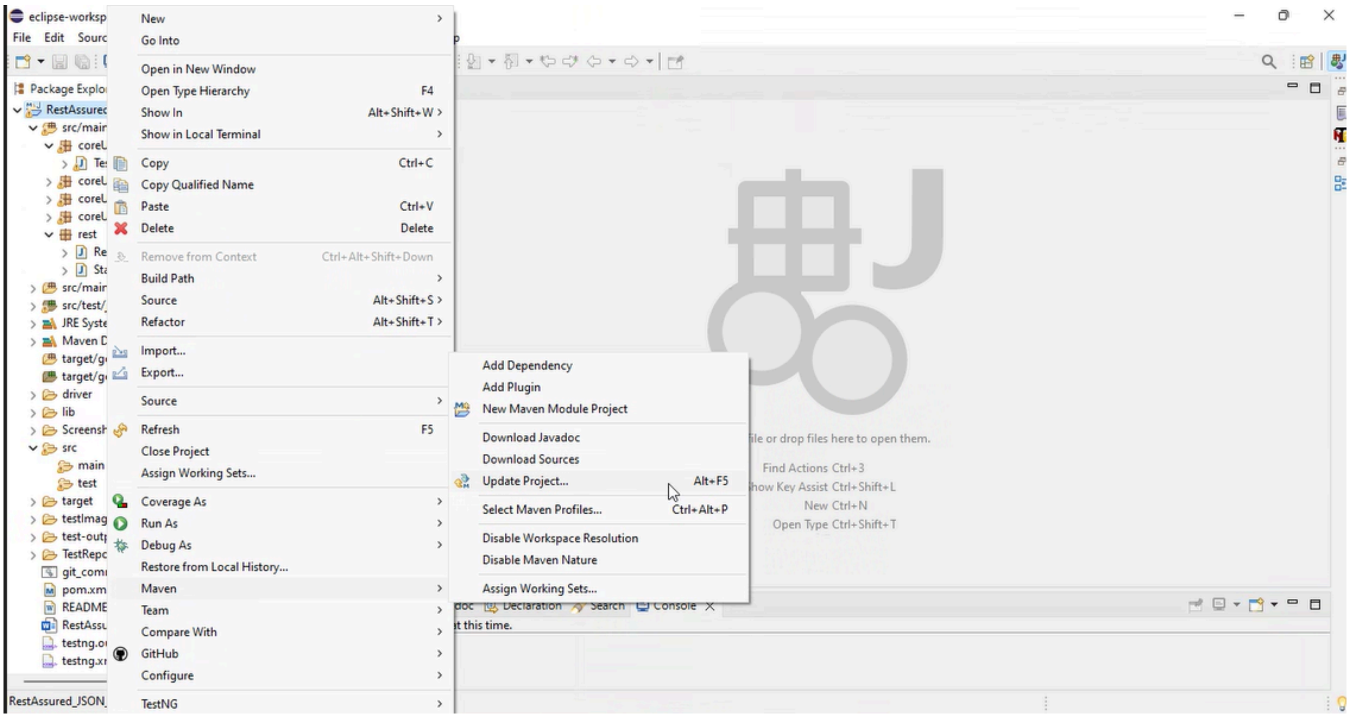
RestAssured API Automation Project

Login A

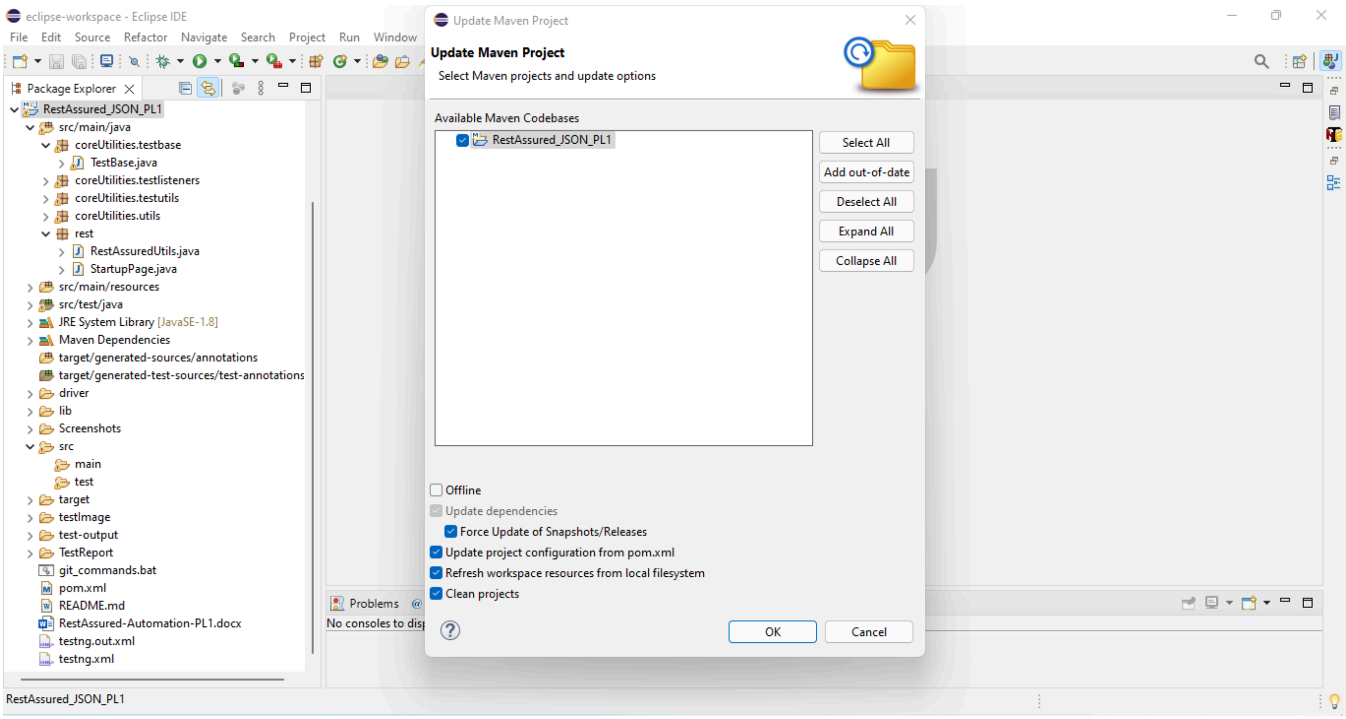
Pre-requisite:

As soon as you import project in eclipse, update the project using 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

