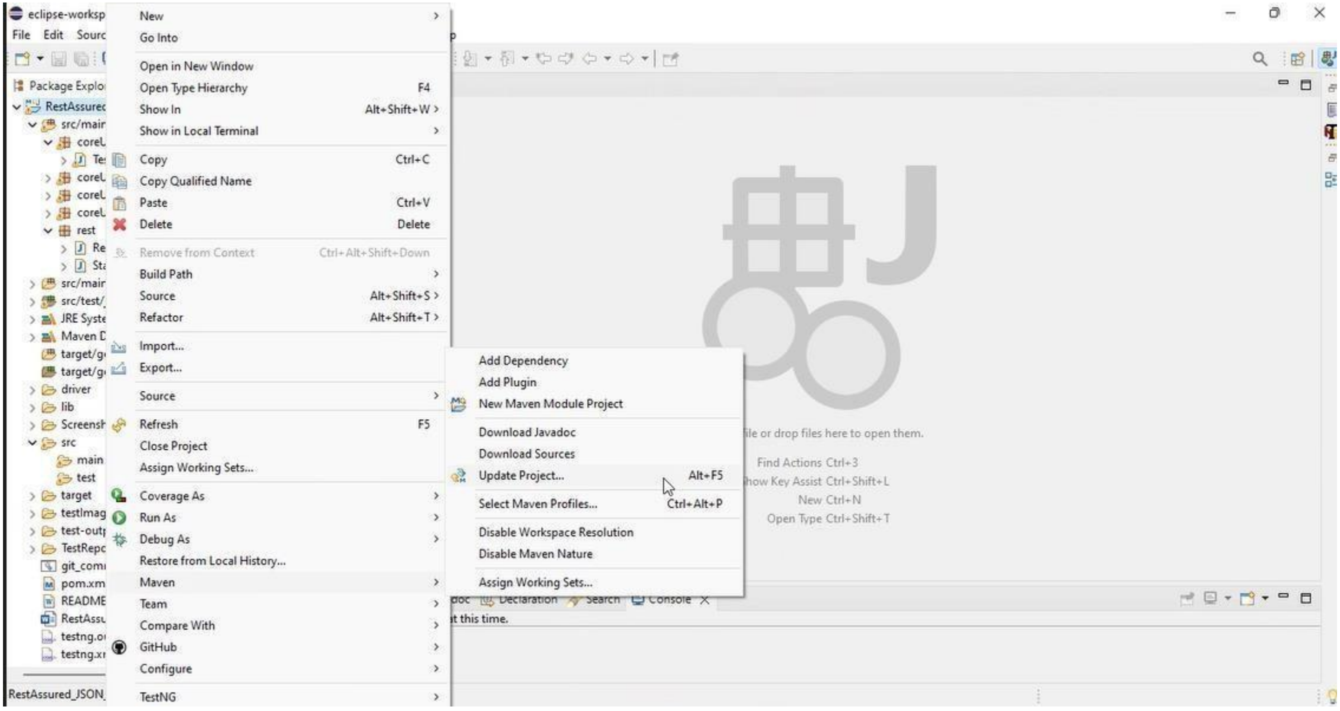
RestAssured API Automation Project

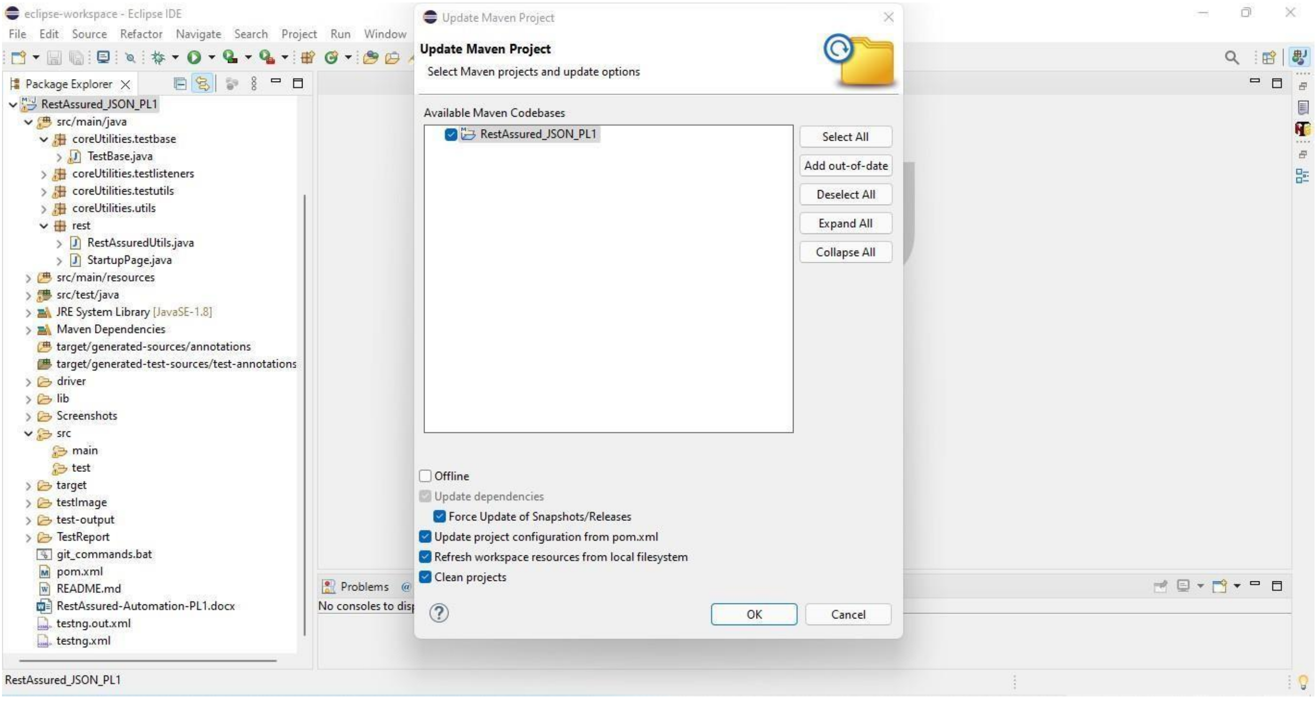
Pre-requisite:

As soon as you import a project in eclipse, update the project using the maven update option as below. This is to resolve issue if any maven dependency not downloaded properly:

1. Right click on project : Go to “Maven” : Select “Update Project”



2. In Update Maven Project Box Select “Force Update of Snapshots/Releases” and click OK



Update Maven Project

Select Maven projects and update options

Available Maven Codebases

☑ RestAssured_JSON_PL1

☐ Offline
☐ Update dependencies
☑ Force Update of Snapshots/Releases
☑ Update project configuration from pom.xml
☑ Refresh workspace resources from local filesystem
☑ Clean projects

Template Code Structure:

a. Below are the packages and files you will be required to work upon.

b. Other Files and packages you can ignore.

c. In other Files and packages do not do any changes. It would affect your evaluation.

d. You are not required to work in "Test" Folder. Files there are non-editable. Editing those files and trying to save them will throw error and would affect your evaluation.

Package	Class/File	Description
src/main/java/coreUtilities/utils/	FileOperations.java	1. Contains methods to read from excel file. 2. Method is in templated form. **3. You will be required to implement these methods as very first activity, because for creating post the data should be read from excel.**
/src/main/java/rest	ApiUtil.java	1. All core activities to be performed here. 2. The comments associated with each templated method here describe the expectation. 3. Declare any variable/object you need to share data/status between different methods. 4. Do not modify the signature of methods declared here. 5. You can create additional supportive common methods in CommonEvents class.
/src/main/java/rest	AuthUtil.java	1. Class already defined to read and return bearer token from

		config.properties file.
/src/main/java/rest	CustomResponse	We have already created custom response object class. You need to use this and use it as return type from ApiUtil.java class methods.
/src/main/resources/	Config.xlsx	Data present to be used in Implementing functions.
/src/main/java/coreUtilities/utils	CommonEvents.java	1. Contains all common activities. 2. Certain templated common methods declared here. 3. You implement them as per your need. 4. You can add any additional method for common activity here
	Testng.xml	Execution needs to kick started from TestNG xml

PROBLEM STATEMENT

Need to automate the following activities using RestAssured.

