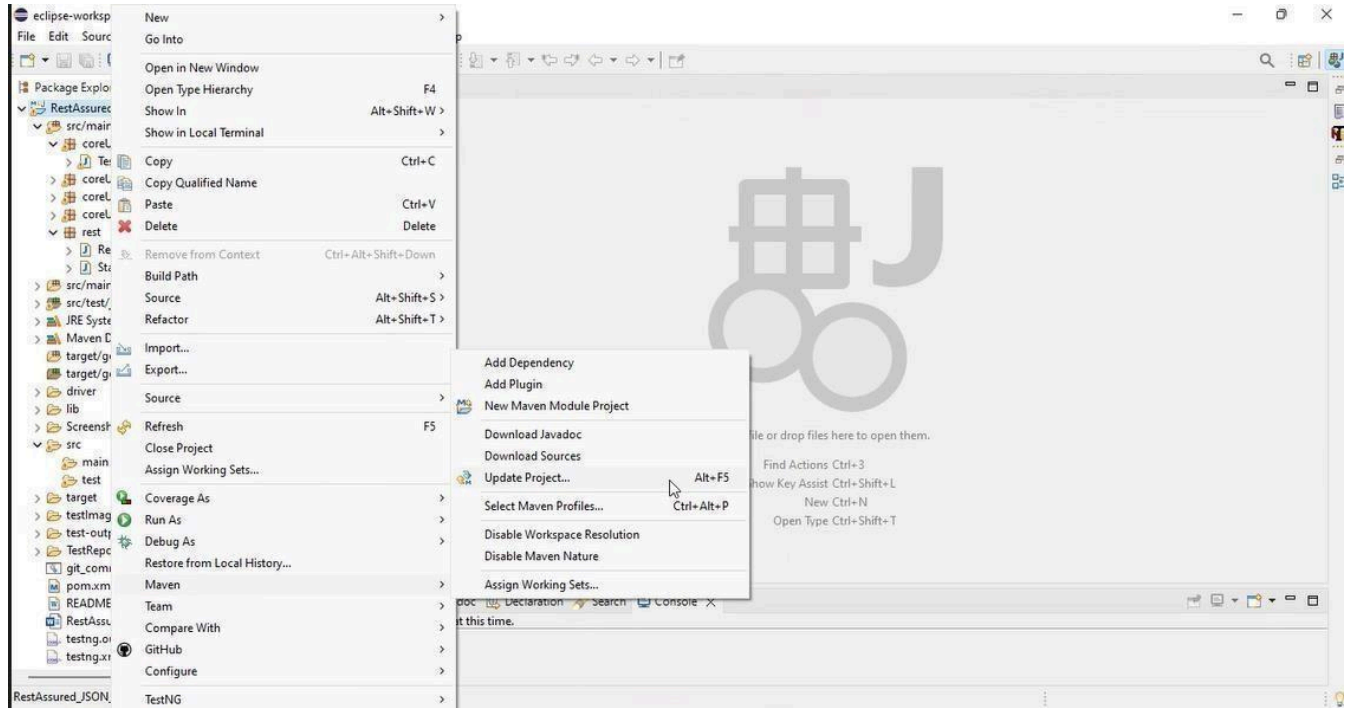


REST ASSURED API AUTOMATION PROJECT

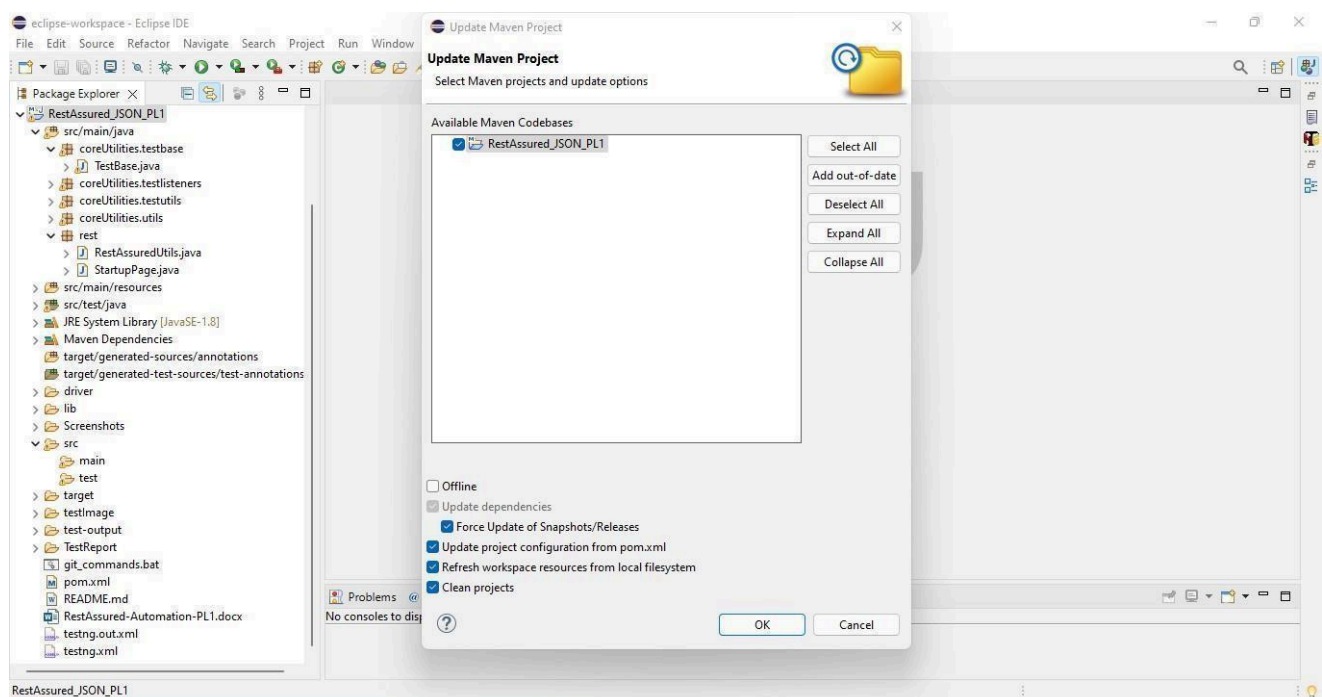
Pre-requisite:

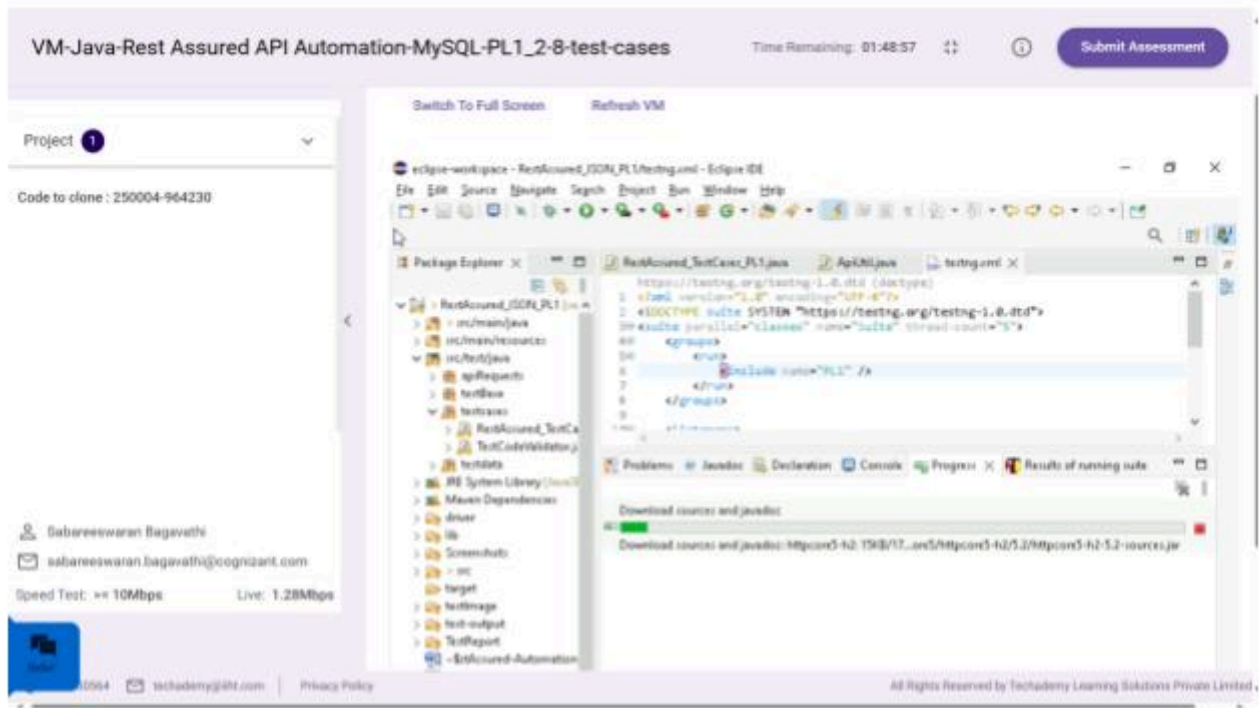
As soon as you import a project in eclipse, update the project using the maven update option as below. This is to resolve issue if any maven dependency not downloaded properly:

