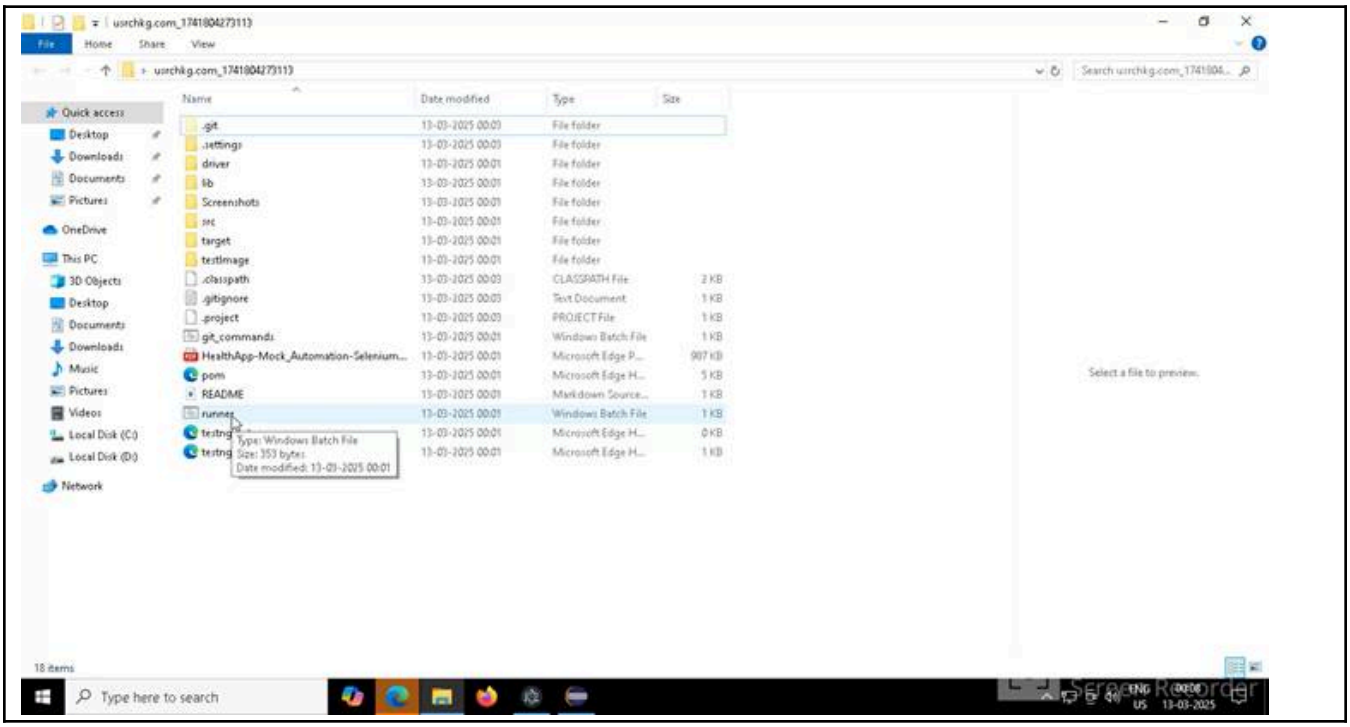
HealthApp Automation Using C# and Selenium