Key Activities to implement:

Below activities need to be implemented in the ApiUtil.java file present in src/main/java/rest package.

Sl No.	Summary	Action	Expected Result
1	Create an Appointment with Authorization in Method: createAppointmentWithAuth(String endpoint, String body)	1.Create an URL by combining the BASE_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes as follows: **https://healthapp.yaksha.com/api/Appointment/AddAppointment** 2. The body required as the part of request is passed in requestBody variable in method argument. 3. Include a bearer token for authentication in authorization header. 4. Trigger a POST call. 5.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● AppointmentId (extracted from Response object) 6. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, AppointmentId and the complete Response object. - StatusCode should be 200. - Status should be OK. - AppointmentId should not be null.

2	Cancel an Appointment with Authorization in Method: cancelAppointmentWithAuth (String endpoint, Object body)	1. Call the PUT on endpoint i.e **https://healthapp.yaksha.com/api/ Appointment/AppointmentStatus?appointmentId=+ appointmentId+ "&status=cancelled"** 2. Include a bearer token for authentication in authorization header. 3. Trigger a PUT call. 4.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as a string (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK.
3	Search for a Patient with Authorization in Method: searchPatientWithAuth(String endpoint, Object body)	1. Call the GET on endpoint i.e **https://healthapp.yaksha.com/api/ /Patient/SearchRegisteredPatient?search=Test** 2. Include a bearer token for authentication in authorization header. 3.Trigger a GET call to the specified endpoint. 4.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as List<Map<String, Object>> (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK.

4	Retrieve a List of Appointments for a Specified Performer in Method: bookingListWithAuthInRange(String endpoint, Object body)	1. Call the GET on endpoint i.e **https://healthapp.yaksha.com/api//Appointment/Appointments?FromDate=" + dateFiveDaysBeforeStr + "&ToDate=" + currentDateStr + "&performerId=" + performerId + "&status=new"** 2. Include a bearer token for authentication in authorization header. 3.Trigger a GET call to the specified endpoint. 4.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as List<Map<String, Object>> (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK.
5	Retrieve Main Store Details in Method: MainStoreDetailsWithAuth(String endpoint, Object body)	1. Call the GET on endpoint i.e **https://healthapp.yaksha.com/api/PharmacySettings/MainStore** 2. Include a bearer token for authentication in authorization header. 3.Trigger a GET call to the specified endpoint. 4.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as Map<String, Object> (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK. - Name, storeDesc and StoreId should not be null.

6	Retrieve a List of Pharmacy Stores in Method: PharmacyStoresWithAuth(String endpoint, Object body)	1. Call the GET on endpoint i.e **https://healthapp.yaksha.com/api/Dispensary/PharmacyStores** 2. Include a bearer token for authentication in authorization header. 3.Trigger a GET call to the specified endpoint. 4.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as List<Map<String, Object>> (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK. - StoreId, and Name should not be null.
7	Activate a Pharmacy Counter Using Details in Method: ActivatePharmCount(String endpoint, Object body)	1. Call the PUT on endpoint i.e **https://healthapp.yaksha.com/api/Security/ActivatePharmacyCounter?counterId=" + counterId + "&counterName=" + counterName** 2. Include a bearer token for authentication in authorization header. 3.Trigger a PUT call. 4.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as Map<String, Object> (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK. - CounterName, and CounterId should not be null.

8	Deactivate a Pharmacy Counter in Method: DeactivatePharmCount(String endpoint, Object body)	1. Call the PUT on endpoint i.e **https://healthapp.yaksha.com/api/Security/DeactivatePharmacyCounter** 2. Include a bearer token for authentication in authorization header. 3.Trigger a PUT call. 4.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as Map<String, Object> (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK.
9	Retrieve Appointment Applicable Departments in Method: AppointApplicDept(String endpoint, Object body)	1. Call the GET on endpoint i.e **https://healthapp.yaksha.com/api/Master/AppointmentApplicableDepartments** 2. Include a bearer token for authentication in authorization header. 3.Trigger a GET call to the specified endpoint. 4.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as List<Map<String, Object>> (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK. - DepartmentId, and DepartmentName should not be null.