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





Note: Once all the project dependencies are downloaded you may find certain documentation jar files getting downloaded which may continue for a longer time. This is the known behaviour. This will not block your application development activity. You can ignore it and continue working on your application.

Template Code Structure:

- Below are the packages and files you will be required to work upon.
- Other Files and packages you can ignore.
- In other Files and packages do not do any changes. It would affect your evaluation.
- 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/rest	ApiUtil.java	<ol style="list-style-type: none"> 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	<ol style="list-style-type: none"> 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/utis	CommonEvents.java	<ol style="list-style-type: none"> 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.

SI No.	Summary	Action	Expected Result
1	Retrieve list of stocks in Method: getAllStocks(String endpoint, Object body)	<ol style="list-style-type: none">1. Create an URL by combining the BASE_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes: https://healthapp.yaksha.com/api/PharmacyStock/AllStockDetails.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:<ul style="list-style-type: none">• Response• statusCode (extracted from Response object)• Status (extracted from Response object)• ItemId list (extracted from Response object)• ItemName list (extracted from Response object)• GenericName list (extracted from Response object)5. Return the CustomResponse object from the method.	<ul style="list-style-type: none">- Returns an object of type CustomResponse containing statusCode, status, ItemId list, ItemName list, GenericName list and the complete Response object.- StatusCode should be 200.- Status should be OK.- ItemId, ItemName and GenericName should not be null and should not be empty.

2	<p>Retrieve main store details in Method:</p> <p><code>getMainStore(String endpoint, Object body)</code></p>	<ol style="list-style-type: none"> 1. Create an URL by combining the BASE_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes: <code>https://healthapp.yaksha.com/api/PharmacySettings/MainStore</code> 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: <ul style="list-style-type: none"> • Response • statusCode (extracted from Response object) • Status (extracted from Response object) • StoreId object (extracted from Response object) • Category object (extracted from Response object) • IsActive object (extracted from Response object) 5. Return the CustomResponse object from the method. 	<ul style="list-style-type: none"> - Returns an object of type CustomResponse containing statusCode, status, StoreId, Category, IsActive, and the complete Response object. - StatusCode should be 200. - Status should be OK. - StoreId, Category and IsActive should not be null.
3	<p>Get Requisition List by Date Range in Method:</p> <p><code>getRequisitionByDateRange(String endpoint, Object body)</code></p>	<ol style="list-style-type: none"> 1. Create an URL by combining the BASE_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes: <code>https://healthapp.yaksha.com/api/DispensaryRequisition/Dispensary/1?FromDate=" + fromDate + "&ToDate=" + toDate</code> 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: <ul style="list-style-type: none"> • Response • statusCode (extracted from Response object) • RequisitionNo list (extracted from Response object) 	<ul style="list-style-type: none"> - Returns an object of type CustomResponse containing statusCode, status, RequisitionNo list, RequisitionStatus list, RequisitionId list and the complete Response object - StatusCode should be 200. - Status should be OK. - RequisitionNo, RequisitionStatus and RequisitionId lists should not be empty. - RequisitionNo, RequisitionStatus and RequisitionId lists should have non-null values.