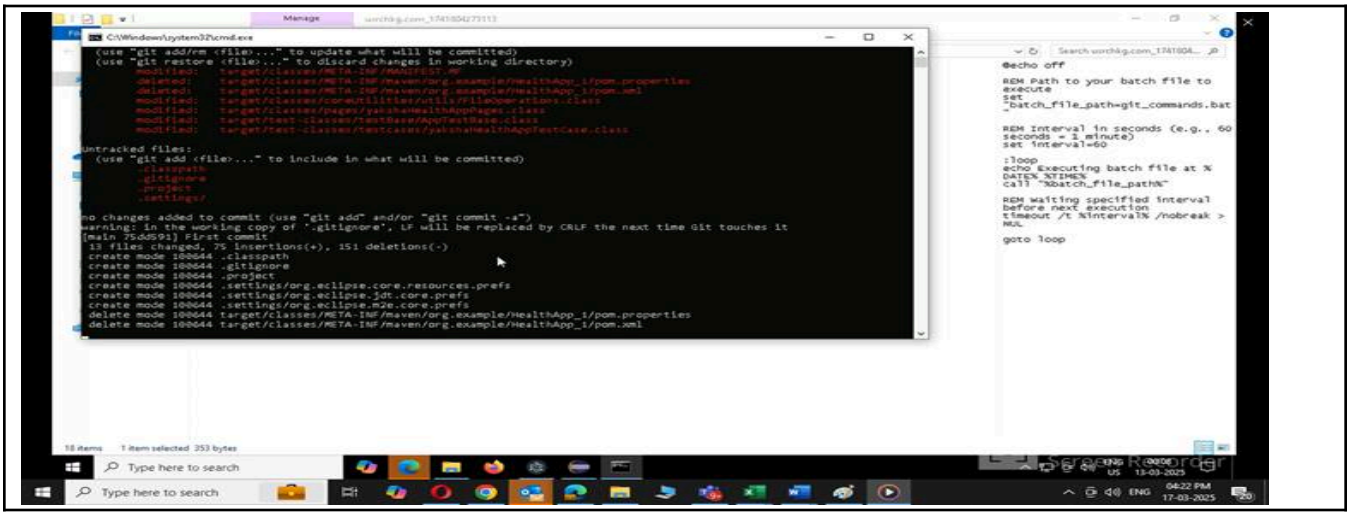
VM-DotNet_PL2-Set 1 (9 TCs)

Pre-requisite:

Before you start working on your project, you must execute the runner file present in your project folder (simply by double-clicking). This is mandatory.

