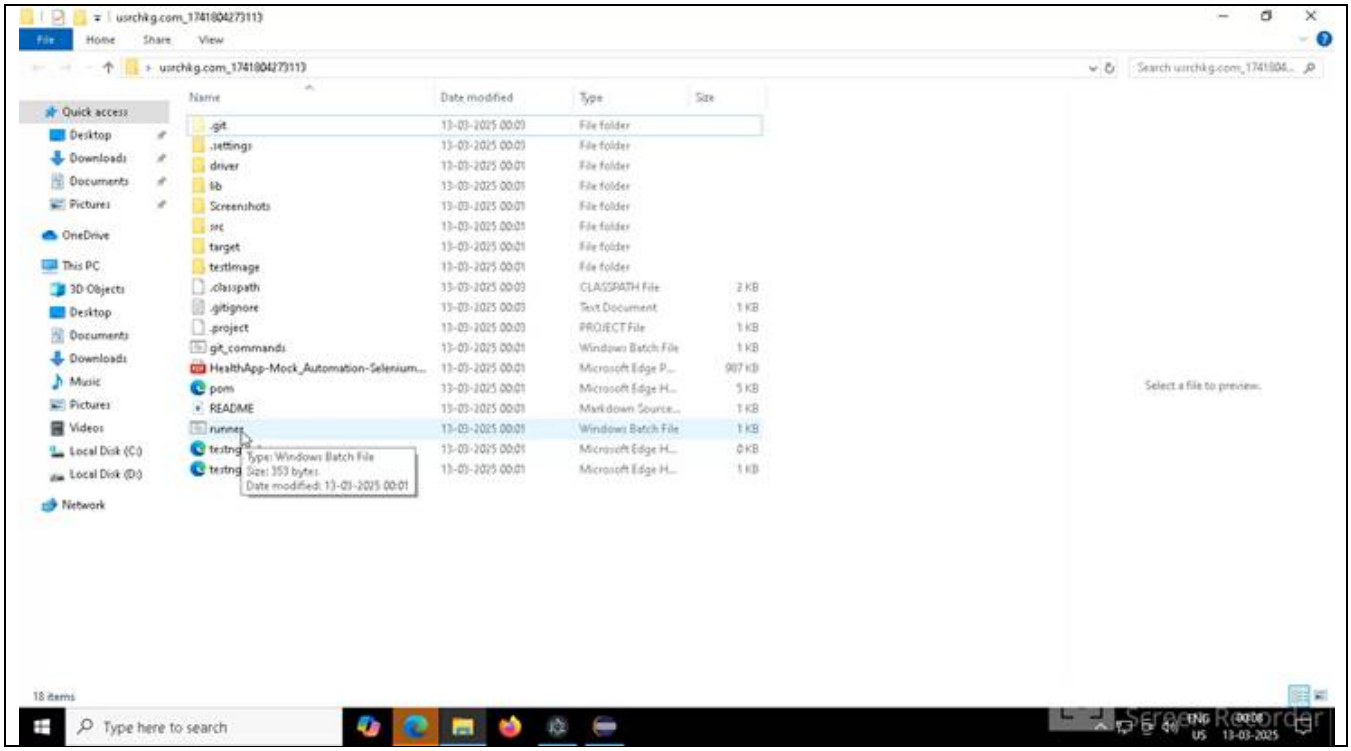
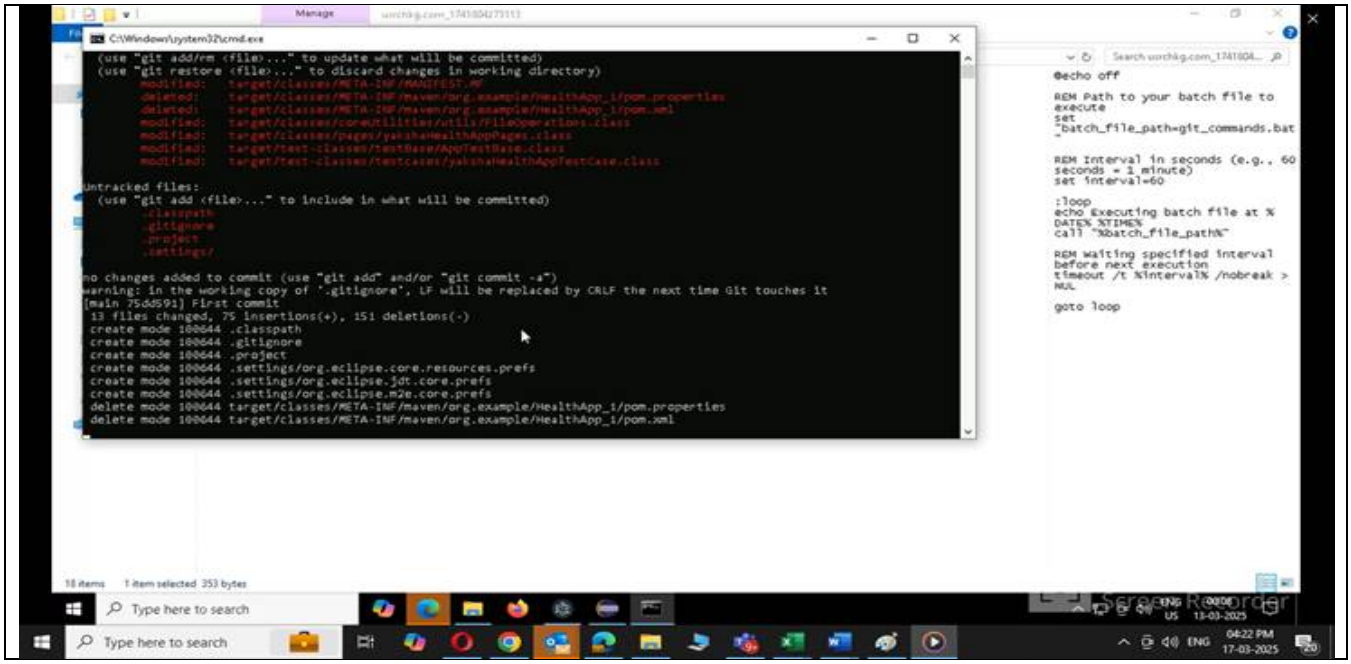
HealthApp Automation Appointment Module PL1(9TCs)

