```java
import org.testng.ITestListener;
import org.testng.ITestResult;

import com.aventstack.extentreports.ExtentReports;
import com.aventstack.extentreports.ExtentTest;
import com.aventstack.extentreports.reporter.ExtentSparkReporter;

public class ExtentTestListener implements ITestListener,
ISuiteListener {

    private static ExtentReports extent;
    private static ExtentTest test;

    @Override
    public void onStart(ISuite suite) {
        ExtentSparkReporter spark = new
ExtentSparkReporter("ExtentReport.html");
        extent = new ExtentReports();
        extent.attachReporter(spark);
        extent.setSystemInfo("Suite", suite.getName());
    }

    @Override
    public void onFinish(ISuite suite) {
        extent.flush();
    }

    @Override
    public void onTestStart(ITestResult result) {
        test = extent.createTest(result.getMethod().getMethodName());
    }

    @Override
    public void onTestSuccess(ITestResult result) {
        test.pass("Test passed");
    }

    @Override
    public void onTestFailure(ITestResult result) {
        test.fail("Test failed: " + result.getThrowable());

    }

    @Override
    public void onTestSkipped(ITestResult result) {
        test.skip("Test skipped");
```

```
    }

    @Override
    public void onStart(ITestContext context) {

    }

    @Override
    public void onFinish(ITestContext context) {

    }
}
```