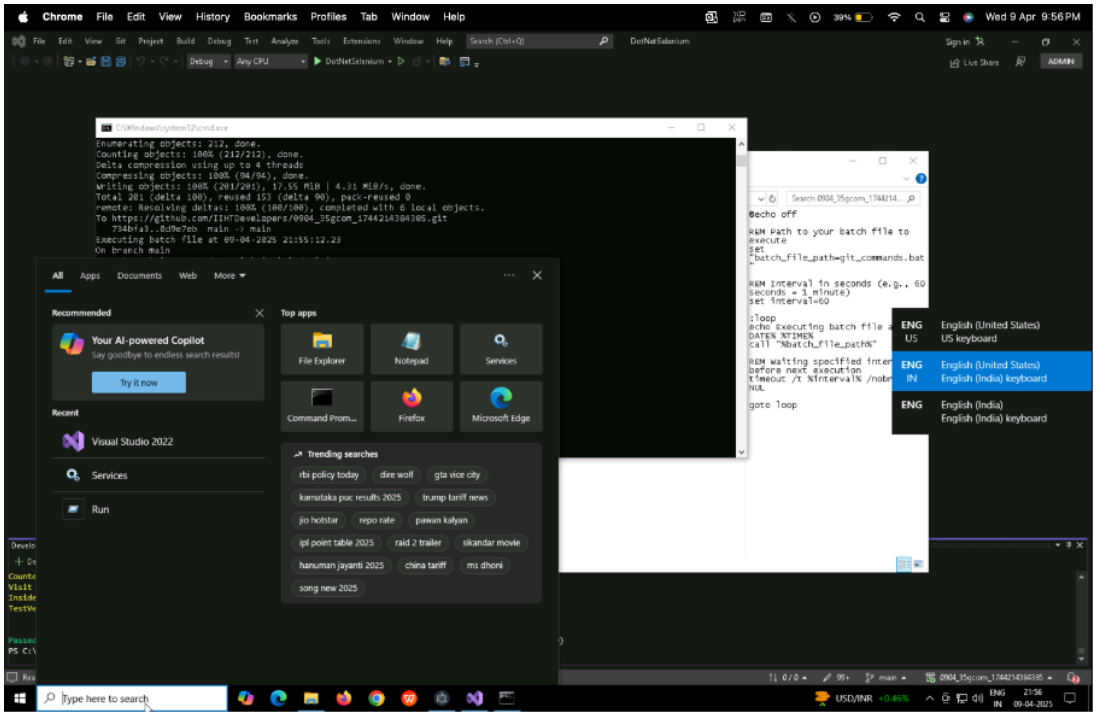


This will launch a command terminal for you where it will keep on pushing your updated code to GIT on regular intervals. Keep that command terminal open at backend and you can continue working on your project.

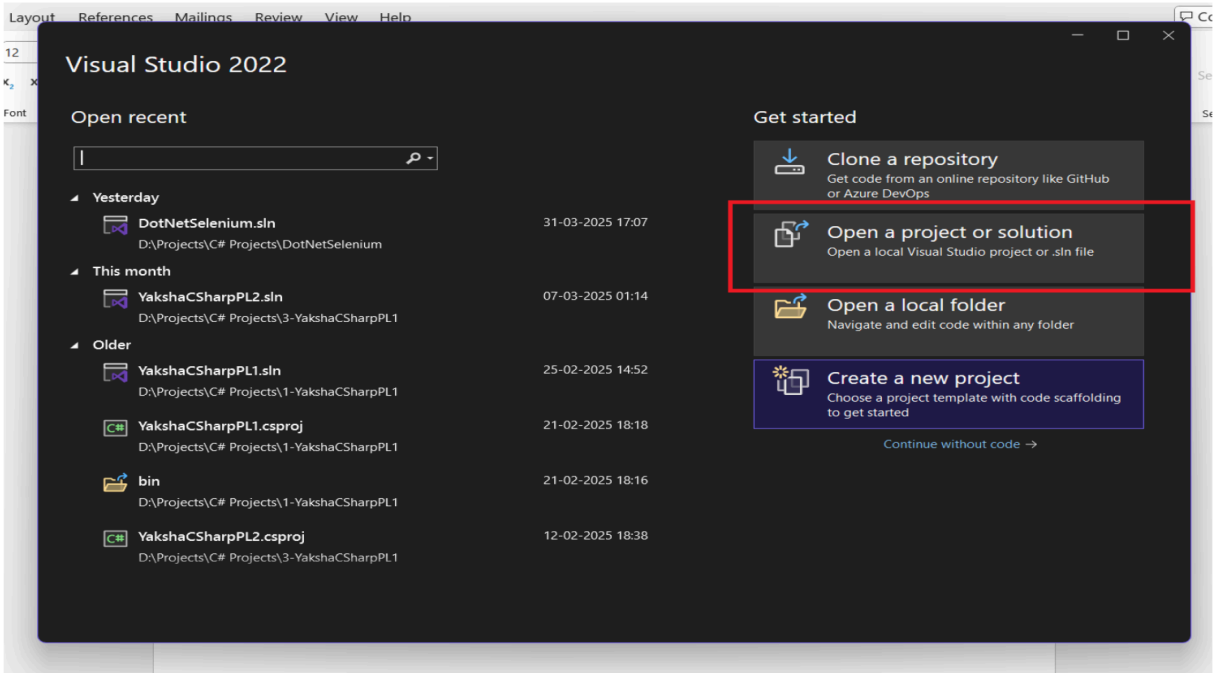


As soon as you import the project into Visual Studio 2022, Please follow the below steps:

1. Open the Visual Studio code.



2. Open Your C# Selenium Project
    a. Launch Visual Studio.
    b. Open your Selenium automation project (.sln file).



Template Code Structure:

a. The packages and files you will be required to work on are listed in the table below.