Pre-requisite:

Before you start working on your project, execute the runner file present in your project folder (Simply by double click). This is mandatory.

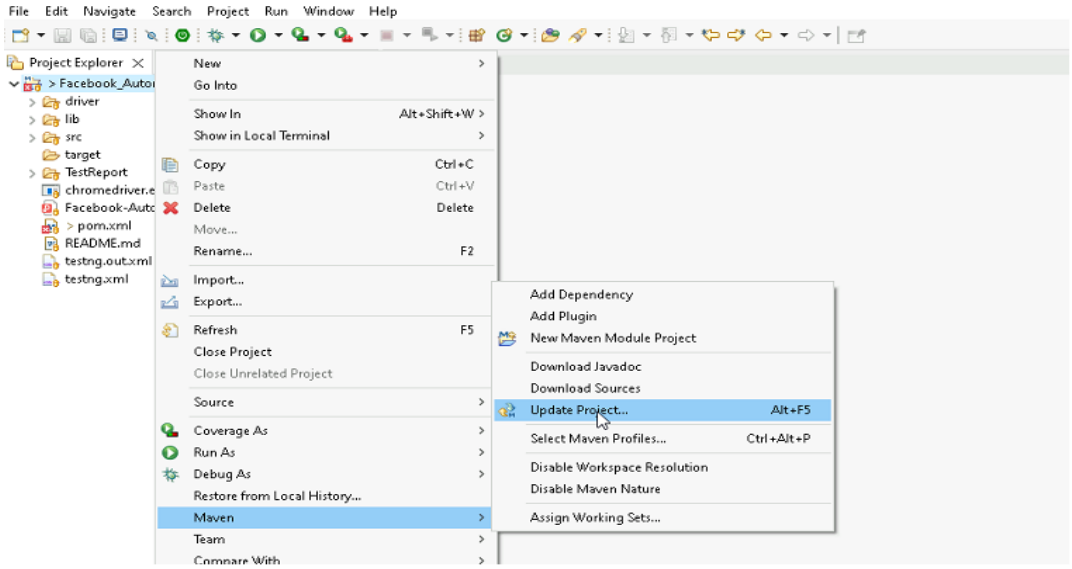


This will launch a command terminal for you where it will keep on pushing your updated code to GIT on regular intervals. Keep that command terminal open at backend and you can continue working on your project.

