FreeNow Coding Challenge Documentation

Tools Used:

Language - Java 1.8
Library - RestAssured
Built in tool - Maven
CI tool - CircleCi
Public Repository - GitHub
Unit Test Framework - TestNg
Test Framework - Hybrid
Code Editor - Eclipse

Framework Setup:

1. Created a maven project by navigating to File->Others->Create a Maven project.
2. Used the appropriate maven dependencies for RestAssured, TestNg and all other relevant dependencies in the pom.xml file.
3. Created five packages under src->test->java folder. The package and its functionalities are listed below:

(i) apiTests -> Contains test class(WorkflowTest.java) that comprises of the test methods. Below is a snapshot of how a test method is written.

```
@Test(priority=1)
	public void validateUserBlogDetails() {
		try {
			validationCaller.getUserId("username", System.getProperty("propertyName"));
			validationCaller.verifyUserPosts();
			validationCaller.addPostCommentsAndVerifyEmailFormat();

		}
		catch(Exception e) {
			Reporter.log("Exception is" +e);
			Assert.assertTrue(false, e.getStackTrace().toString());
		}

	}
```

(ii) assertions -> Contains assertion class(ValidationCaller.java) that comprises of assertion methods that the test class calls.

(iii) bin -> Contains DTO classes for users, posts and comments api that are basically POJO implementations of json responses received.

(iv) common -> Contains Endpoints interface that comprises of all api endpoints and Utility class that comprises of methods to read reference excel file.

(v) springRestServices -> Contains classes that returns the request specification and responses for all the get apis.

4. Created a package under src->main->java folder. The package and its functionality is listed below:

config -> This package contains RestAssuredConfig.java class which comprises of the reusable rest assured configurations used in this project.

5. There is a folder created under src->test->resources folder . The package and its functionality is listed below:

PostTitleAndBody -> This folder comprises of the reference excel file which contains reference post body and title details used for validation purpose.

6. testNg.xml file controls the test classes used for test execution. The package name is mentioned representing the class files that should be picked up for test execution. Below is a snapshot of the testNg file used.

```
<test name="FreeNowApiTests">
    <packages>
      <package name="config"/>
      <package name="apiTests"/>
    </packages>
  </test>
```

7. pom.xml controls the testNg.xml file and is used to trigger the test. The surefire plugin used in the pom file invokes the testNg file. Below is the snapshot of the control feature that the pom has on testNg.

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
        <configuration>
                <systemPropertyVariables>
                        <propertyName>Samantha</propertyName>
                </systemPropertyVariables>
                <suiteXmlFiles>
                        <suiteXmlFile>testNg.xml</suiteXmlFile>
                </suiteXmlFiles>
        </configuration>
</plugin>
```

8. We have a config.yml file written inside .circleci folder that helps in continuous integration and deployment process. The file is configured so that whenever we initiate a deployment to the respective branch a build is auto triggered in circleci.The config file uses the maven test command to run the tests in the pipeline invoking the pom file. Below is a snapshot of the config file:

```
workflows:
  maven_test:
    jobs:
```

```
  - maven/test
version: 2
```

Api Details :

Base uri -> https://jsonplaceholder.typicode.com
getUserEndpoint -> /users
getPostEndpoint -> /posts
getCommentEndpoint -> /comments

Test Method Descriptions:

Test Method **#1**: This test method automates the entire workflow. The test method and the workflow tells us to search for the user with "Samantha" as the username using the getUser() api and then fetch the user id and pass it to the getPost() api to retrieve the post bodies and titles. The postId is used from this api to hit the getcomments() api to store the comments body and title and validate the email format as presented in the comments section.

Actual Result : The apis respond with a status code of 200.
Expected Result : It should match with the actual result.

Test Case Status: Passed.

Test Method **#2:** This method hits the user api with the query parameter as usernames instead of username. The field usernames is an invalid parameter according to the api structure. Henceforth we expect a not found(404) status code for this api response.

Actual Result : The api responds with a status code of 200.
Expected Result : The api should respond with a status code of 404.

Test Case Status : Failed.

Test Method **#3:** This test method uses an invalid alphanumeric parameter passed as an argument to the post api. The api status code returned in this case should be 200 but the response body should return a null array.

Actual Result : The api responds with a status code of 200 and with an empty response array.
Expected Result : It should match with the actual result.

Test Case Status: Passed.

Test Method **#4:** This test method tests the validation of the email key in the json response of the getComments() api. If the key shows up as anything else other than email then the test response should return a status code of 500 and the test case is meant to fail with a message printed in the log file.
Actual Result : The api responds with a status code of 200.
Expected Result : It should match with the actual result.