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 “TestCases” 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
/DotNetSelenium/Utilities/	JsonReader.cs	**The method for reading data as input from a JSON file.**
/DotNetSelenium/PageObjects	AdminPage.cs AppointmentPage.cs DoctorPage.cs IncentivePage.cs LoginPage.cs OperationTheatrePage.cs PatientPage.cs ProcurementPage.cs SettingsPage.cs UtilitiesPage.cs (Each file has the TestCase function in templated form here. You need to write the required logic and xpath into it.)	1. All core activities listed below in the activity table 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 if required.
/DotNetSelenium/TestData	LoginData.json PatientName.json	It contains data for login and filling in the form.

PROBLEM STATEMENT

We need to automate the following activities using Selenium + C#.

Before the execution of any test steps, the user must be logged in to the health app.

Key Activities to implement:

#	Summary	Action	Expected Result

1	AppointmentPage.cs ========== Selects "New Patient" from the Visit Type dropdown and verifies that all results reflect "New Visit" in the Visit Type column.	Steps: ● Clicks on the Appointment link and selects the first available counter item (if present). ● Navigates to the Appointment Booking List page. ● Selects "New Patient" from the Visit Type dropdown. ● Sets the From Date to "01-01-2024" to filter results. ● Clicks the "Show Patient" button to retrieve appointment data. ● Validates that the Visit Type column displays only "New Visit" entries.	Returns True if all entries in the Visit Type column contain "New"; otherwise, returns False.
2	OperationTheatrePage.cs ========== Attempts to book an OT (Operation Theatre) without selecting a patient and verifies the alert message.	Steps: ● Navigates to the Operation Theatre module. ● Clicks on the "New OT Booking" button to open the booking modal. ● Validates that the OT booking modal is displayed. ● Clicks the "Add New OT" button without selecting a patient. ● Waits for the browser alert and captures the alert message. ● Accepts the alert to close it.	Returns: The text of the alert message shown when no patient is selected during OT booking.
3	DoctorPage.cs ========== Searches for a specific patient in the In-Patient list and verifies that their overview page is displayed.	Steps: ● Reads the patient name from a JSON test data file. ● Clicks on the Doctor module and navigates to the In-Patient tab. ● Locates the appropriate search box and enters the patient's name. ● Waits briefly and clicks the preview icon from the Actions column. ● Retrieves and returns the patient name heading from the overview page for validation.	Returns the patient name displayed on the overview page, converted to lowercase and trimmed.
4	ProcurementPage.cs ========== This method navigates to the Purchase Request page, accesses the Currency Settings, adds a new currency with a unique code and description, and verifies that the new currency is successfully added to the table.	Steps: ● Click on the "Procurement" link to open the procurement module. ● Navigate to the "Settings" tab and then to the "Currency" sub-tab. ● Click the "Add Currency" button to open the currency input form. ● Enter a unique currency code and a test description. ● Click the "Add Currency" button to submit the form. ● Use the search bar to find the newly added currency. ● Verify that the currency appears in the table with the correct code.	The new currency should be added successfully and displayed in the table with the correct currency code and description. Returns True if the currency is added and visible in the table; otherwise, false.
5	UtilitiesPage.cs ========== This method verifies that a warning pop-up is displayed when attempting to save a new Scheme Refund entry without filling in any mandatory fields.	Steps: ● Click on the "Utilities" link. ● Navigate to the "Scheme Refund" tab. ● If any counter items are available, select the first one. ● Click on "New Scheme Refund Entry". ● Click the "Save" button without entering any data.	A warning message should be displayed indicating that mandatory fields must be filled before submission.