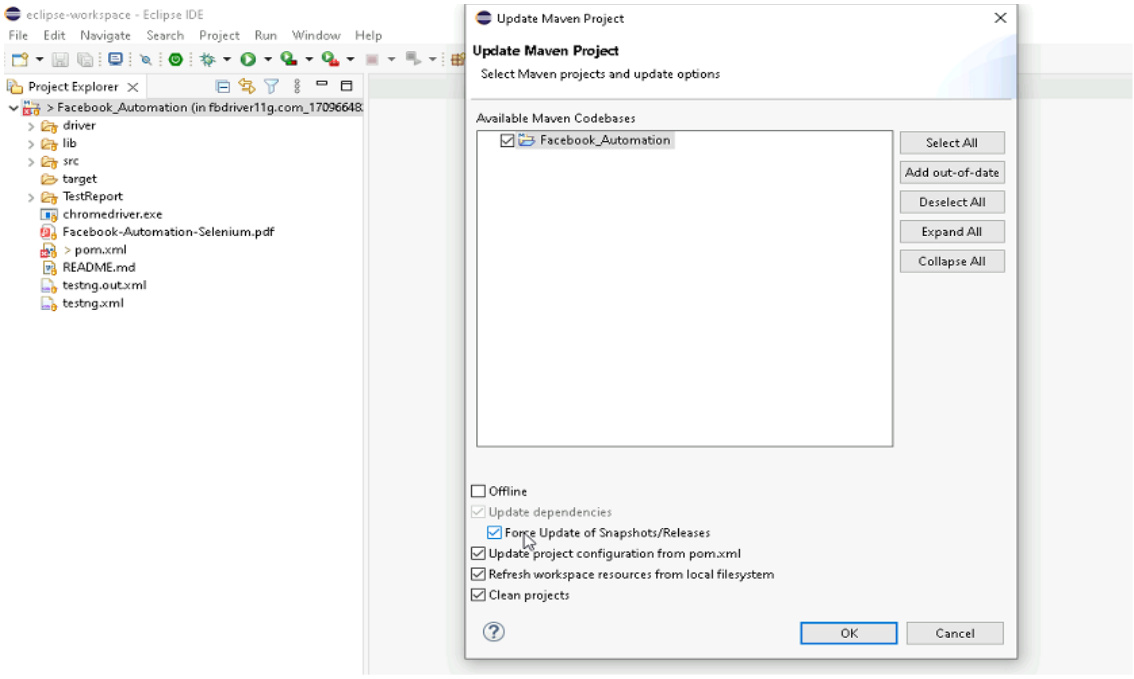


As soon as you import the project in Eclipse, update the project using the maven update option as below. This is to resolve the issue if any Maven dependency is not downloaded properly:

1. Launch Eclipse and import the project as “Existing Maven project”.
2. Right-click on the project: Go to “Maven” and select “Update Project”.



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



Template Code Structure:

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

b. Other Files and packages you can ignore.

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

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

Package	Class/File	Description
src/main/java/coreUtilities/utils/	FileOperations.java	1. It contains methods to read data from Excel files.
/src/main/java/pages	appointment_Pages.java	1. All core activities (mentioned in the list "Key Activities to Implement") are 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 the CommonEvents class.
/src/main/resources/	Config.xlsx	URL to navigate to, already URL is defined here
/src/main/resources/	expected_data.xlsx	Contains data to fill in the form
/src/main/java/coreUtilities/utils	CommonEvents.java	1. Contains all common activities. 2. Certain templated common methods are declared here. 3. You implement them as per your needs. 4. You can add any additional method for common

		activity here
	Testng.xml	Execution needs to be kick-started from TestNG.xml

Test Suite Objective / Verification of the below CUJ :

The primary goal of this **Selenium Test Suite** is to ensure that users can interact seamlessly with the **Appointment Module** of the application, with every step of the user journey carefully validated. The journey begins with verifying the accessibility and functionality of the **Appointment Module** on the homepage, followed by actions like creating a new patient, filling out patient information, and successfully submitting forms.