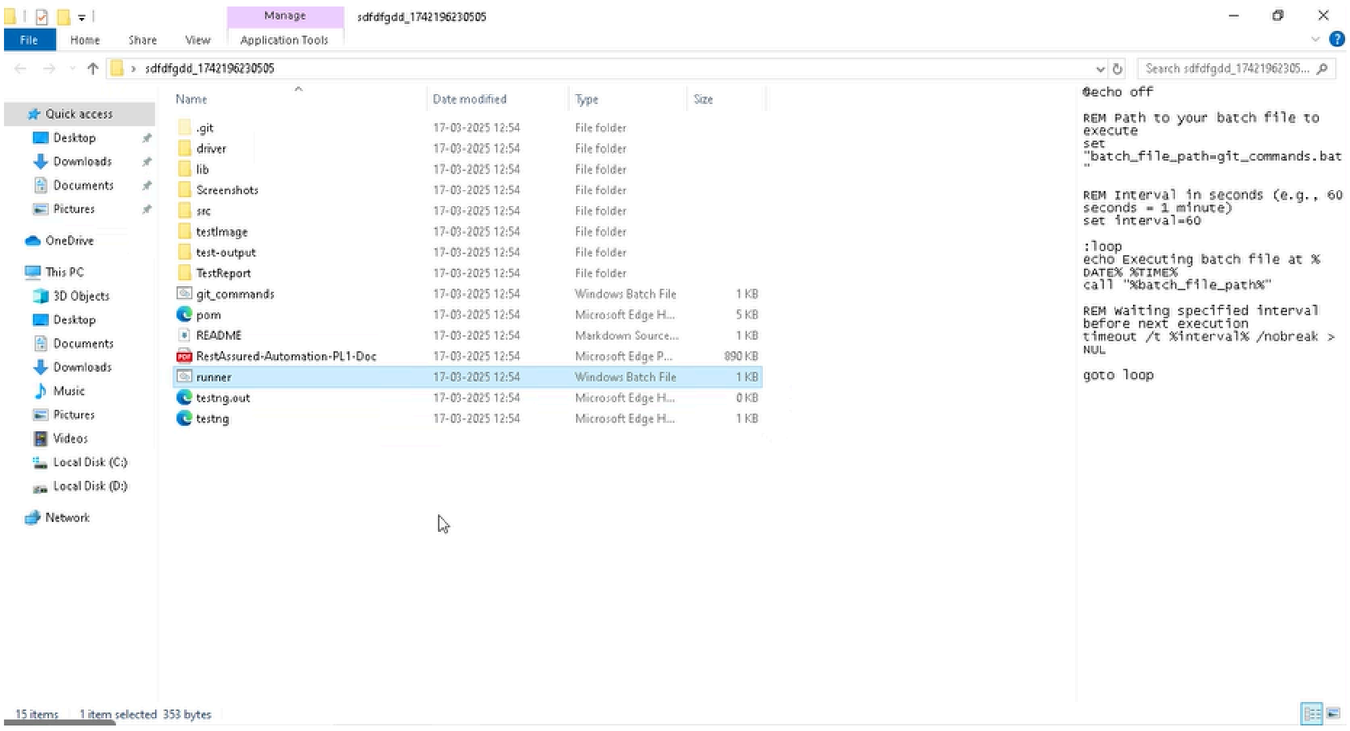
Available Maven Codebases

- [✓] 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

Login A

** Before starting any implementation. Please execute this “runner” file present in the cloned folder. This will keep pushing the changes at regular intervals.