		• Capture and return the warning message displayed in the pop-up.	
6	AdminPage.cs ========== Navigates to the "User Profile" page via the Admin dropdown and verifies successful navigation.	Steps: • Clicks on the Admin dropdown in the header. • Selects the "My Profile" option from the dropdown menu. • Waits for the User Profile page to load. • Retrieves and returns the header text of the User Profile page for validation.	The header text from the User Profile page, used to assert successful navigation.
7	PatientPage.cs ========== This method uploads a profile picture for a patient by navigating to the "Register Patient" tab and completing the image upload workflow.	Steps: • Click on the "Patient" link from the navigation menu. • Select the "Register Patient" tab to open the registration section. • Click on the profile picture icon followed by the "New Photo" button. • Upload a test image from the predefined directory. • Wait for the image to upload and click the "Done" button to finalise. • Verify that the uploaded image is displayed on the UI.	The uploaded profile image should be visible on the patient registration screen, confirming success. Verify that it returns True if the image is displayed after upload, otherwise, it throws an exception.
8	IncentivePage.cs ========== Edits the TDS percentage for a specific employee and verifies that the updated value is reflected in the UI.	Steps: • Reads the employee name from a JSON test data file. • Navigate to the Incentive module and select the Settings tab. • Searches for the employee using the search bar. • Clicks the "Edit TDS%" button to open the TDS input modal. • Clears the existing value and enters a new random TDS percentage. • Clicks the "Update TDS" button to save changes. • Repeats the search and retrieves the updated TDS value from the table. • Verifies that the displayed TDS value matches the newly entered value.	Returns true if the updated TDS percentage is correctly reflected; otherwise, throws an exception.
9	SettingsPage.cs ========== This method disables and then re-enables a price category (e.g., "NHIF-1") via the Price Category tab, verifying that appropriate success messages are displayed for both actions.	Steps: • Click on the "Settings" link to navigate to the settings module. • Click on the "More..." dropdown and select the "Price Category" tab. • Click on the "Disable" button for the code "NHIF-1". • Verify that the message "Deactivated" appears after disabling. • Click on the "Enable" button for the same code. • Verify that the message "Activated" appears after enabling.	Verify that the Success messages "Deactivated" and "Activated" should be displayed after each corresponding action.

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

Expectations:

1) Learners should write automation scripts using C# 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