		<ul style="list-style-type: none"> • RequisitionStatus list (extracted from Response object) • RequisitionId list (extracted from Response object) • Status (extracted from Response object) <p>5. Return the CustomResponse object from the method.</p>	
4	<p>Get Patient Consumptions List in Method:</p> <p><code>getPatientConsumptions(String endpoint, Object body)</code></p>	<ol style="list-style-type: none"> 1. Create an URL by combining the BASE_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes: <code>https://healthapp.yaksha.com/api/PatientConsumption/PatientConsumptions</code> 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: <ul style="list-style-type: none"> • Response • statusCode (extracted from Response object) • PatientId list (extracted from Response object) • HospitalNo list (extracted from Response object) • PatientVisitId list (extracted from Response object) • Status (extracted from Response object) 5. Return the CustomResponse object from the method. 	<p>- Returns an object of type CustomResponse containing statusCode, status, PatientId list, HospitalNo list, PatientVisitId list and the complete Response object</p> <p>- StatusCode should be 200.</p> <p>- Status should be OK.</p> <p>- PatientId, HospitalNo and PatientVisitId lists should not be empty.</p> <p>PatientId, HospitalNo and PatientVisitId lists should have non-null values.</p>

5	<p>Retrieve patient consumption info in Method:</p> <p><i>getPatientConsumptionInfoByPatientIdAndVisitId(String endpoint, Object body)</i></p>	<ol style="list-style-type: none"> 1. Create an URL by combining the BASE_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes: https://healthapp.yaksha.com/api/PatientConsumption/PatientConsumptionInfo?PatientId=114&patientVisitId=53" 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: <ul style="list-style-type: none"> • Response • statusCode (extracted from Response object) • PatientName object (extracted from Response object) • HospitalNo object (extracted from Response object) • StoreId object (extracted from Response object) • Status (extracted from Response object) 5. Return the CustomResponse object from the method. 	<ul style="list-style-type: none"> - Returns an object of type CustomResponse containing statusCode, status, PatientName, HospitalNo, StoreId, and the complete Response object. - StatusCode should be 200. - Status should be OK. - PatientName object, HospitalNo object and StoreId object should not be null.
6	<p>Retrieve Billing Scheme by Scheme ID in Method:</p> <p><i>getBillingSchemeBySchemeId(String endpoint, Object body)</i></p>	<ol style="list-style-type: none"> 1. Create an URL by combining the BASE_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes: https://healthapp.yaksha.com/api/PatientConsumption/Pharmacy/pBillingScheme?schemeld=" + schemeld 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: <ul style="list-style-type: none"> • Response • statusCode (extracted from Response object) 	<ul style="list-style-type: none"> - Returns an object of type CustomResponse containing statusCode, status, SchemeCode object, SchemeName object, Schemeld object, and the complete Response object. - StatusCode should be 200. - Status should be OK. - SchemeCode object, SchemeName object and Schemeld object should not be null.

		<ul style="list-style-type: none"> • SchemeCode object (extracted from Response object) • SchemeName object (extracted from Response object) • Schemeld object (extracted from Response object) • Status (extracted from Response object) <p>5. Return the CustomResponse object from the method.</p>	
7	<p>Retrieve Billing Summary by Patient ID in Method:</p> <p><code>getBillingSummaryByPatientId(String endpoint, Object body)</code></p>	<ol style="list-style-type: none"> 1. Create an URL by combining the BASE_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes: <code>https://healthapp.yaksha.com/api/PharmacySales/PatientBillingSummary?patientId=" + patientId</code> 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: <ul style="list-style-type: none"> • Response • statusCode (extracted from Response object) • PatientId object (extracted from Response object) • TotalDue object (extracted from Response object) • Status (extracted from Response object) 5. Return the CustomResponse object from the method. 	<p>- Returns an object of type CustomResponse containing statusCode, status, PatientId object, TotalDue object and the complete Response object.</p> <p>- StatusCode should be 200.</p> <p>- Status should be OK.</p> <p>- PatientId object, TotalDue object should not be null.</p>
8	<p>Retrieve Consumptions List of a Patient By ID in Method:</p> <p><code>getConsumptionsListOfAPatientById(String endpoint, Object body)</code></p>	<ol style="list-style-type: none"> 1. Create an URL by combining the BASE_URL (already declared) and path parameter (provided as an argument in method). The final URL becomes: <code>https://healthapp.yaksha.com/api/PatientConsumption/ConsumptionsOfPatient?patientId=" + patientId + "&patientVisitId=" + patientVisitId</code> 2. Include a bearer token for authentication in authorization header. 3. Trigger a GET call to the specified 	<p>- Returns an object of type CustomResponse containing statusCode, status, PatientConsumptionId list, ConsumptionReceiptNo list, TotalAmount list, and the complete Response object.</p> <p>- StatusCode should be 200.</p> <p>- Status should be OK.</p> <p>- PatientConsumptionId list,</p>

