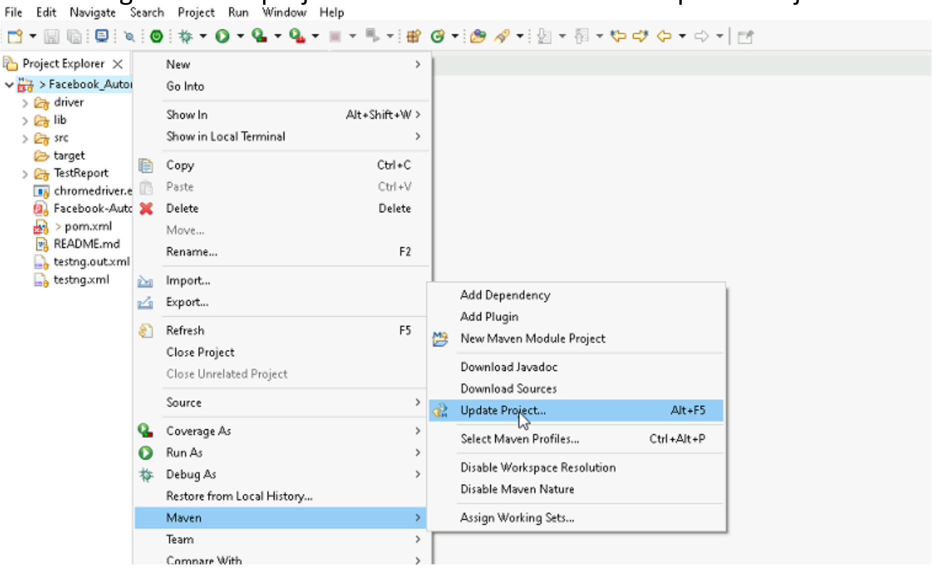
HealthApp Automation Appointment Module-PL2

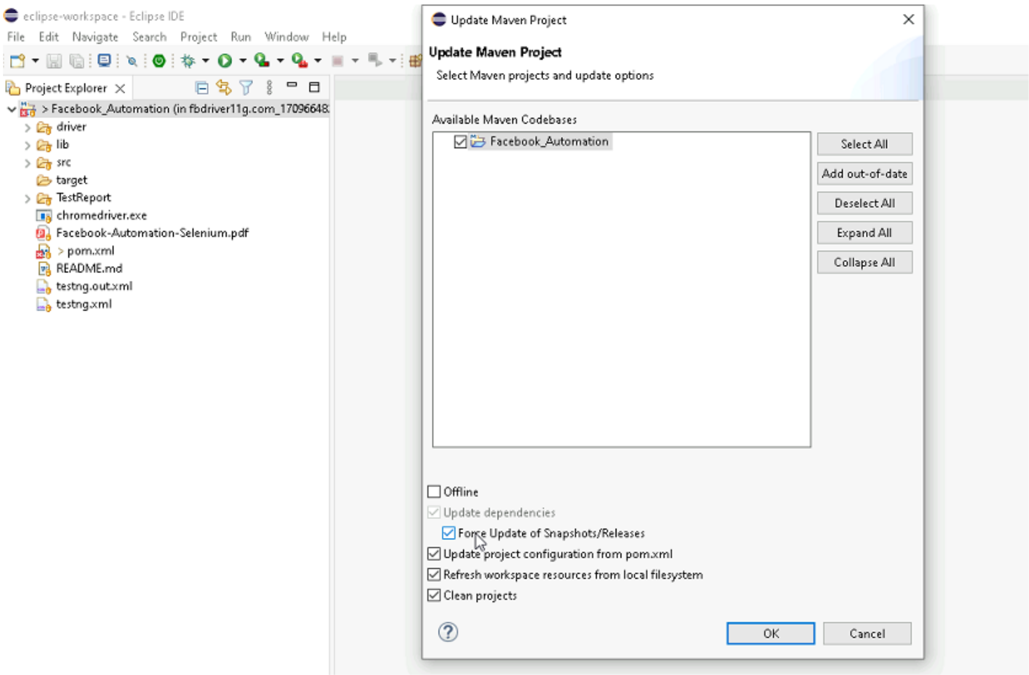
Pre-requisite:

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

☑ Facebook_Automation

□ 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 even URL to navigate to, is read using these methods.**
/src/main/java/pages	appointment_Pages.java	1. All core activities (mentioned in list above) to be performed here. 2. The comments associated with each templated method here describe the expectation. 3. You can define locators and xpath here. 4. Declare any variable/object you need to share data/status between different methods. 5. Do not modify the signature of methods declared here. 6. You can create additional supportive common methods in CommonEvents class.
/src/main/resources/	Config.xlsx	URL to navigate to. Already URL is

		defined here
	expected_data.xlsx	Contains data to fill in form
/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 Selenium + Java.