ContactListApplicationTest.java:

```
package com.telecom.testcases;

import static io.restassured.RestAssured.given;

import java.util.HashMap;

import org.testng.Assert;
import org.testng.annotations.Test;

import io.restassured.response.Response;


public class ContactListApplicationTest {

      String userToken;
      String loginToken;
      String userEmail;
      String firstContactID;
      String updatedPwd;
      String updatedEmail;

      @Test(priority = 1)

      public void addNewUser() {

            userEmail = "dilipkool" + System.currentTimeMillis() +
"@gmail.com";

            HashMap<String, Object> data = new HashMap<String,
Object>();
            data.put("firstName", "Dilip");
```

```
            data.put("lastName", "Kuldip");
            data.put("email", userEmail);
            data.put("password", "Heroku@123");

            Response response = given().header("Content-Type",
"application/json").body(data).when()
                        .post("https://thinking-tester-contact-
list.herokuapp.com/users");

            response.then().log().all();

            Assert.assertEquals(response.getStatusCode(), 201, "Expected
status code 201 Created");

            String statusLine = response.getStatusLine();

            Assert.assertTrue(statusLine.contains("Created"), "Expected
status line to contain 'Created'");

            userToken = response.jsonPath().getString("token");

      }

      @Test(priority = 2, dependsOnMethods = "addNewUser")

      public void getUserProfile() {

            Response response = given().header("Content-Type",
"application/json")
                        .header("Authorization", "Bearer " +
userToken).when()
                        .get("https://thinking-tester-contact-
list.herokuapp.com/users/me");

            Assert.assertEquals(response.statusCode(), 200);

            String statusLine = response.getStatusLine();

            Assert.assertTrue(statusLine.contains("OK"), "Expected
status line to contain 'OK'");

      }

      @Test(priority = 3)

      public void updateUser() {
```

```
            updatedEmail = "yamuna" + System.currentTimeMillis() +
"@gmail.com";
            updatedPwd = "Heroku@1234";

            HashMap<String, Object> data = new HashMap<String,
Object>();
            data.put("firstName", "Yamuna");
            data.put("lastName", "Anisetti");
            data.put("email", updatedEmail);
            data.put("password", updatedPwd);

            Response response = given().header("Content-Type",
"application/json")
                        .header("Authorization", "Bearer " +
userToken).body(data).when()
                        .patch("https://thinking-tester-contact-
list.herokuapp.com/users/me");

            Assert.assertEquals(response.getStatusCode(), 200, "Expected
status code 200 OK");

            String statusLine = response.getStatusLine();

            Assert.assertTrue(statusLine.contains("OK"), "Expected
status line to contain 'OK'");

            response.then().log().all();
      }

      @Test(priority = 4)

      public void loginUser() {

            HashMap<String, Object> data = new HashMap<String,
Object>();
            data.put("email", updatedEmail);
            data.put("password", updatedPwd);

            Response response = given().header("Content-Type",
"application/json").body(data).when()
                        .post("https://thinking-tester-contact-
list.herokuapp.com/users/login");

            Assert.assertEquals(response.getStatusCode(), 200, "Expected
status code 200 OK");
```

```
            String statusLine = response.getStatusLine();

            Assert.assertTrue(statusLine.contains("OK"), "Expected
status line to contain 'OK'");

            loginToken = response.jsonPath().getString("token");

            response.then().log().all();
      }

      @Test(priority = 5, dependsOnMethods = "loginUser")


      public void addContact() {

            HashMap<String, Object> data = new HashMap<String,
Object>();
            data.put("firstName", "Ram");
            data.put("lastName", "nithin");
            data.put("birthdate", "1977-01-31");
            data.put("email", "ram@gmail.com");
            data.put("phone", "9848754623");
            data.put("street1", "Sector 11");
            data.put("street2", "Rahul Enclave");
            data.put("city", "Vijayawada");
            data.put("stateProvince", "Andhra Pradesh");
            data.put("postalCode", "528965");
            data.put("country", "India");

            Response response = given().header("Content-Type",
"application/json")
                        .header("Authorization", "Bearer " +
loginToken).body(data).when()
                        .post("https://thinking-tester-contact-
list.herokuapp.com/contacts");

            Assert.assertEquals(response.getStatusCode(), 201, "Expected
status code 201 Created");

            String statusLine = response.getStatusLine();

            Assert.assertTrue(statusLine.contains("Created"), "Expected
status line to contain 'Created'");

            response.then().log().all();
```

```
	}

	@Test(priority = 6, dependsOnMethods = "loginUser")

	public void getContactList() {

		Response response = given().header("Content-Type",
"application/json")
				.header("Authorization", "Bearer " +
loginToken).when()
				.get("https://thinking-tester-contact-
list.herokuapp.com/contacts");

		Assert.assertEquals(response.getStatusCode(), 200, "Expected
status code 200 OK");

		String statusLine = response.getStatusLine();

		Assert.assertTrue(statusLine.contains("OK"), "Expected
status line to contain 'OK'");

		firstContactID = response.jsonPath().getString("[0]._id");

		System.out.println("Contact Token is :" + firstContactID);

		response.then().log().all();

	}

	@Test(priority = 7)

	public void getContactById() {

		Response response = given().header("Content-Type",
"application/json")
				.header("Authorization", "Bearer " +
loginToken).when()
				.get("https://thinking-tester-contact-
list.herokuapp.com/contacts/" + firstContactID);

		Assert.assertEquals(response.getStatusCode(), 200, "Expected
status code 200 OK");

		String statusLine = response.getStatusLine();
```

```
            Assert.assertTrue(statusLine.contains("OK"), "Expected
status line to contain 'OK'");

            response.then().log().all();
      }

      @Test(priority = 8)


      public void updateFullContact() {

            HashMap<String, Object> data = new HashMap<String,
Object>();
            data.put("firstName", "Amy");
            data.put("lastName", "Miller");
            data.put("birthdate", "1998-07-04");
            data.put("email", "amiller@fake.com");
            data.put("phone", "984888845");
            data.put("street1", "DTS");
            data.put("street2", "Road no 25 MVP Colony");
            data.put("city", "Vizag");
            data.put("stateProvince", "AP");
            data.put("postalCode", "530017");
            data.put("country", "India");

            Response response = given().header("Content-Type",
"application/json")
                        .header("Authorization", "Bearer " +
loginToken).body(data).when()
                        .put("https://thinking-tester-contact-
list.herokuapp.com/contacts/" + firstContactID);

            Assert.assertEquals(response.getStatusCode(), 200, "Expected
status code 200 OK");

            String statusLine = response.getStatusLine();

            Assert.assertTrue(statusLine.contains("OK"), "Expected
status line to contain 'OK'");

            String email = response.jsonPath().getString("email");

            Assert.assertEquals(email, "amiller@fake.com", "Expected
updated email to match");

            response.then().log().all();
```

```
	}

	@Test(priority = 9)
	public void updatePartialContact() {

		String requestBody = "{\n" + "  \"firstName\": \"Anna\"\n" +
"}";

		Response response = given().header("Content-Type",
"application/json")
				.header("Authorization", "Bearer " +
loginToken).body(requestBody).when()
				.patch("https://thinking-tester-contact-
list.herokuapp.com/contacts/" + firstContactID);

		Assert.assertEquals(response.getStatusCode(), 200, "Expected
status code 200 OK");

		String statusLine = response.getStatusLine();

		Assert.assertTrue(statusLine.contains("OK"), "Expected
status line to contain 'OK'");

		String firstName =
response.jsonPath().getString("firstName");

		Assert.assertEquals(firstName, "Anna", "Expected updated
first name to be 'Anna'");

		response.then().log().all();
	}

	@Test(priority = 10)
	public void logoutUser() {

		Response response = given().header("Content-Type",
"application/json")
				.header("Authorization", "Bearer " +
loginToken).when()
				.post("https://thinking-tester-contact-
list.herokuapp.com/users/logout");

		Assert.assertEquals(response.getStatusCode(), 200, "Expected
status code 200 OK");

		String statusLine = response.getStatusLine();
```

```
            Assert.assertTrue(statusLine.contains("OK"), "Expected
status line to contain 'OK'");

            response.then().log().all();
      }
}
```

testing.xml:

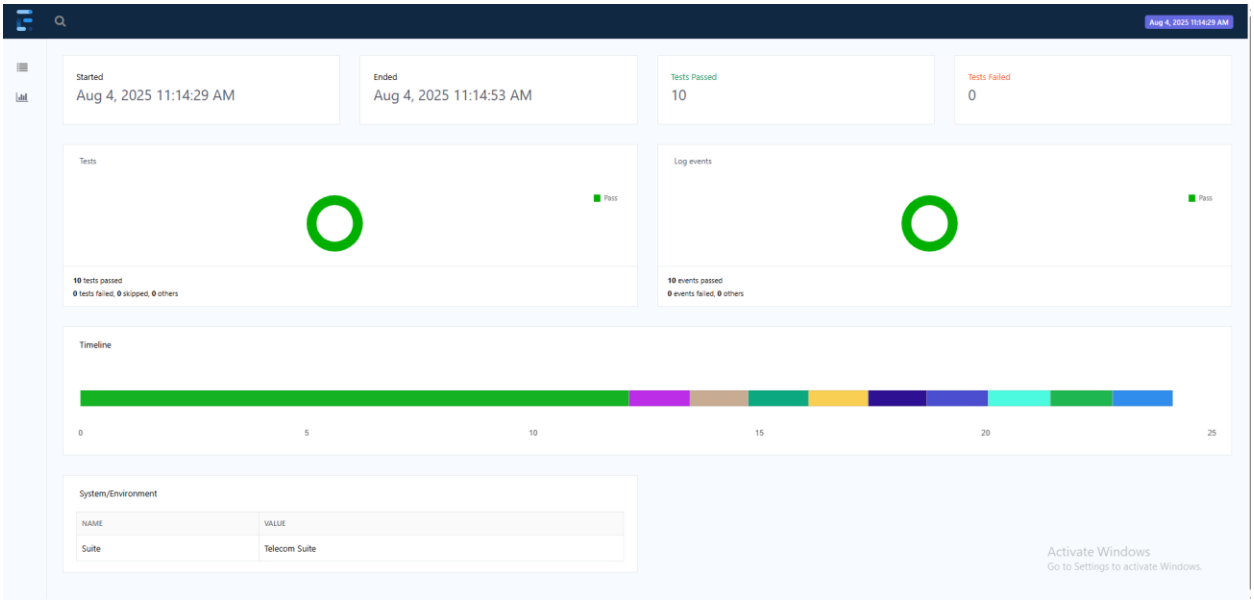
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Telecom Suite">

      <listeners>
            <listener
                  class-
name="com.telecom.extentreportlistener.ExtentTestListener" />
      </listeners>

      <test name="Contact List API Testing">
            <classes>
                  <class
name="com.telecom.testcases.ContactListApplicationTest" />
            </classes>
      </test>

</suite>
```

Result:



GitHub repo: https://github.com/Dilip2012/Telecom-Domain-API-Project/upload

Project Folder Structure:

```
Telecom_API_Testing_Project/
├── src/
│   ├── main/
│   │   └── java/
│   │   └── com/telecom/extentreportlistener/
│   │   └── ExtentTestListener.java
│   └── test/
│   ├── java/
│   │   └── com/telecom/testcases/
│   │   └── ContactListApplicationTest.java
│   └── resources/
├── Postman_Collection/
│   └── Telecom_API_Testing_Project.postman_collection.json
├── Postman_Environment/
│   └── Telecom_APIProject.postman_environment.json
├── Postman_Result/
│   └── NewmanReport.html
├── test-output/
├── target/
├── ExtentReport.html
├── testNG.xml
├── pom.xml
└── README.md
```

Running Postman Collection using Newman:

Run Collection Without Environment:

```
newman run
Postman_Collection/Telecom_API_Testing_Project.postman_collection.json
```

Run Collection With Environment File:

```
newman run
Postman_Collection/Telecom_API_Testing_Project.postman_collection.json
-e Postman_Environment/Telecom_APIProject.postman_environment.json
```

Generate HTML Report with htmlextra Reporter:

```
npm install -g newman-reporter-htmlextra
```

Then:

```
newman run
Postman_Collection/Telecom_API_Testing_Project.postman_collection.json
-e Postman_Environment/Telecom_APIProject.postman_environment.json -r
htmlextra --r eporter-htmlextra-export
Postman_Result\NewmanReport.html
```

After execution:

✅ **View Newman Report**: Open NewmanReport.html in your browser.

Running RestAssured TestNG Suite:

Maven Command:

```
mvn test

OR

Use the TestNG XML file:
Right-click on testNG.xml → Run As → TestNG Suite
```

After execution:

✅ **View Extent Report:** Open `ExtentReport.html` in your browser.