		<p>endpoint.</p> <p>4. Create an object of type CustomResponse and initialize it with values extracted from the Response object:</p> <ul style="list-style-type: none"> • Response • statusCode (extracted from Response object) • PatientConsumptionId list (extracted from Response object as list) • ConsumptionReceiptNo list (extracted from Response object as list) • TotalAmount list (extracted from Response object as list) • Status (extracted from Response object) <p>5. Return the CustomResponse object from the method.</p>	<p>ConsumptionReceiptNo list and TotalAmount list should not be null and should not be empty.</p>
--	--	--	---

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.

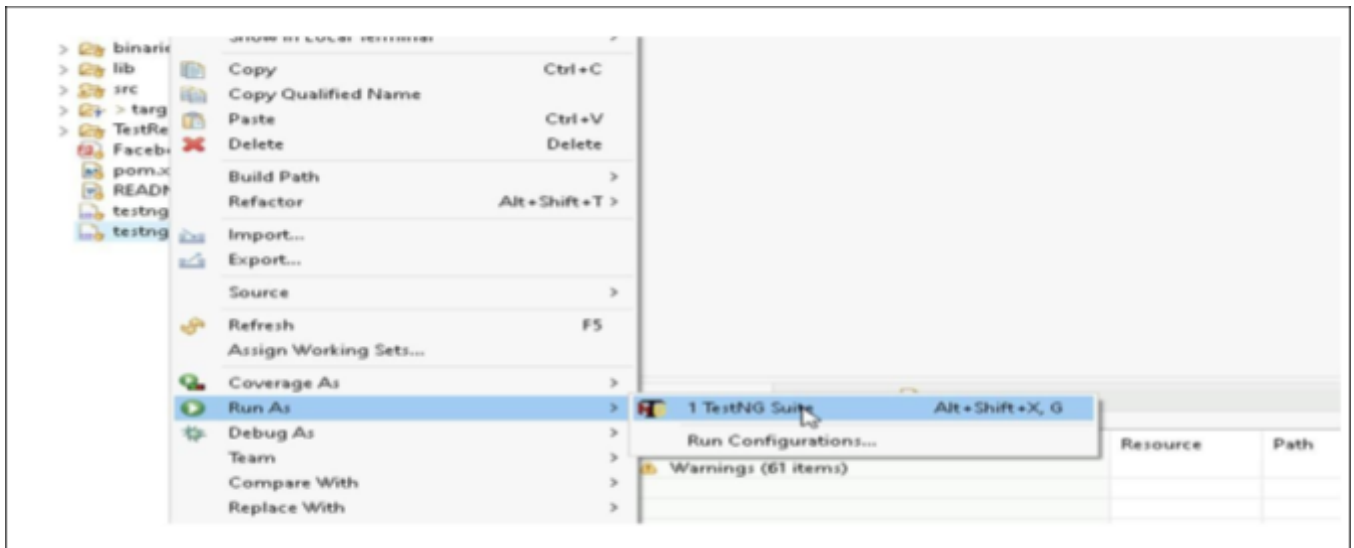
IMPLEMENTATION/FUNCTIONAL REQUIREMENT

1.1 CODE QUALITY/OPTIMIZATIONS

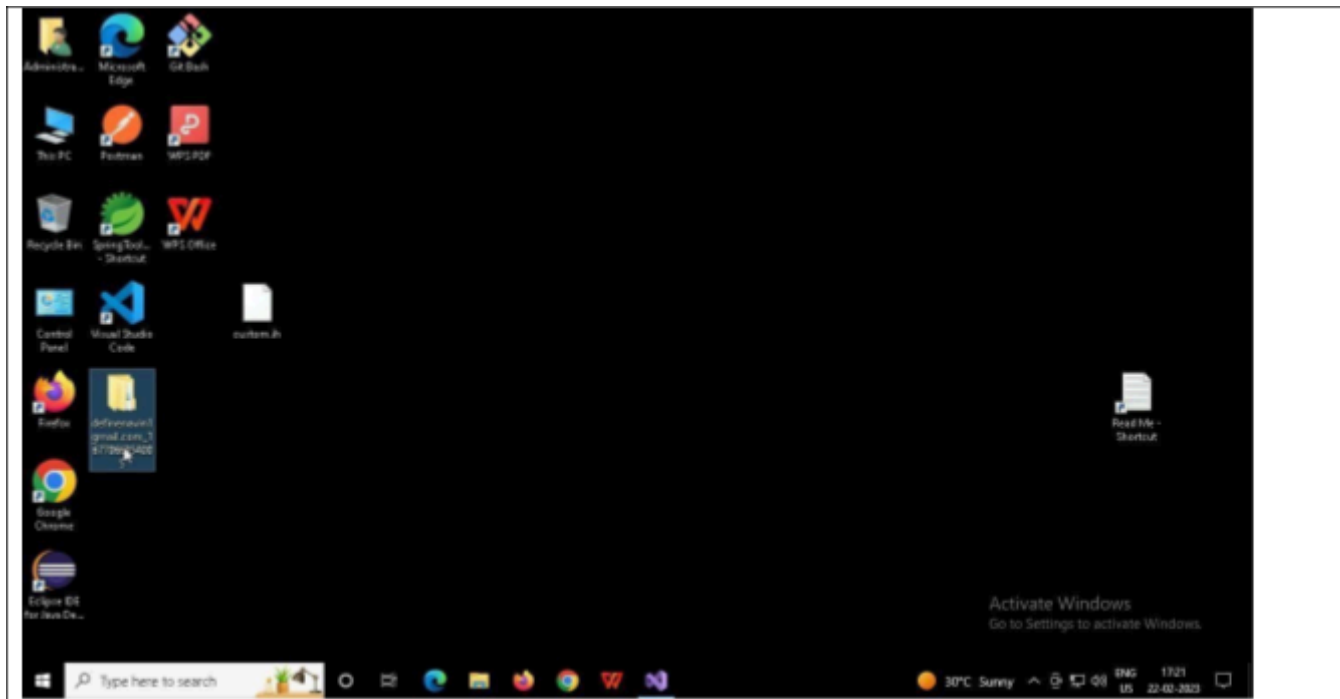
1. Associates should have written clean code that is readable.
2. Associates need to follow SOLID programming principles.

EXECUTION STEPS TO FOLLOW

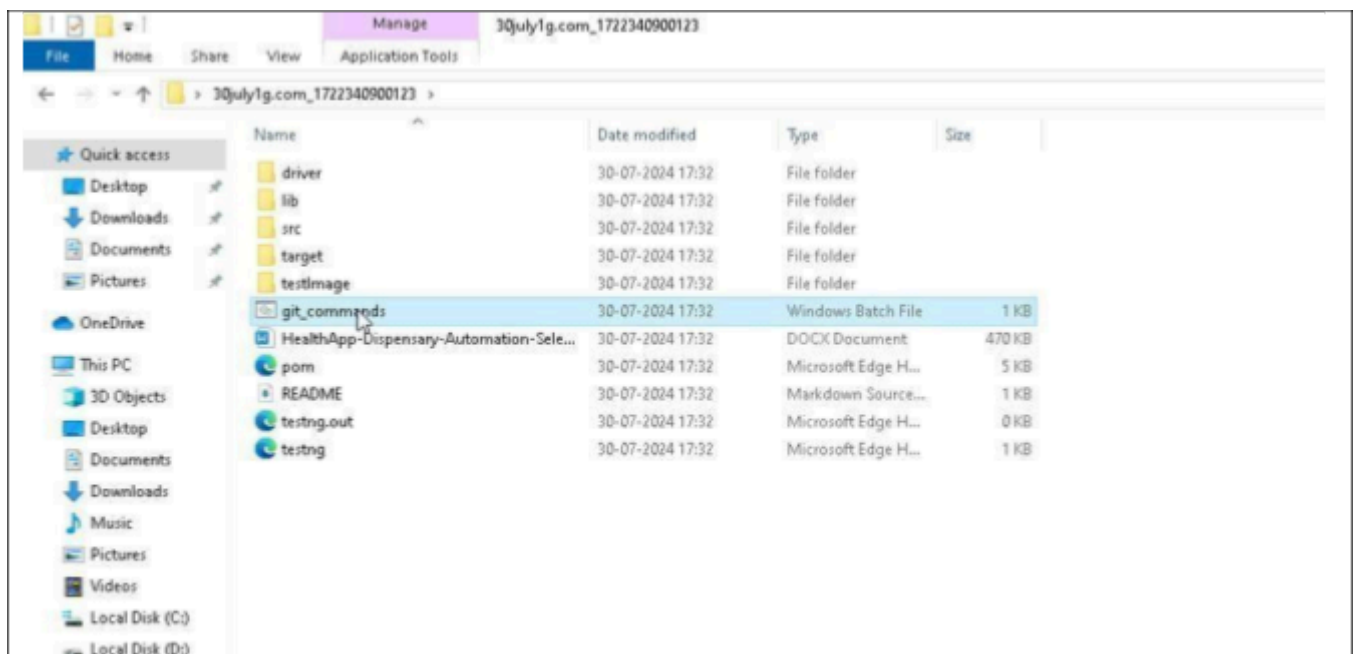
1. **You are mandatorily required to run test cases for applications before final submission. Without which project evaluation will not happen.**
2. **You can launch test cases any time as follows: Right click on testng.xml and run TestNGSuite**