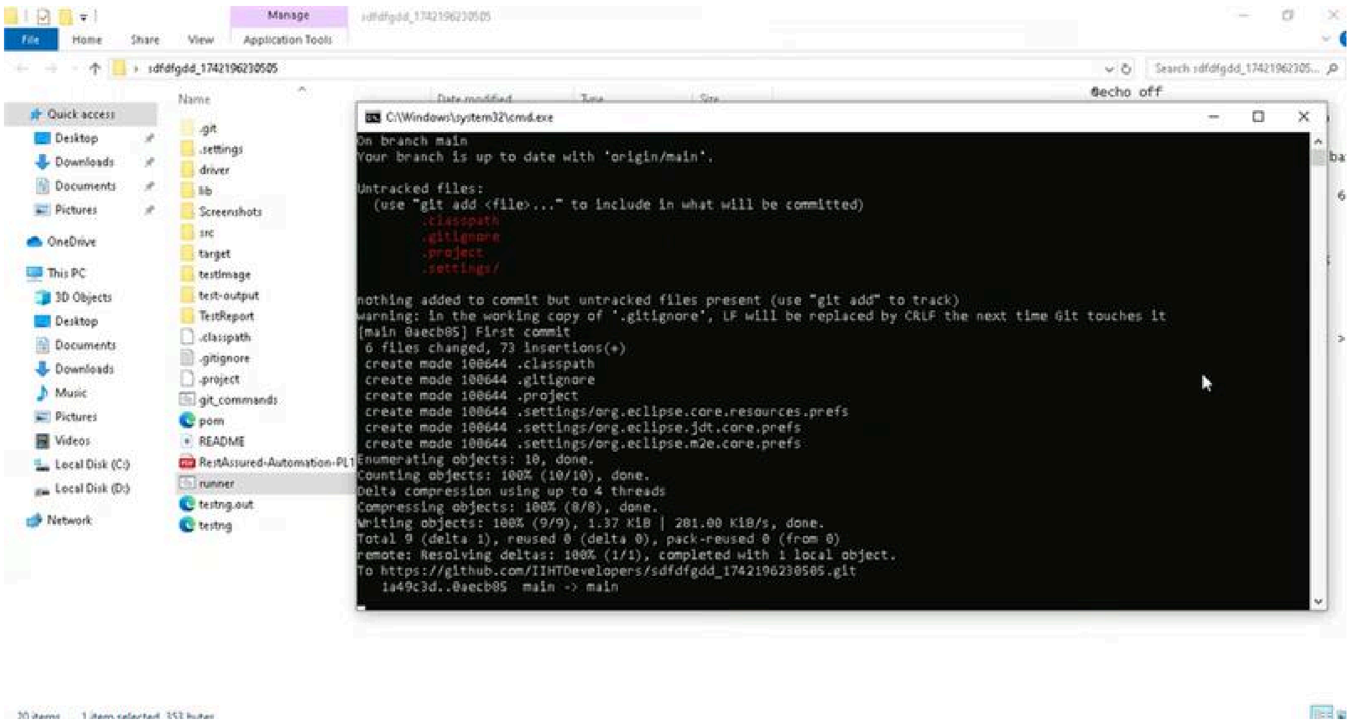


sdfdfgdd_1742196230505

Name	Date modified	Type	Size
.git	17-03-2025 12:54	File folder	
driver	17-03-2025 12:54	File folder	
lib	17-03-2025 12:54	File folder	
Screenshots	17-03-2025 12:54	File folder	
src	17-03-2025 12:54	File folder	
testImage	17-03-2025 12:54	File folder	
test-output	17-03-2025 12:54	File folder	
TestReport	17-03-2025 12:54	File folder	
git_commands	17-03-2025 12:54	Windows Batch File	1 KB
pom	17-03-2025 12:54	Microsoft Edge H...	5 KB
README	17-03-2025 12:54	Markdown Source...	1 KB
RestAssured-Automation-PL1-Doc	17-03-2025 12:54	Microsoft Edge P...	890 KB
runner	17-03-2025 12:54	Windows Batch File	1 KB
testng.out	17-03-2025 12:54	Microsoft Edge H...	0 KB
testng	17-03-2025 12:54	Microsoft Edge H...	1 KB

```
@echo off

REM Path to your batch file to execute
set "batch_file_path=git_commands.bat"

REM Interval in seconds (e.g., 60 seconds = 1 minute)
set interval=60

:loop
echo Executing batch file at %DATE% %TIME%
call "%batch_file_path%"

REM Waiting specified interval before next execution
timeout /t %interval% /nobreak > NUL

goto loop
```

15 items   1 item selected 353 bytes



```
C:\Windows\system32\cmd.exe
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .classpath
        .gitignore
        .project
        .settings/

nothing added to commit but untracked files present (use "git add" to track)
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
[main 0aecb05] First commit
 6 files changed, 73 insertions(+)
 create mode 100644 .classpath
 create mode 100644 .gitignore
 create mode 100644 .project
 create mode 100644 .settings/org.eclipse.core.resources.prefs
 create mode 100644 .settings/org.eclipse.jdt.core.prefs
 create mode 100644 .settings/org.eclipse.m2e.core.prefs
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 1.37 KiB | 281.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/IIHTDevelopers/sdfdfgdd_1742196230505.git