Key Activities to implement:

Sl No.	Summary	Action	Expected Result
1	* Navigate to the URL. * Retrieve Title and URL of the current page. * Verify Title & URL: Check if the title & URL matches the expected title.	1. Go to URL : https://healthapp.yaksha.com/ 2. login as valid credential (username : **admin** , password : **pass123**) and click on "Sign in" 3. get the title and URL of the Home page, post login 4. validate the title and URL of the Home page	Title should be : DanpheHealth URL should be : https://healthapp.yaksha.com/Home/Index#/
2	Confirm the presence of the Appointment module and ensure the "Select Counter" popup is correctly triggered and labeled when attempting to access the module.	**Preconditions**: User must be logged into the health system. **Steps**: 1. **Confirm Module Presence**: Check the left navigation of the health system interface to verify if the Appointment module is listed. 2. **Navigate to Appointment Module**: Click on the expand icon next to the Appointment module. 3. **Observe Popup Behavior**: Monitor the system response upon trying to access the Appointment module. 4. **Verify Popup Presence**: Determine if the "Select Counter" popup appears as expected. 5. **Check Popup Page Name**: Validate that the popup is correctly titled "Select Counter".	Appointment module should be present While trying to navigate to the Appointment Module, Select Counter popup should come and popup page name should be "Select Counter"

3	To ensure that the "New Patient" button is present on the "New Visit" page and that clicking this button reveals the "Patient Information" text.	**Preconditions**: The user must be logged into the health system. The user is on the "Select Counter" popup within the Appointment module. **Steps**: 1. **Navigate from "Select Counter" Popup**: Click on the "New 1" link within the "Select Counter" popup. 2. **Verify "New Patient" Button Presence**: Once on the "New Visit" page, check for the presence of the "New Patient" button. 3. **Click on "New Patient" Button**: If the "New Patient" button is verified to be present, click on it. This step is expected to trigger additional UI elements related to new patient information. 4. **Verify "Patient Information" Text Presence**: After clicking the "New Patient" button, verify that the "Patient Information" text appears on the page.	Navigation to the "New Visit" page should be seamless from the "Select Counter" popup. The "New Patient" button should be clearly visible and functional on the "New Visit" page. Upon clicking the "New Patient" button, the "Patient Information" text should appear, confirming that the system is ready for new patient data entry.
4	To ensure that the "Care of Person Contact" textbox can be interacted with and highlighted effectively on the "New Visit" page.	**Preconditions**: The user is logged into the health system . The user is on the "New Visit" page, ready to input or modify patient information. **Steps**: 1. **Presence on New Visit Page**: Confirm that the user is already on the "New Visit" page. 2. **Scroll to the Bottom of the Page**: Navigate to the bottom of the "New Visit" page where the "Care of Person Contact" textbox is typically located. 3. **Click on "Care of Person Contact" Textbox**: Interact with the "Care of Person Contact" textbox by clicking on it. 4. **Highlight Textbox**: Apply a visual highlight to the textbox by changing its background color to yellow.	User should able to scroll to the bottom of the "**New Visit**" page. "Care of Person Contact" textbox should be clickable and highlighted.
5	To ensure that the "Care of Person" textbox on the New Visit page of the Appointment module contains the correct placeholder text.	**Preconditions**: The user is logged into the health system. The user is on the New Visit page within the Appointment module, specifically positioned at the bottom of the page. **Steps**: 1. **Locate and Interact with Textbox**: Identify and click on the "Care of Person" textbox located at the bottom of the New Visit page. 2. **Retrieve Placeholder Text**: Capture the placeholder text from the "Care of Person" textbox. Placeholder text is typically a grayed-out text within the textbox. 3. **Verify Placeholder Accuracy**: Check that the retrieved placeholder text matches the expected value, "Care Taker Person".	**"Care of Person"** textbox should be clickable and placeholder name should be **"Care Taker Person"**