10	Retrieve Admitted Patients Data in Method: admittedPatientData(String endpoint, Object body)	1. Call the GET on endpoint i.e **https://healthapp.yaksha.com/api//Admission/AdmittedPatientsData?admissionStatus=admitted** 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call to the specified endpoint. 4. Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as List<Map<String, Object>> (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK. - PatientId, and AdmittedDate should not be null.
11	Retrieve Profile Details of Admitted Patients in Method: getProfileDetails(String endpoint, Object body)	1. Call the GET on endpoint i.e **https://healthapp.yaksha.com/api/Employee/Profile?empId=" + expectedEmployeeId** 2. Include a bearer token for authentication in authorization header. 3.Trigger a GET call to the specified endpoint. 4.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as Map<String, Object> (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK. - EmployeeId should not be null.

12	Add a New Department in Method: addDepartment(String endpoint, Object body)	1. Call the POST on the endpoint i.e., **https://healthapp.yaksha.com/api/Settings/Department** 2. The body required as the part of request is passed in requestBody variable in method argument. 3. Include a bearer token for authentication in authorization header. 4. Trigger a POST call. 5.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as Map<String, Object> (extracted from Response object) 6. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK. - DepartmentId should not be null.
13	Retrieve the List of Departments in Method: getDepartmentsList(String endpoint, Object body)	1. Call the GET on endpoint i.e **https://healthapp.yaksha.com/api/Settings/Department** 2. Include a bearer token for authentication in authorization header. 3.Trigger a GET call to the specified endpoint. 4.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as List<Map<String, Object>> (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK. - DepartmentId and DepartmentName should not be null.

14	Edit Department Details in Method: editDepartmentDetails(String endpoint, Map<String, String> body)	1. Call the PUT on endpoint i.e **https://healthapp.yaksha.com/api/Settings/Department** 2. Include a bearer token for authentication in authorization header. 3.Trigger a PUT call. 4.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as Map<String, Object> (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK. - DepartmentCode and DepartmentName should not be null.
15	Retrieve Imaging Data Response in Method: getImagingDataResponse(String endpoint, Object body)	1. Call the GET on endpoint i.e **https://healthapp.yaksha.com/api/RadiologySettings/ImagingTypes** 2. Include a bearer token for authentication in authorization header. 3.Trigger a GET call to the specified endpoint. 4.Create an object of type CustomResponse and initialize it with values extracted from the Response object: ● Response ● statusCode (extracted from Response object) ● Status (extracted from Response object) ● Results as List<Map<String, Object>> (extracted from Response object) 5. Return the CustomResponse object from the method.	- Returns an object of type CustomResponse containing statusCode, status, Results and the complete Response object. - StatusCode should be 200. - Status should be OK. - ImagingTypeId and ImagingTypeName should not be null.

NOTE: "Please do not delete any file in the src folder. But you are free to add any other file".

Expectations:

1) **Learners should write automation scripts using Java and REST Assured to automate the API testing for all the provided methods (e.g., GET, POST, PUT, DELETE).** In other words, the automation script should perform all mentioned API interactions, including validation of responses.
2) **Learners should not use any pre-built libraries or tools to validate API responses (e.g., JSON schema validation tools).** They should manually validate the response content (e.g., status codes, response body, etc.) by writing their own logic for assertion.