   1a49c3d..0aecb05  main -> main
```

Login A

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	RestAssuredUtils.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/resources/	Config.xlsx	Data present to be used in creating a new Post.

/src/main/java/coreUtilities/utils	CommonEvents.java	1. Contains all common activities. 2. Certain templated common method 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:

Sl No.	**Summary**	**Action**	**Expected Result**
1	Retrieve Comments for a Post and validate the response in method: getCommentsForPost(int postId)	1. Call the GET **https://jsonplaceholder.typicode.com/posts/{postId}/comments** endpoint 2. Pass the postId as a parameter 3. Check if the response body contains an array of comments 4. Validate each comment has a postId, id, name, email, and body properties	Returns a 200 OK status with a list of comments related to the specified post. Response body contains an array of comments with the specified properties
2	Delete a Post and validate the deletion in method: deletePost(int id)	1. Call the DELETE **https://jsonplaceholder.typicode.com/posts/{id}** endpoint. 2. Pass the id of the post to delete. 3. Call GET /posts/{id} to verify the post no longer exists.	Returns a 200 OK status code if the post is successfully deleted. The post should not be found in the subsequent GET request, returning 404 Not Found.
3	Update an existing post and validate the update in method: updatePost(int id, String newTitle, String newBody, int userId)	1. Call the PUT **https://jsonplaceholder.typicode.com/posts/{id}** endpoint. 2. Provide newTitle, newBody, and userId in the request body. 3. Validate the response contains the updated title, body, and userId fields.	Returns a 200 OK status code with the updated post. Response body contains the updated title, body, userId, and id.