6	To confirm that an appropriate error message is displayed when attempting to print an invoice without filling in details the fields in the 'Patient Information' form on the 'New Visit' page.	**Preconditions**: The user must be logged into the health system. The user is on the 'New Visit' page within the Appointment module. **Steps**: 1. **Attempt to Print Invoice Without Data Entry**: Click on the "Print Invoice" button without entering any information in the 'Patient Information' form. 2. **Popup Message Interaction**: Upon clicking "Print Invoice", a confirmation popup appears with the message "Confirm! Are you sure you want to Print Invoice?". Click on the "Confirm" button to proceed. 3. **Observe Error Message**: After confirming the action, check for an error message under the 'Last Name' textfield. 4. **Validate Error Message**: Confirm that the error message specifically states "Last Name is required."	Clicking on "Print Invoice" without filling out the 'Last Name' should trigger a validation error. The error message, "Last Name is required," should display clearly under the 'Last Name' textfield.
7	To ensure that all required fields in the 'Patient Information' form on the 'New Visit' page are filled out correctly and that the data entered is accurately reflected in the system.	**Preconditions**: The user must be logged into the health system. The user is on the 'New Visit' page within the Appointment module, specifically in the 'Patient Information' form section. **Steps**: 1. **Enter Data in First Name Textbox**: Input a valid first name into the 'First Name' textbox. 2. **Enter Data in Middle Name Textbox**: Fill in the middle name if applicable in the 'Middle Name' textbox. 3. **Enter Data in Last Name Textbox**: Input a valid last name in the 'Last Name' textbox. 4. **Enter Data in Age Textbox**: Type the patient's age in the 'Age' textbox. 5. **Enter Data in Phone No. Textbox**: Provide a valid phone number in the 'Phone No.' textbox. 6. **Validate Entered Data**: After entering all the required data, Validate it.	Given text fields value inside the "**Patient Information"** form are filled with appropriate information. The entered data should be correctly displayed in First Name, Middle Name, Last Name, Age and Contact Number Field on "Patient Information" form of New visit page
8	To ensure that the 'Have DOB ?' checkbox functions correctly by toggling its selection and triggering the display of the Datepicker field on the 'Patient Information' form.	**Preconditions**: The user must be logged into the health system. The user is on the 'New Visit' page within the 'Patient Information' form. **Steps**: 1. **Toggle 'Have DOB ?' Checkbox**: Click on the 'Have DOB ?' checkbox. 2. **Verify Checkbox Selection**: Check to confirm that the 'Have DOB ?' checkbox is indeed selected following the click. 3. **Observe Datepicker Field Activation**: After verifying the checkbox selection, observe whether a Datepicker field becomes visible or accessible. This field is expected to appear as part of the form’s dynamic response to the checkbox selection.	The 'Have DOB ?' checkbox should toggle to a selected state when clicked. Upon selection, the Datepicker field should become visible.

9	To validate that entering data into the "Referrer Name*" textbox in the "Add External Referral" form and clicking the "Add" button triggers a successful update notification.	**Preconditions**: The user must be logged into the health system. The user is on the 'New Visit' page within the 'Visit Information' form. The "External?" checkbox is pre-selected. **Steps**: 1. **Confirm Presence on Form**: Verify that the user is on the 'Visit Information' form of the 'New Visit' page and that the "External?" checkbox is selected. 2. **Initiate Referral Form**: Click on the "+" icon associated with adding external referrals. This action should open the "Add External Referral" form. 3. **Enter Referrer Name**: Input the referrer name into the "Referrer Name*" textbox. The data for this field should be retrieved from an Excel file, ensuring it meets any specified requirements (e.g., format, mandatory field). 4. **Submit Form**: Click the "Add" button to submit the referral information. Upon submission, a notification popup is expected to appear at the bottom of the page indicating the outcome of the action. 5. **Verify Success Notification**: Observe and confirm that the success notification message states "Success update". This message should be clearly visible and confirm the successful addition of the external referral.	After entering the data and clicking "Add", a success notification should pop up, stating "Success update".
10	To confirm that the option "Son" can be selected from the 'Relation With Patient' dropdown and verify that the selection is properly captured on the 'Patient Information' form.	**Preconditions**: The user must be logged into the health system. The user is on the 'Patient Information' form of the 'New Visit' page. **Steps**: 1. **Navigate to Relation Dropdown**: Scroll to locate the 'Relation With Patient' dropdown menu. 2. **Select 'Son' from Dropdown**: Click on the dropdown menu and select the option "Son". 3. **Verify Selection**: Confirm that "Son" is selected and properly displayed as the current choice in the dropdown menu. This verification ensures that the user's input is correctly captured and reflects in the system without any errors.	Upon selecting "Son", the dropdown should correctly show "Son" as the selected option, confirming that the system accurately records user selections.
11	Ensure the functionality of the 'Month' radio button within the 'Book Appointment' submodule, confirming that it can be selected and subsequently deselected by switching to another radio button.	**Preconditions:** The user must be logged into the health system. The user has navigated to the 'Appointment' module. **Steps:** 1. **Access Book Appointment:** Click on the "Book Appointment" submodule under the 'Appointment' module. **2.** **Initiate New Patient Booking:** Within the 'Book Appointment' interface, click on the "New Patient" button to begin the process of scheduling a new **appointment.** 3. **Select 'Month' Radio Button:** Find and select the 'Month' radio button.	The 'Month' radio button should be selectable and show as selected when clicked. Upon clicking "Days," the 'Month' radio button should be deselected.