IMPLEMENTATION/FUNCTIONAL REQUIREMENT

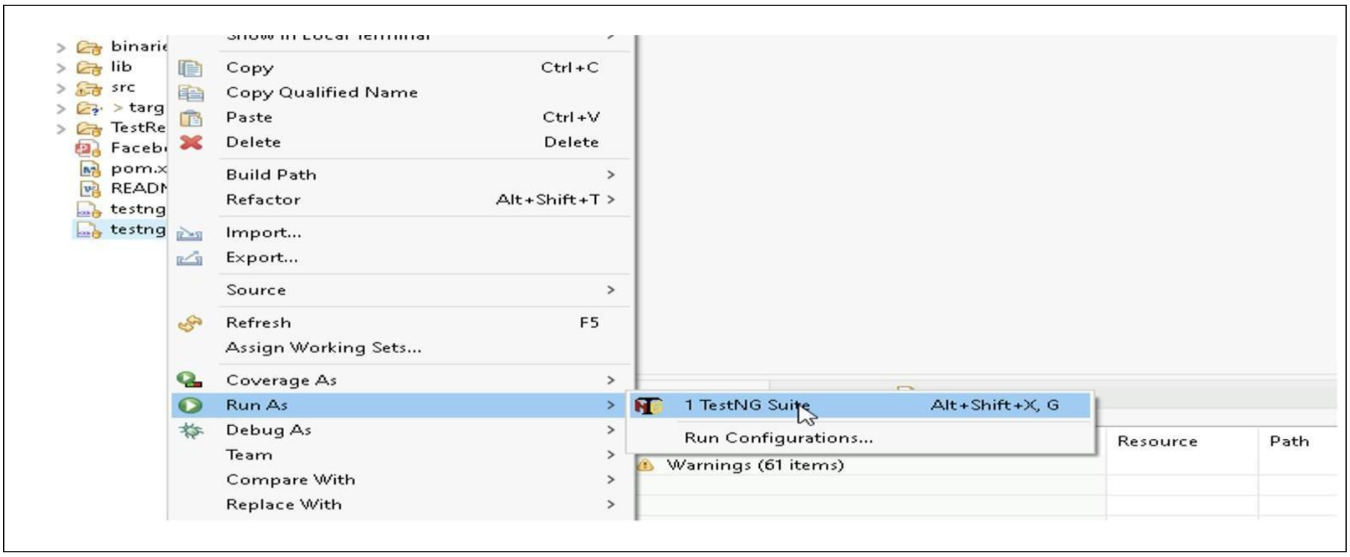
1.1 Code Quality/Optimizations

1. Associates should have written clean code that is readable.
2. Associates need to follow SOLID programming principles.