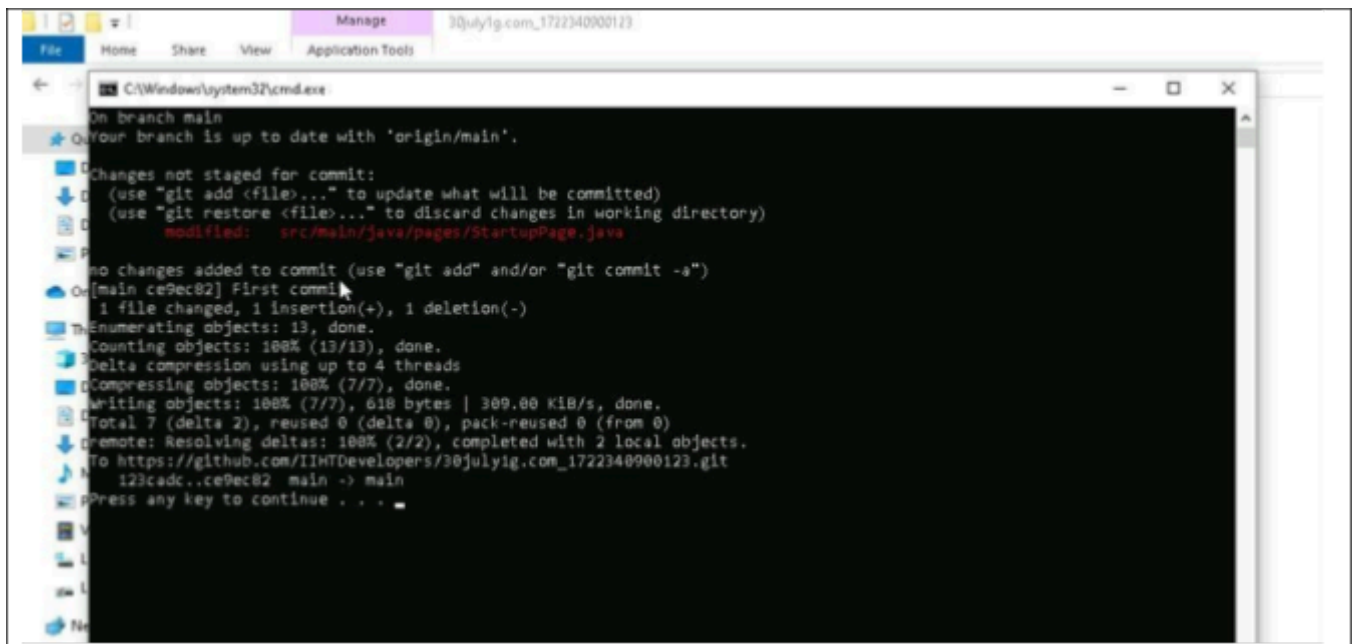
3. **Before final submission, you are also required to push your code to GIT. Following are the steps to follow:**



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



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



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window shows the output of a Git commit and push operation. The text is as follows:

```
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 -s")
[master 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/30julyig.com_1722340900123.git
   123cadc..ce9ec82  main -> main
Press any key to continue . . .
```

=====

All the Best