		4. **Verify 'Month' Radio Button Selection:** Confirm that the 'Month' radio button is indeed selected. 5. **Deselect 'Month' Radio Button:** To test the toggle functionality, click on the "Days" radio button, which should deselect the 'Month' radio button. 6. **Validate Deselection:** Verify that selecting "Days" successfully deselects "Month".	
12	To demonstrate the use of JavaScript for entering data into textboxes and selecting a radio button on the 'Book Appointment' page within the Appointment module.	**Preconditions:** The user is logged into the health system and has navigated to the 'Book Appointment' page within the Appointment module. **Steps:** 1. **Confirm Page Access:** Verify that the user is on the 'Book Appointment' page. This ensures that all JavaScript commands will be executed in the correct context. 2. **Execute JavaScript to Fill Textboxes:** Use JavaScript to input values into the 'FirstName', 'Middle Name', and 'Last Name' textboxes. 3. **Click 'Male' Radio Button Using JavaScript:** Similarly, use JavaScript to select and click the 'Male' radio button.	The JavaScript commands should correctly populate the 'FirstName', 'Middle Name', and 'Last Name' textboxes with the specified values. The 'Male' radio button should be selected without requiring physical user interaction.
13	Ensure that the 'Add Appointment' button can be identified using the Name attribute locator, verify its presence, and visually confirm it on the 'Book Appointment' page within the Appointment module.	**Preconditions**: The user must be logged into the health system. The user is on the 'Book Appointment' page within the Appointment module. **Steps**: 1. **Confirm Page Access**: Verify that the user is on the correct 'Book Appointment' page to ensure accurate context for the test. 2. **Locate 'Add Appointment' Button Using Name Locator**: Locate the 'Add Appointment' button by searching for its Name attribute. 3. **Retrieve Button Attributes**: Examine the attributes of the 'Add Appointment' button to ensure it matches expected properties. 4. **Verify Button Presence**: Check that the button is indeed present on the page and that the attributes confirm it is the correct button as intended for use in the appointment scheduling process. 5. **Highlight the Button**:Apply a temporary visual enhancement, like changing the button's background color as yellow.	Attribute value of "Add Appointment" button should be "addappointment". "Add Appointment" button should be highlighted.

14	To confirm that clicking the 'Add Appointment' button without the required input fields completed results in an appropriate error notification, and to validate this error message rigorously using a hard assert approach.	**Preconditions:** The user must be logged into the health system. The user is on the 'Book Appointment' submodule within the Appointment module. **Steps:** 1. **Click on 'Add Appointment' Button:** Proceed to click the 'Add Appointment' button. **2. Observe Error Notification Popup:** After clicking the button, an error notification should appear. **3. Validate Error Message Using Hard Assert:** The content of the error message should be critically validated. A hard assert ensures that the test passes only if the error message exactly matches the expected text.	Error message should be : **Failed** **Failed!! Cannot create appointment. Check the Details Correctly.**
15	To ensure that the Sub-Category name corresponding to a specific code (Code: 0003) can be accurately retrieved and validated within the 'Sub Category' Tab of the Procurement Module's settings page.	**Preconditions**: The user must be logged into the health system. The user has navigated to the Procurement module. **Steps:** **1. Access Procurement Module:** Navigate to the 'Procurement' module from the main interface of the health system. **2. Navigate to Settings Page:** Within the Procurement module, go to the settings page. **3. Select 'Sub Category' Tab:** Click on the 'Sub Category' tab within the settings page. 4. **Retrieve Sub-Category Name:** Identify and fetch the name of the sub-category associated with the given code '0003'. **5. Validate Sub-Category Name:** Confirm that the retrieved sub-category name is correct for the given code.	user should be able to fetch and validate the data from Specified table.

