

YAKSHA HEALTH APP WITH TYPESCRIPT AND PLAYWRIGHT

Mymedic automation using playwright

Usecase summary

Project Name: healthapp.yaksha app – Medical Record Management System

Use Case Summary: healthapp.yaksha is a healthcare application designed to manage Electronic Medical Records (EMR). It allows users to view, search, and manage patient records. It features functionality such as adding/editing patient records, filtering data by doctor and department, and exporting records. The primary use case is to automate the process of medical record management, ensuring efficient and reliable operations for healthcare providers.

Technology Stack:

- **Automation Tool:** Playwright (for testing)

Key Features:

- **Patient Record Management:** Add, edit, and delete patient records.
- **Filtering and Search:** Search medical records by date range, doctor, department, and more.
- **Export Functionality:** Export records for offline access.

Expected Outcomes:

- Automate key healthcare operations like patient record handling, filtering, and validation.
- Ensure the accurate retrieval and modification of medical records, enhancing operational efficiency.

Overview of the application

Pages/Features that are to be focused for the application

Please use the Application URL <https://healthapp.yaksha.com>

PROBLEM STATEMENT

Need to automate the following activities using playwright+typescript

You will be given a excel file named to validate and search data

Path	File	Description
src\data	result.xlsx	1. Contains data to read from excel file.
src\tests\commonMethods	readExcel(filePath, sheetName)	Should be implemented such way that it reads the data from shared "sheetName" from excel of given "filePath" and return the data in Record<string, string>

Mymedic automation using playwright

src\ pages	<ul style="list-style-type: none"> • AppointmentPage • UtilitiesPage • DispensaryPage • ProcurementPage • LoginPage • PatientPage • ADTPage • RadiologyPage • LaboratoryPage 	<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.
src\tests	keywords.ts	Implement methods SearchPatients() and VerifyResults()

Here's a detailed table format for the test cases to be tested

Test Case No.	Test Case Name	Test Steps to be performed	Path & Method Used	Expected Result
1	Verify Login with Valid Credentials	<ol style="list-style-type: none"> 1.the application will read the result.xlsx file to fetch the user name and password using "login" string from common methods 2.it should call the method Use performLogin () 3.perform login method will perform authentication with the username and password 4. Verify admin name is visible on the home page. 	Reference path \src\pages\LoginPage methods performLogin()	Successfully logs in with provided credentials. The user is logged in the admin page Returns true for success; otherwise, false.
2	Verify Page Navigation and Load Time for Billing Counter	<ol style="list-style-type: none"> 1. Use verifyBillingCounterLoadState() to check module load. 2. Open "Change Billing Counter" module using ChangeBillingCounter. 3. Set acceptable load time: 1000ms. 4. Verify counter presence; select the first counter if available. 5. Log a message if no counters are found. 	Reference path \src\ pages\ UtilitiesPage methods verifyBillingCounterLoadState()	Navigates to "Change Billing Counter". Proceeds if counters are available and loaded within 1 second; logs a message otherwise.
3	Patient Search with Valid Data	<ol style="list-style-type: none"> 1. Navigate to Appointment page using navigateToAppointmentPage(). 2. Use selectFirstPatient() to verify the first patient's name. 3. Search for a patient by name using the search bar and press Enter. 4. Validate that search results contain the searched name. 5. Verify results using verifyPatientNames(). 	Reference path \src\ pages\ AppointmentPage methods navigateToAppointmentPage(), searchPatient()	Displays patient name in search results. Ensures short name matches the searched data.
4	Activate Counter in Dispensary	<ol style="list-style-type: none"> 1. Use verifyActiveCounterMessageInDispensary() to: <ul style="list-style-type: none"> - Navigate to the Dispensary page. - Select a random counter if available. - Activate the counter and verify the 	Reference path \src\ pages\ DispensaryPage methods	Counter activation message matches the selected counter name. Returns true for success; false otherwise.