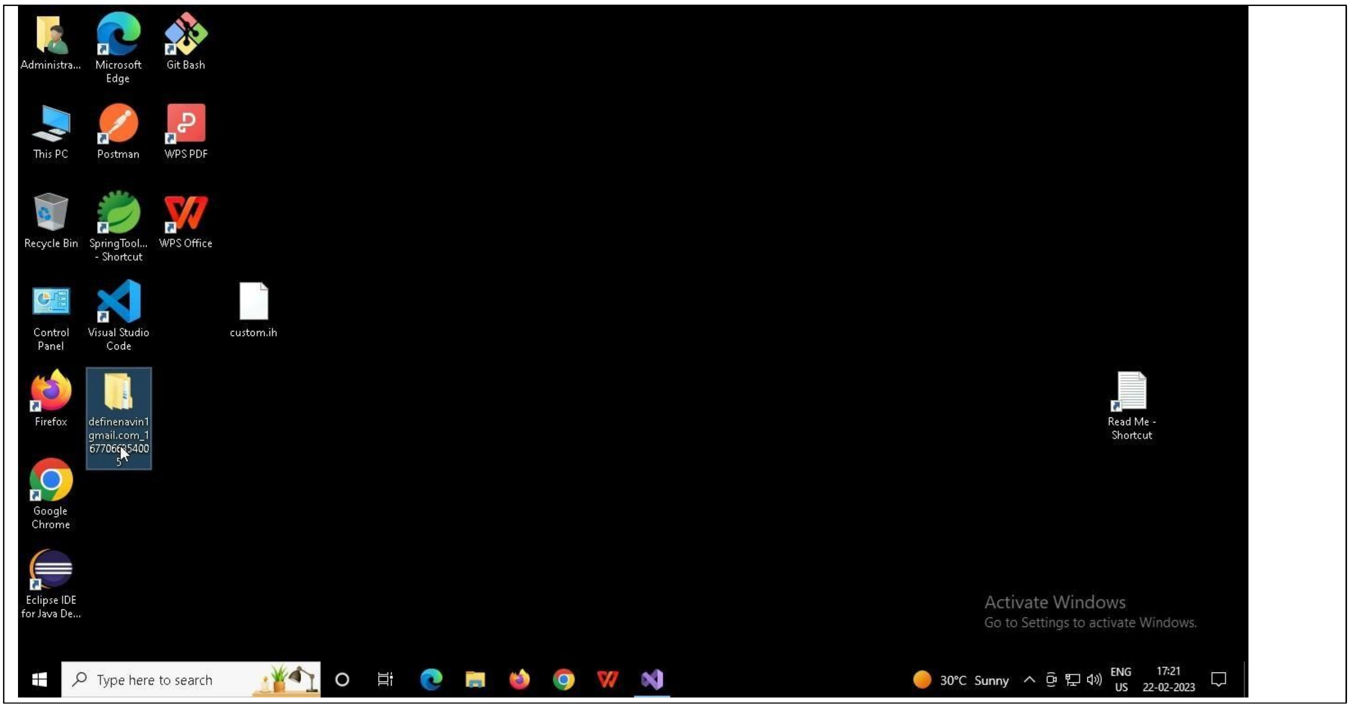
Execution Steps to Follow

1. **You are mandatorily required to run test cases for applications before final submission. Without which project evaluation will not happen.**

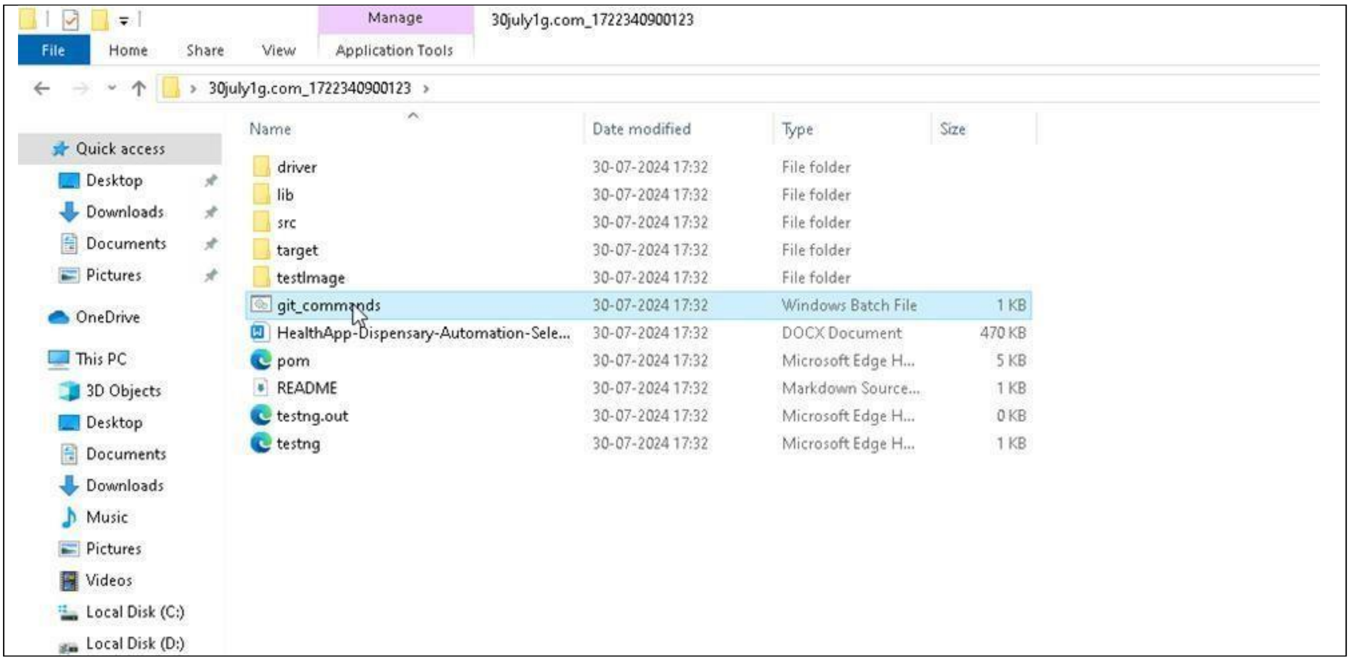
2. **You can launch test cases any time as follows: Right click on testng.xml and run TestNGSuite**



3. **Before final submission, you are also required to push your code to GIT. Following are the steps to follow:**