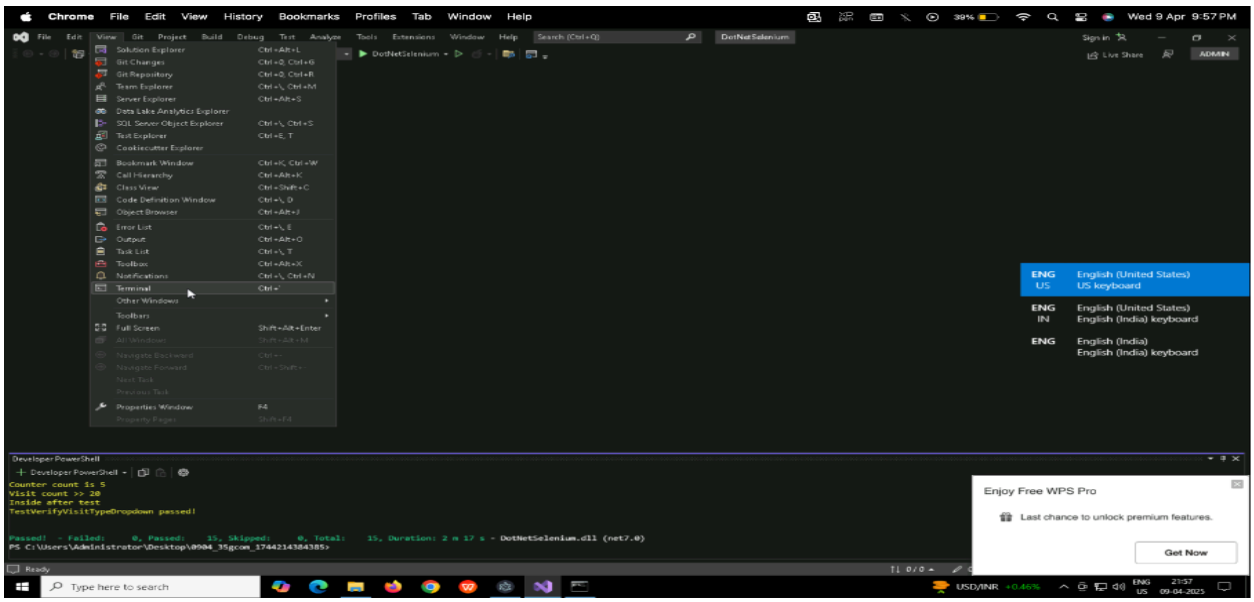
1.1 CODE QUALITY/OPTIMIZATIONS

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 required to run test cases for applications before final submission. Without this, the project evaluation will not happen.**

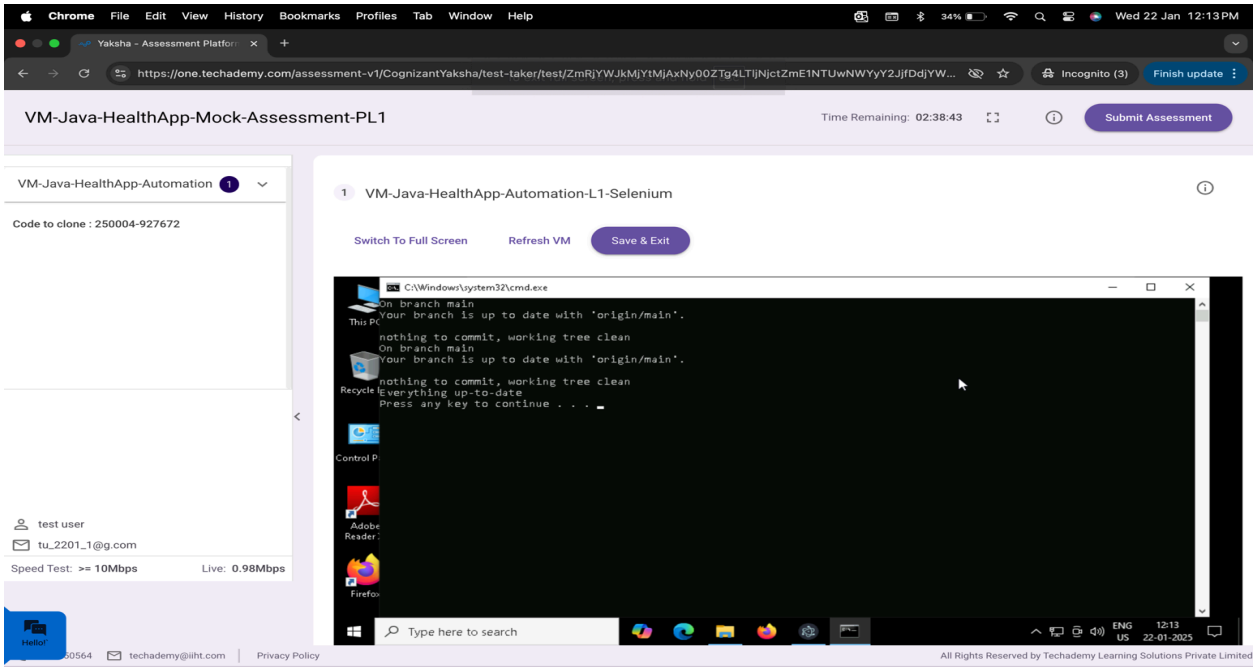
2. **You can launch test cases at any time as follows: go to View menu and open a new terminal window.**



3. **Use the following commands :**

a. **Dotnet clean (to clean the project)**
b. **Dotnet build (to build the project with latest changes)**
c. **Dotnet test (to execute the test cases)**

4. **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