As users navigate through these steps, the tests validate that the relevant form fields, buttons, and error handling mechanisms are in place. Specifically, the suite tests critical aspects such as the presence and functionality of the "New Patient" button, ensuring that the "Patient Information" form displays correctly when the button is clicked. It also checks that placeholders are correct, and proper error messages appear when the form is incomplete.

Further, this suite tests the integration with the backend APIs, ensuring that appointments can be created, searched, canceled, and managed programmatically using authorized API calls. This includes ensuring that both the frontend and backend work in tandem to allow for seamless appointment booking, patient searches, and retrieval of necessary store information.

By the end of this testing suite, we ensure that both the UI elements and API methods in the Appointment Module function as expected, offering a reliable and efficient experience for the user.

PROBLEM STATEMENT

Need to automate the following activities using Java, Selenium and REST Assured to verify the complete CUJ(Customer user journey) described above.

Prerequisite Activates to implement :

1. Go to URL: https://healthapp.yaksha.com/
2. login as a valid credential (username: **admin**, password: **pass123**) and click on "Sign in"

Key Activities to implement:

Sl No.	Summary	Action	Expected Result

1	Navigate to the URL given and retrieve the Title and URL of the current page.	1. Get the title and URL of the Home page, post login 2. Validate the title and URL of the Home page	The title should be : DanpheHealth URL should be : https://healthapp.yaksha.com/Home/Index#/
2	Ensure that the Appointment module is present.	**Preconditions**: The user must be logged into the health system. **Steps**: 1. Click on the Appointment module.	Verify that the select Counter popup should come and the popup page name should be "Select Counter".
3	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. Click on the "New 1" link within the "Select Counter" popup. 2. Click on the "New Patient" Button.	Verify that upon clicking the "New Patient" button, the "Patient Information" text should appear.
4	Ensure that the "Care of Person Contact" textbox is prest in the “Patient Information” section.	**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. Scroll to the Bottom of the Page (if required) 2. Click on the "Care of Person Contact" Textbox.	Verify that the "Care of Person Contact" textbox should be present.
5	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**: **NA (Not applicable here)**	placeholder text of the **"Care of Person"** textbox should be **"Care Taker Person"**
6	Ensure the error message in the "Patient Information" form's last name text field after clicking on the "Print Invoice" Button without filling any information in the form.	**Preconditions**: The user must be logged into the health system. The user is on the 'New Visit' page within the Appointment module. **Steps**: 1. Scroll to the bottom of the page(if required) 2. Click the “Print Invoice” button. 3. Click the “Confirm” button of the pop-up.	Verify that the error message specifically states "Last Name is required."
7	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	- 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.

		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) Return the CustomResponse object from the method.	
8	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) 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	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) 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.

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 Selenium to automate all the steps in the above question. In other words, the automation script should perform all the 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