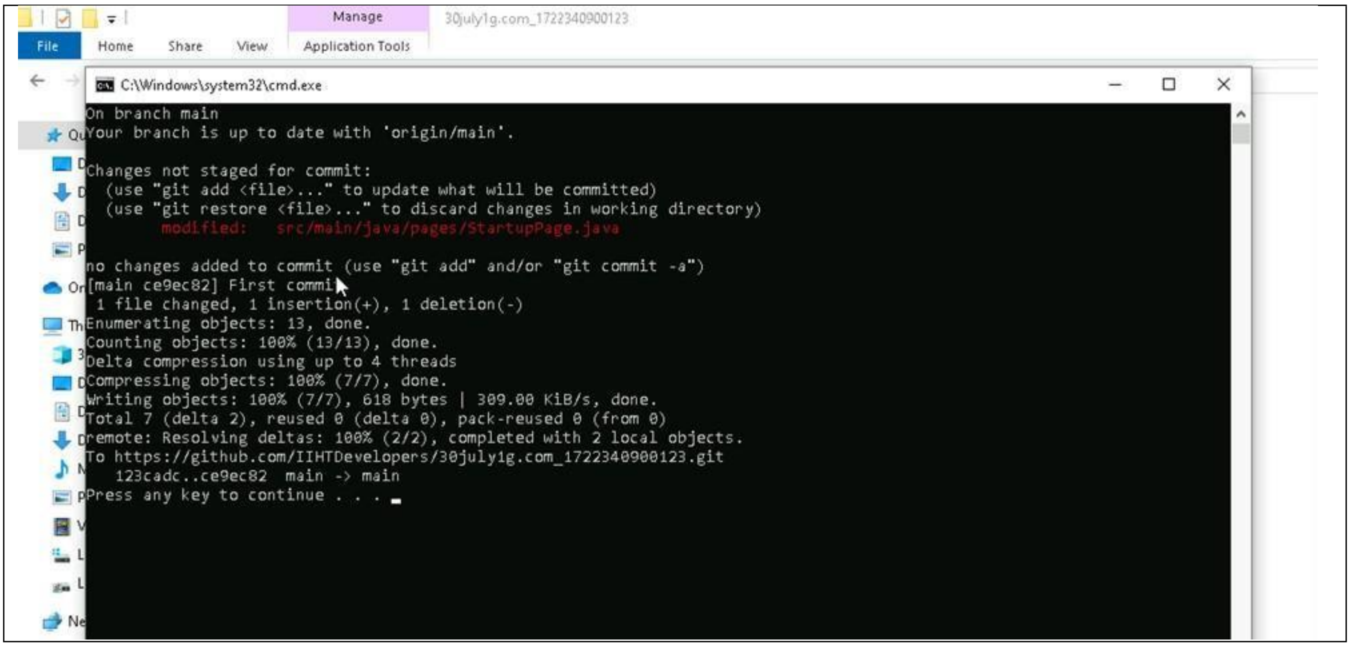


In your project folder, you will find a batch file named git_commands



Name	Date modified	Type	Size
driver	30-07-2024 17:32	File folder	
lib	30-07-2024 17:32	File folder	
src	30-07-2024 17:32	File folder	
target	30-07-2024 17:32	File folder	
testImage	30-07-2024 17:32	File folder	
git_commands	30-07-2024 17:32	Windows Batch File	1 KB
HealthApp-Dispensary-Automation-Sele...	30-07-2024 17:32	DOCX Document	470 KB
pom	30-07-2024 17:32	Microsoft Edge H...	5 KB
README	30-07-2024 17:32	Markdown Source...	1 KB
testng.out	30-07-2024 17:32	Microsoft Edge H...	0 KB
testng	30-07-2024 17:32	Microsoft Edge H...	1 KB

Double-click the batch file to run it. It will run the commands to push your code to GIT.



```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/main/java/pages/StartupPage.java

no changes added to commit (use "git add" and/or "git commit -a")
[main ce9ec82] First commit
 1 file changed, 1 insertion(+), 1 deletion(-)
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 618 bytes | 309.00 KiB/s, done.
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/IIHTDevelopers/30july1g.com_1722340900123.git
   123cadc..ce9ec82  main -> main
Press any key to continue . . .
```

==

All the Best