Mymedic automation using playwright

Test Case No.	Test Case Name	Test Steps to be performed	Path & Method Used	Expected Result
		activation message. 2. Log counter selection and activation status.	verifyActiveCounterMessageInDispensary()	
5	Purchase Request List Load	1. Use verifyPurchaseRequestListElements() to check verify of elements: purchaseRequest, purchaseOrder, goodsArrivalNotification, quotations, settings, reports, favoriteButton, okButton, printButton, firstButton, previousButton, nextButton, lastButton.	Reference path \src\ pages\ ProcurementPage methods verifyPurchaseRequestListElements()	All elements are present and visible on the procurement page Returns true for success; logs missing elements and returns false otherwise.
6	Verify Error Message While Adding New Lab Test	1. Navigate to Laboratory > Settings. 2. Select "Add New Lab Test". 3. Click "Add" without providing inputs. 4. Capture and verify the error message: "Lab Test Code Required".	Reference path \src\ pages\ LaboratoryPage methods verifyErrorMessage()	Error message: "Lab Test Code Required" is displayed. Logs success or failure and ensures the modal is closed.
7	Handle Alert on Radiology Module	1. Navigate to Radiology module and select "List Request" sub-module. 2. Apply filter using dates . 3. the application will read the data from excel file using the common methods name ("DateRange") (eg fromDate: 01-01-2020 to toDate: 11-11-2024.) 3. Click "Add Report" button. 4. Use handleAlert() to verify and accept the alert if the message matches. 5. the application will check for the matched data .	Reference path \src\ pages\ RadiologyPage methods handleAlert()	Alert dialog matches expected message and is accepted. Returns true for success; false for failure in handling the alert.
8	Data-Driven Testing for Patient Search	1. Navigate to Patient Section. 2. application will use the common method to use read the excel to search for patient which is there in the dataset using the name ("patientnames") 3. Use the search bar to find patients. 4. Compare retrieved names with expected names from Excel. 5. Log success or failure for each match.	Reference path \src\ pages\ PatientPage methods searchAndVerifyPatients()	Matches all patient names from Excel. Returns true for success; logs mismatches and returns false if any fail.
9	Error Handling and Logging in Purchase Request List	1. Navigate to Procurement module. 2. Apply invalid date filter. 3. Click "OK". 4. Capture and verify the error message: "Date is not between range. Please enter again."	Reference path \src\ pages\ ProcurementPage methods verifyNoticeMessageAfterIncorrectFilters()	Triggers and verifies the error message for invalid date range. Logs success for match; failure for mismatch.
10	Keyword-Driven Framework for	1. Use searchAndVerifyPatient() to: - Retrieve patient name. - Validate patient name is not empty.	Reference path \src\ pages\ AppointmentPage	Successfully retrieves and verifies patient in search results.

Mymedic automation using playwright

Test Case No.	Test Case Name	Test Steps to be performed	Path & Method Used	Expected Result
	Appointment Search	2. Verify results contain the searched patient. 3. Timeout is set to 2 seconds.	methods searchAndVerifyPatient(), verifyResults()	
11	Modular Script for Patient Search	1. Navigate to the Appointment section. 2. Use searchPatientInAppointment() to execute the search. 3. Validate the patient search result.	Reference path \src\ pages\ AppointmentPage methods searchPatientInAppointment()	You should be able to search patient search data
12	Verify Assertion for Counter Activation	1. Navigate to the Dispensary section. 2. Verify the visibility of the counter button. 3. Activate counter using activateCounter button. 4. Verify deactivateCounterButton visibility.	Reference path \src\ pages\ DispensaryPage methods verifyCounterisActivated()	Returns true if counter activation is successful. Returns false if activation fails or required element is not found.
13	Verify Locator Strategy for Appointment Search	1. Navigate to Appointment page. 2. Verify visibility of patient list. 3. Use the search bar to find a patient by name or hospital code. 4. Verify search results.	Reference path \src\ pages\ AppointmentPage methods searchAndVerifyPatientList()	Verify that each patient's name in the result matches the search term
14	Verify Tooltip Text on Star Icon in Laboratory	1. Navigate to Laboratory Dashboard. 2. Verify visibility of star icon. 3. Hover over the star icon and retrieve tooltip text. 4. Log the retrieved tooltip text.	Reference path \src\ pages\ LaboratoryPage methods retrieveTooltipText(), hoverOverElement()	"remember this tool tip" message should be displayed Returns empty string if no tooltip text is present.
15	Navigation Exception Handling on Dispensary Page	1. Attempt navigation to Dispensary page with maxRetries. 2. Click activateCounter button. 3. Confirm page load by verifying key element presence.	Reference path \src\ pages\ DispensaryPage methods navigateToDispensary()	activation of the should be enabled
16	Web Element Handling for Dropdowns in Purchase Request	1. Navigate to Purchase Request List. 2. Apply date range filter. 3. Retrieve dates from "Requested Date" column and validate each date within range. 4. the data range will be called from the common methods "data range" 5. wait for success or failure	Reference path \src\ pages\ ProcurementPage methods verifyRequestedDateColumnDateWithinRange()	It should be able to retrieve dates are within range. Returns false if any date falls outside the range.
17	Login with Invalid Credentials	1. Reset state by logging out if already logged in. 2. Use performLoginWithInvalidCredentials to check the invalid user.	Reference path \src\ pages\ LoginPage methods	Displays error message: "Invalid User". Logs success if message matches; logs failure otherwise.

