Author: Hussain hussain.si1999@gmail.com

Project Documentation

The project structure is organized based on the **TestNG framework** and **Page Object Model (POM)** design pattern. Each logical section of the application, such as **API endpoints** and **test case validations**, is encapsulated within separate packages.

Prerequisites:

1. **Install JDK**
2. **Install Intellij**
3. **Install Maven**

Project Structure:(src/test/java)

org.project.endpoint:

- **Houses classes defining different API endpoints (e.g., AlbumEndpoint, CommentEndpoint, PhotosEndpoint, etc).**
- **Each class encapsulates the logic for interacting with a specific API resource.**

Endpoint Classes:

AlbumEndpoint, CommentEndpoint, PhotosEndpoint, etc.,

- **Each class includes methods to perform API requests (e.g., getAlbums, getAlbumById).**
- **Utilizes RestAssured for building requests and extracting responses.**

org.project.testCaseValidation:

- **Contains classes responsible for validating API responses (e.g., validationForAlbum, validationForComment, validationForPhotos, etc).**
- **These classes use RestAssured for making API calls and TestNG assertions for validation.**

Validation Classes:

validationForAlbum, validationForComment, validationForPhotos, etc.,

- **These classes contain TestNG test methods for validating API responses.**
- **The setup method in each class sets the base URI using RestAssured.**

TestNG XML Configuration (testng.xml):

- **Defines the test suite structure, including the classes to be executed and their order.**
- **Promotes parallel execution of test classes for efficiency.**

Pom.xml:

Add dependency for testNG and RestAssured

```
    <!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured -->
    <dependency>
      <groupId>io.rest-assured</groupId>
      <artifactId>rest-assured</artifactId>
      <version>5.1.1</version>
      <scope>test</scope>
    </dependency>
  <!-- https://mvnrepository.com/artifact/org.testng/testng -->
  <dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.0.0</version>
    <scope>test</scope>
  </dependency>
```

For parallel Execution added plugin

```
 <build>
  <plugins>
   <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>3.0.0-M5</version>
    <configuration>
     <suiteXmlFiles>
      <suiteXmlFile>./testSuite/testng.xml</suiteXmlFile>
     </suiteXmlFiles>
     <parallel>methods</parallel>
     <threadCount>2</threadCount>
    </configuration>
   </plugin>
  </plugins>
 </build>
```

This project is organized using the Page Object Model (POM) framework, separating concerns into packages for endpoints, test case validations, and test suites. The TestNG framework is used for test execution, and the project is built and managed using Maven.

Execution Steps:

- **Ensure that Java, IntelliJ IDEA, and Maven are installed on the machine.**
- **Clone the project repository from its source.**
- **Open the project in IntelliJ IDEA.**
- **Run desired test validation classes**
- **Execute the entire test suite by running the testng.xml file**

Test Results:

- **Observe test results, assertions, and execution logs in the IntelliJ IDEA console.**
- **View detailed TestNG reports generated after test execution, located in the test-output directory where we have HTML report named as index.html.**

Brief Document:

Page Object Model (POM):

- **Divided the codebase into distinct sections, separating API endpoints and test case validations.**
- **Promotes maintainability and reusability by encapsulating related functionalities in dedicated classes.**

RestAssured and TestNG:

- **Used RestAssured for simplifying API testing through a fluent and expressive syntax.**
- **Employed TestNG as the testing framework for organizing and executing tests.**

Potential Improvements:

- **Enhanced Error Handling: Implement more robust error handling mechanisms to capture and report errors effectively.**

- **Increased Test Coverage: Expand test coverage by adding more test cases, covering different scenarios and edge cases.**

- **Parameterized Test Data: Enhance reusability and flexibility by parameterizing test data, making tests more versatile.**

- **Parallel Test Execution: Investigate and implement parallel test execution to optimize testing efficiency.**

Challenges Faced in API Testing:

- **Constructing complex requests with various parameters, headers, and payloads can be challenging.**
- **Managing diverse test data for different test scenarios.**
- **Testing the same functionality with different input combinations.**

THANK YOU