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 script using Java and selenium to automate all the steps in the above question. In other words, automation script should perform all mentioned steps.
2) Learners should not use any tools to create the xpath. They should develop the xpath/cssselector on their own.

IMPLEMENTATION/FUNCTIONAL REQUIREMENT

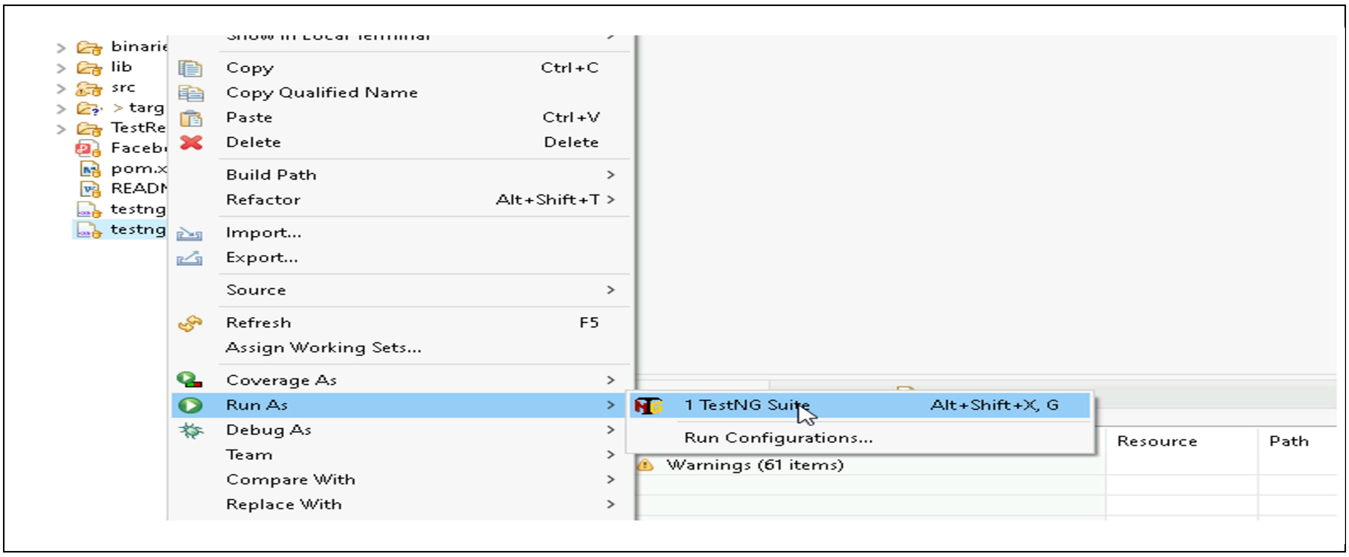
1.1 Code Quality/Optimizations

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

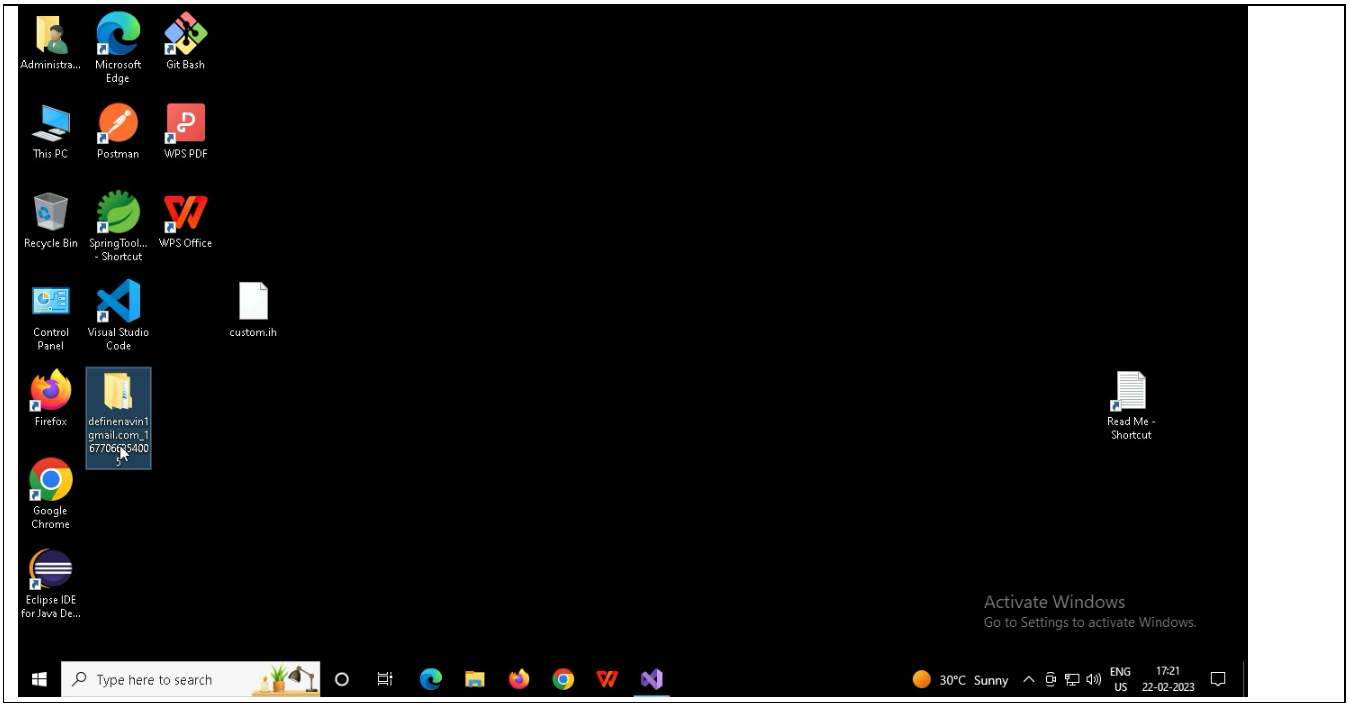
Execution Steps to Follow

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

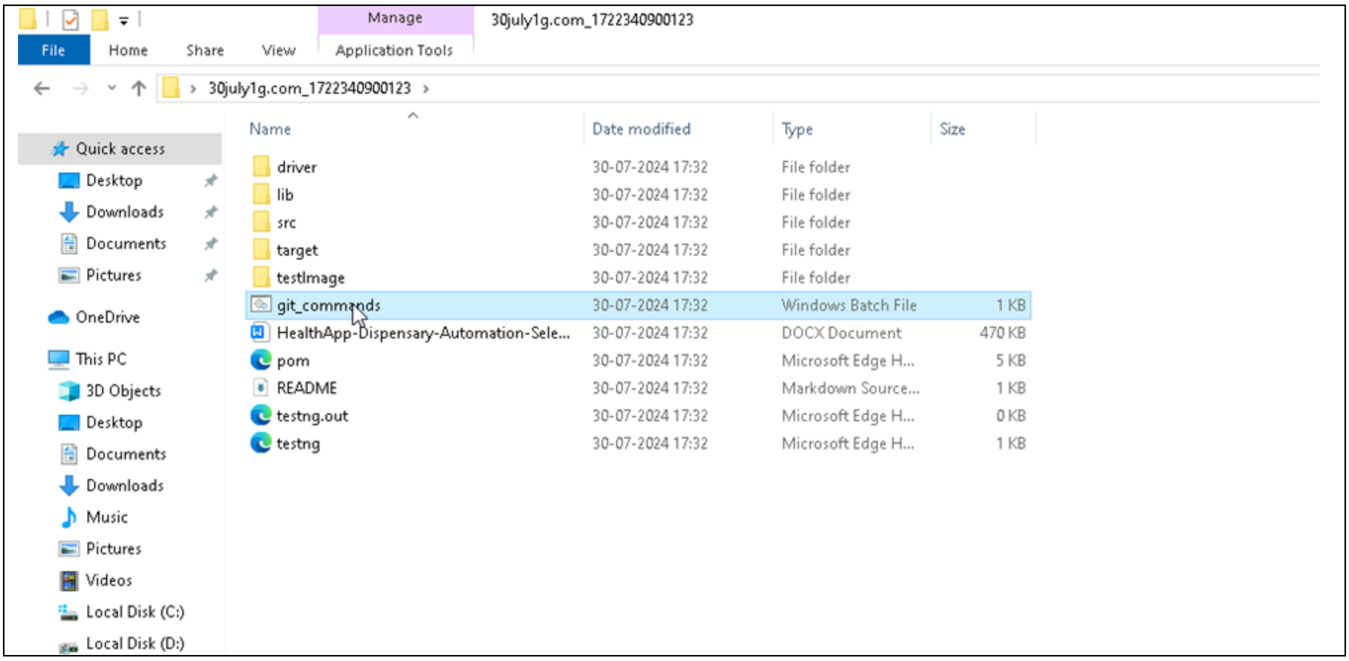
2. **You can launch test cases any time as follows: Right click on testng.xml and run TestNGSuite**