4	Create a new post and validate the creation in method: addPost(Map<String, String> data)	1. Call the POST **https://jsonplaceholder.typicode.com/posts** endpoint. 2. Provide title, body, and userId in the request body fetched from passed data arguement (data to be read from the config.xlsx file). 3. Validate the response contains the created post	Returns a 201 Created status code. Response body contains the created title, body, userId, and auto-generated id.
5	Retrieve a post by ID and validate the response in method: getPostById(int id)	1. Call the GET **https://jsonplaceholder.typicode.com/posts/{id}** endpoint. 2. Pass the id of the post to retrieve. 3. Validate the response body contains the correct post details.	Returns a 200 OK status code. Response body contains the correct title, body, userId, and id.

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.

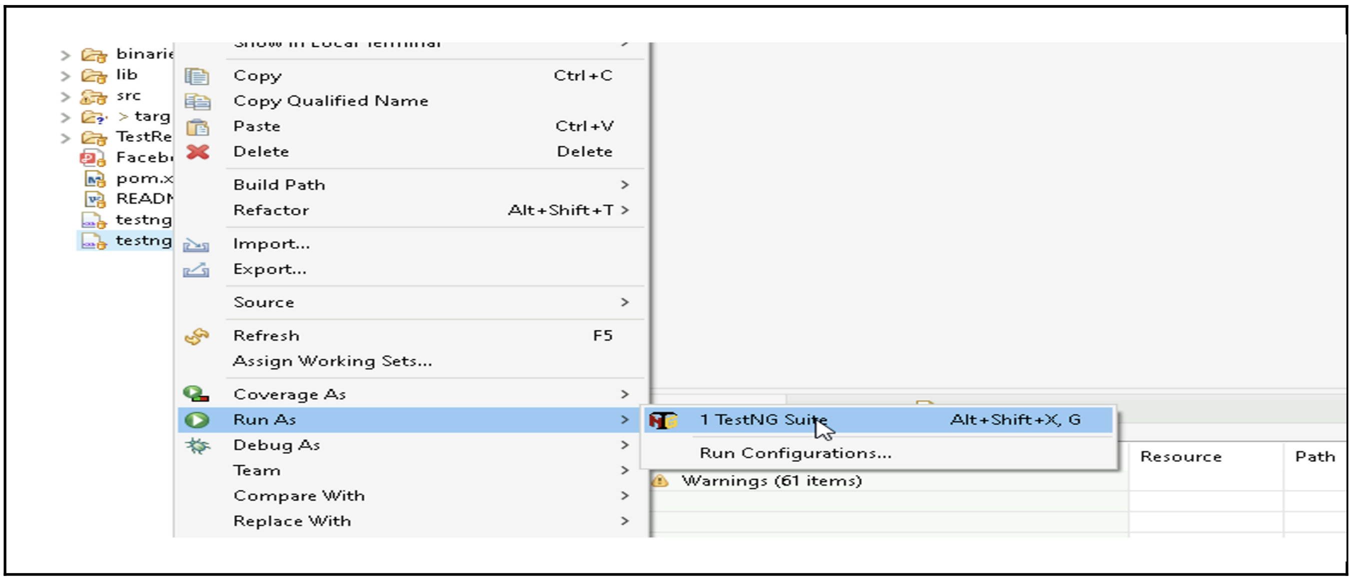
IMPLEMENTATION/FUNCTIONAL REQUIREMENT

1.1 CODE QUALITY/OPTIMIZATIONS

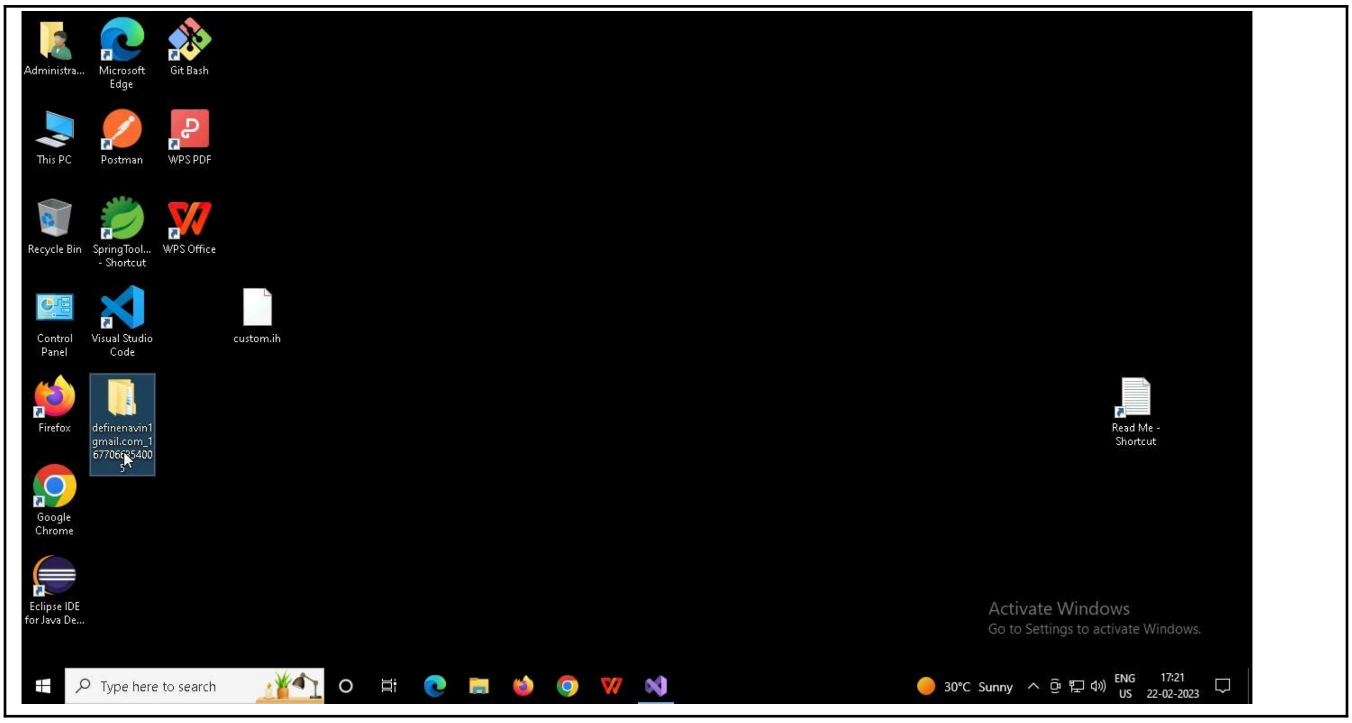
1. Associates should have written clean code that is readable.
2. Associates need to follow SOLID programming principles.

EXECUTION STEPS TO FOLLOW

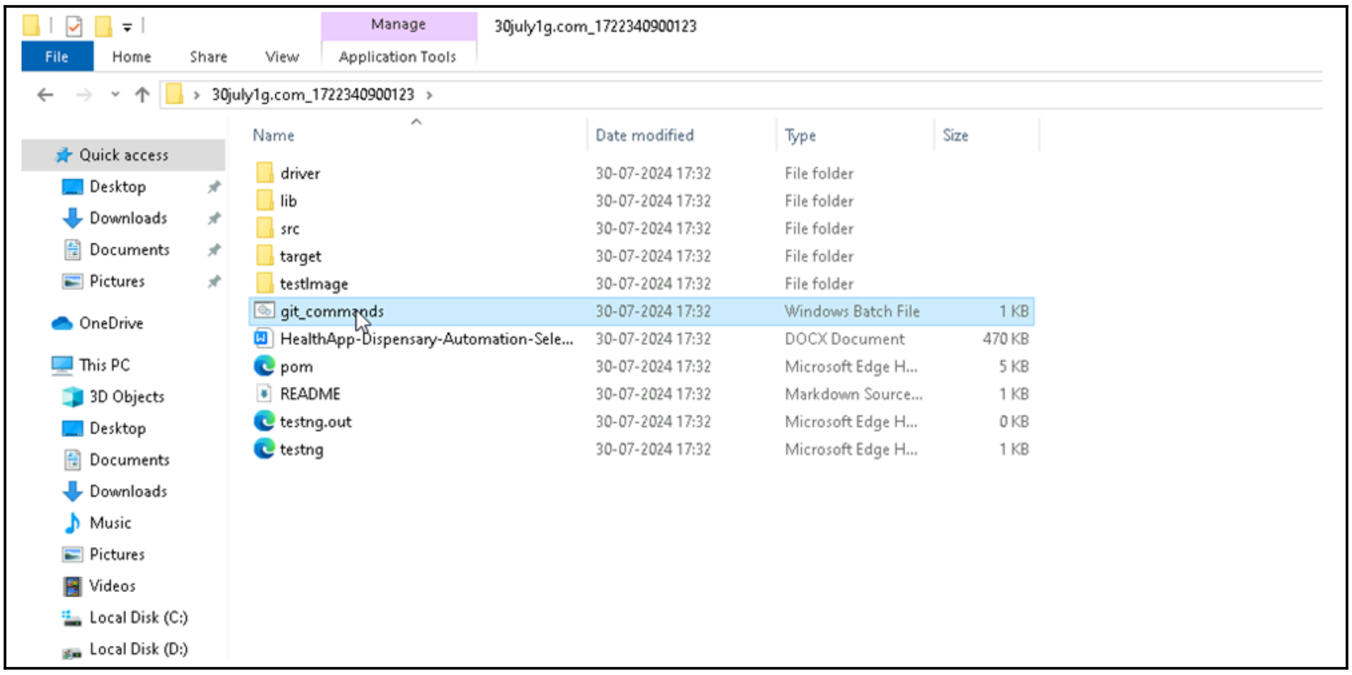
1. **You are mandatory required to run test cases for applications before final submission. Without which project evaluation will not happen.**
2. **You can launch test cases any time as follows: Right click on testng.xml and run TestNGSuite**