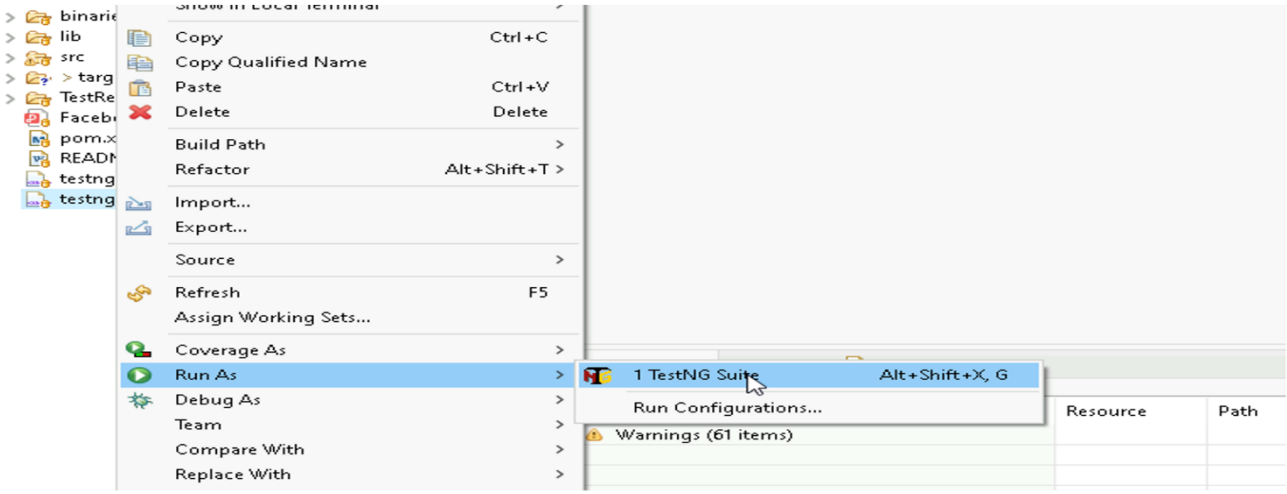
1.1 CODE QUALITY/OPTIMIZATIONS

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

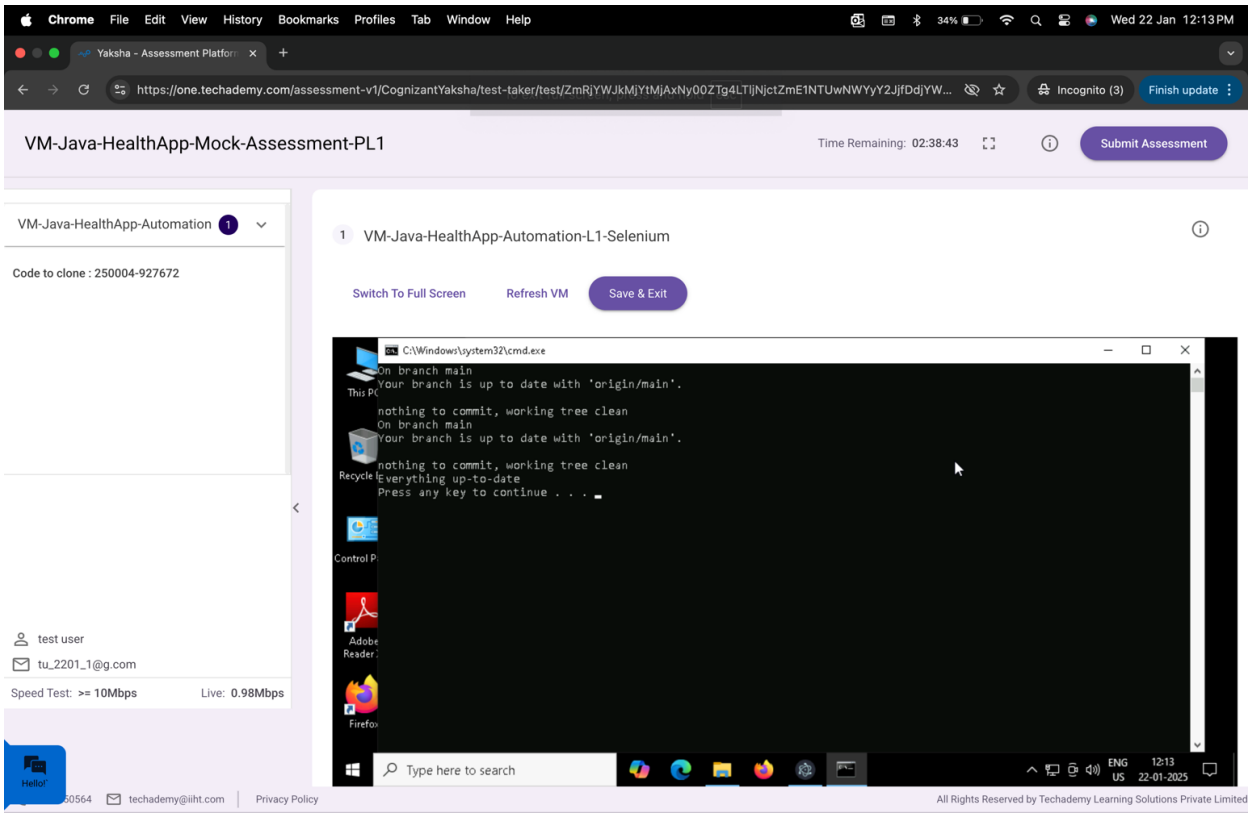
EXECUTION STEPS TO FOLLOW

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

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



3. To do the final submission of the assessment :
    a. Press escape to come out of Fullscreen mode.
    b. Submit the assessment.



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

==========

All the Best