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

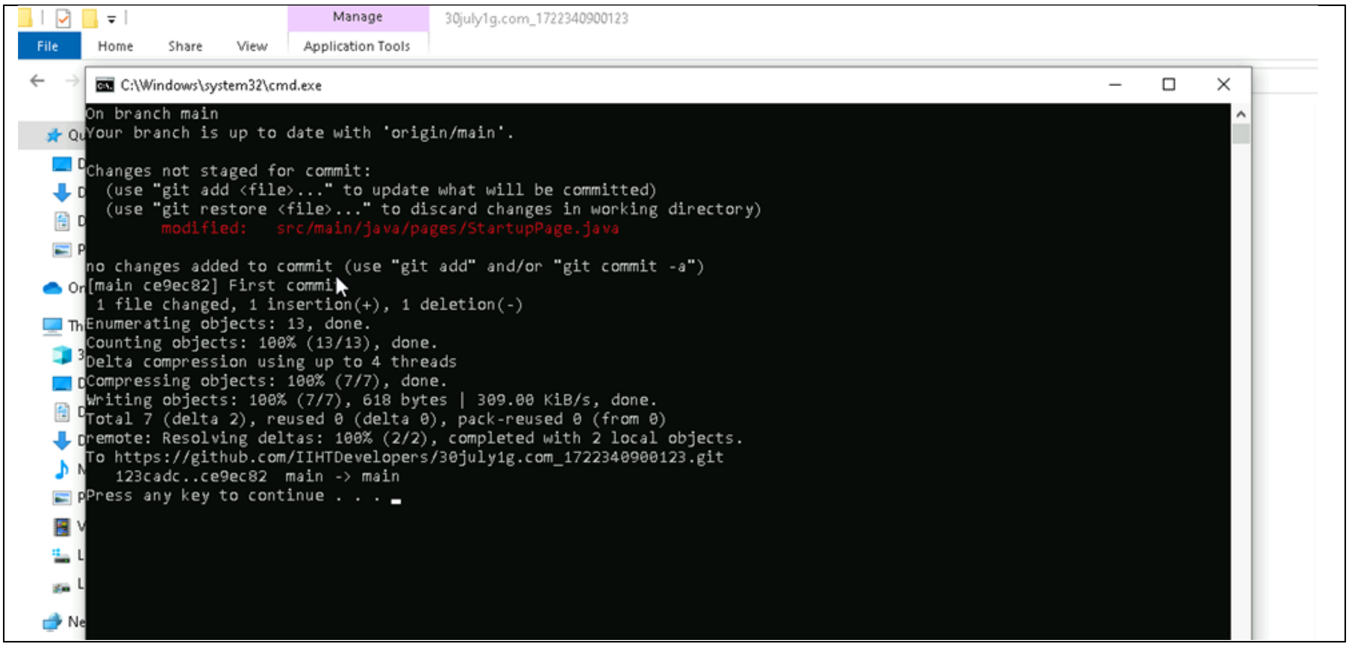


In your project folder, you will find a batch file named git_commands



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.