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

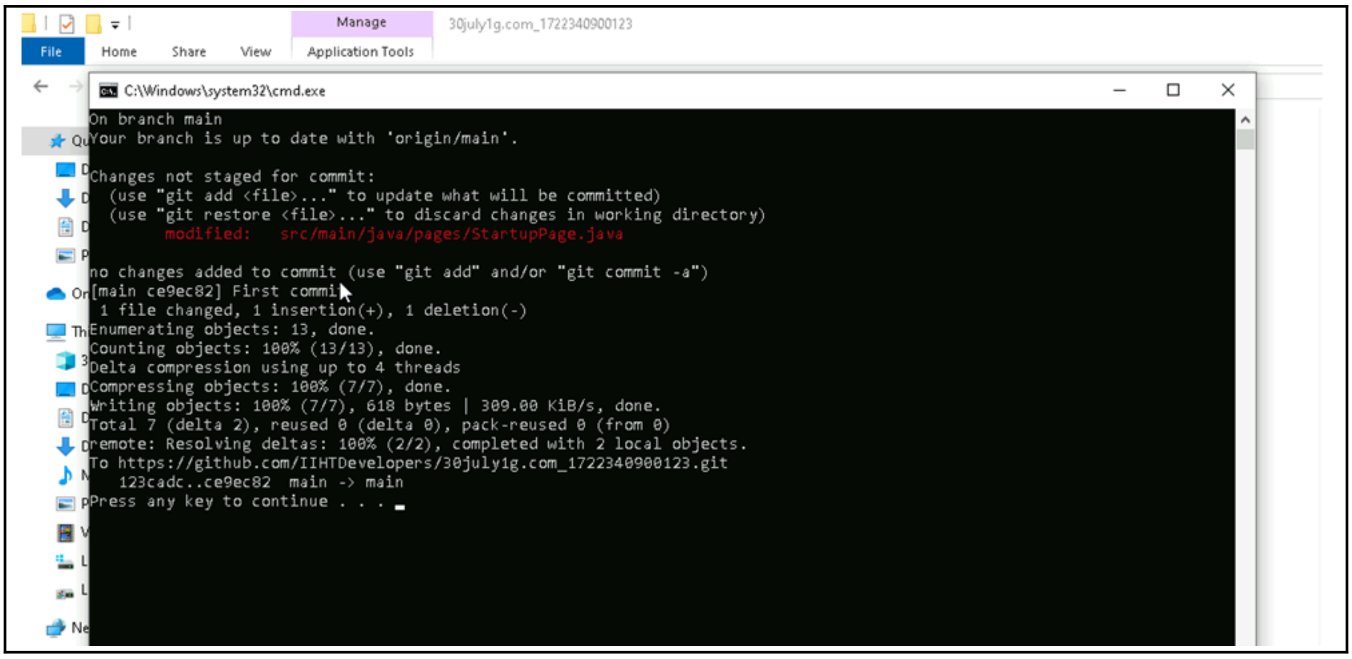


In your project folder, you will find a batch file named git_commands



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 commi
 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