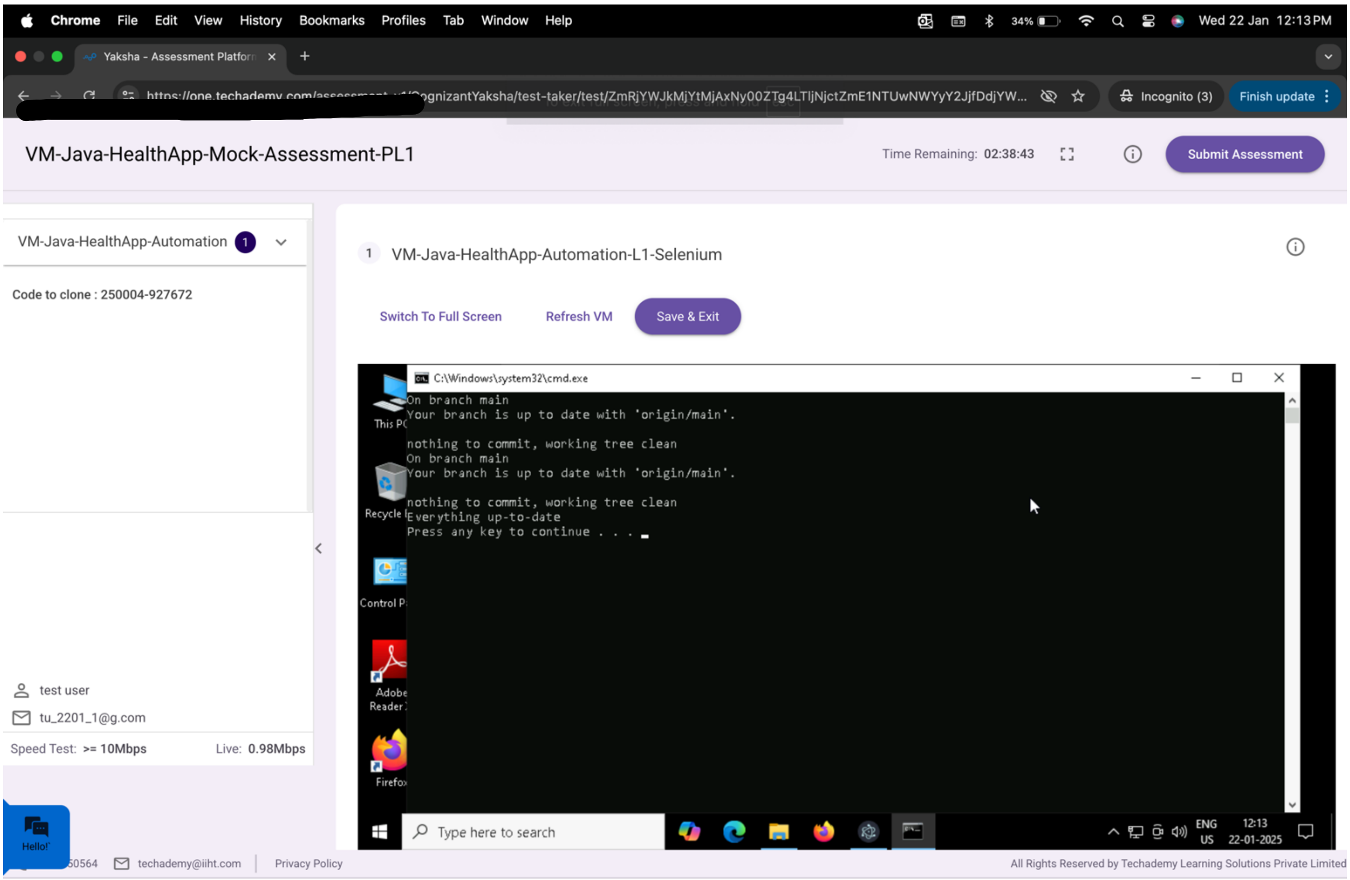
```
On branch main
Your branch is up to date with 'origin/main'.

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 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 . . .
```

Once the code is pushed to git, you can go for the final submission of the assessment.

- Press escape to come out of Fullscreen mode.
- Submit the assessment.



[redacted]

VM-Java-HealthApp-Mock-Assessment-PL1

Time Remaining: 02:38:43

Code to clone : 250004-927672

1 VM-Java-HealthApp-Automation-L1-Selenium

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

nothing to commit, working tree clean
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
Everything up-to-date
Press any key to continue . . .
```

test user
tu_2201_1@g.com
Speed Test: >= 10Mbps Live: 0.98Mbps

Hello!
50564
techademy@iiht.com
Privacy Policy
All Rights Reserved by Techademy Learning Solutions Private Limited

HealthApp Automation – Appointment Module

After the successful submission of the assessment, you will get a confirmation message displayed on your screen.

==

All the Best