Test Case No.	Test Case Name	Test Steps to be performed	Path & Method Used	Expected Result
		3 loginData common method is used to fetch the username and password from the excel 3. Capture and verify error message: "Invalid Usser".	performLoginWithInvalidCredentials()	

Learners will gain experience in building strongly-typed applications using React.js and managing data flow with **TypeScript**. They'll learn how to define interfaces, use types for error prevention, and improve code maintainability.

With **Playwright**, learners will learn to write and execute automated tests for the <https://healthapp.yaksha.com> app. Key skills include:

- **Browser Automation:** Interacting with web elements and testing multiple browsers.
- **Assertions & Validations:** Ensuring app behavior meets expected results.
- **End-to-End Testing:** Automating real user interactions and validating overall app functionality.

IMPLEMENTATION/FUNCTIONAL REQUIREMENT

1.1 CODE QUALITY/OPTIMIZATIONS

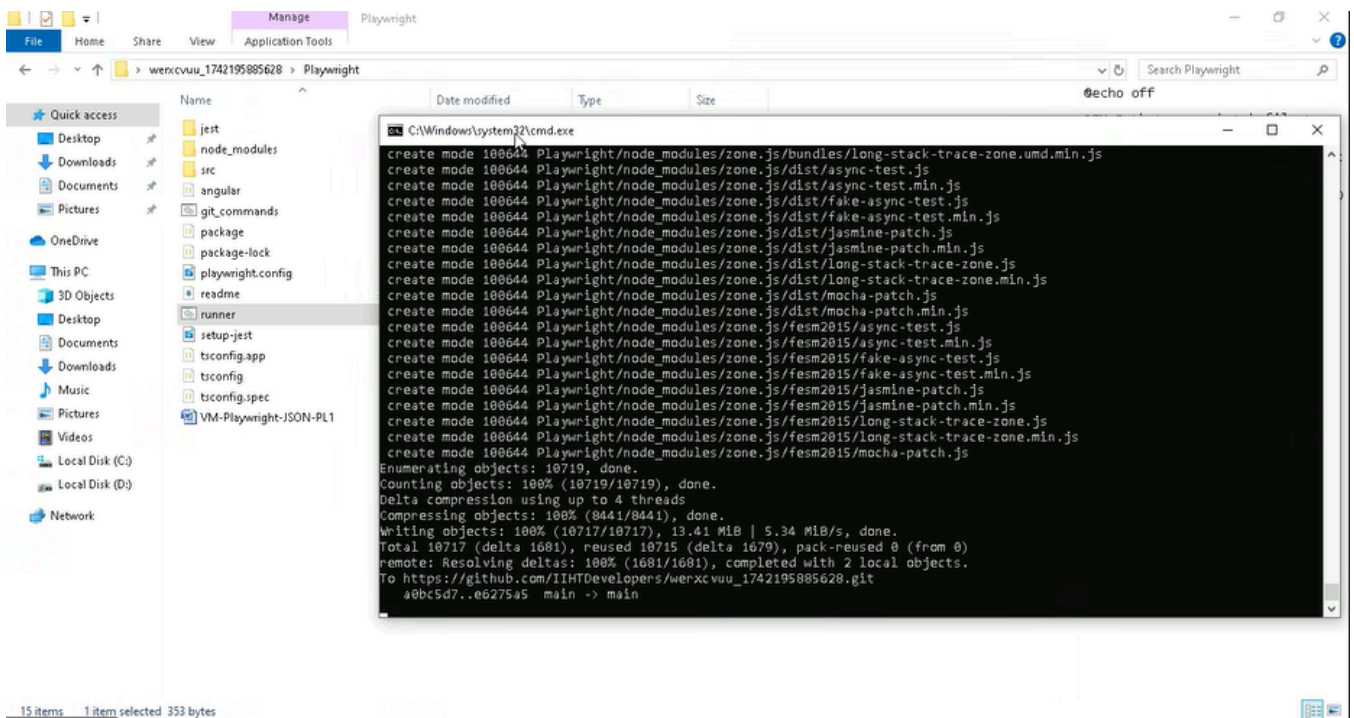
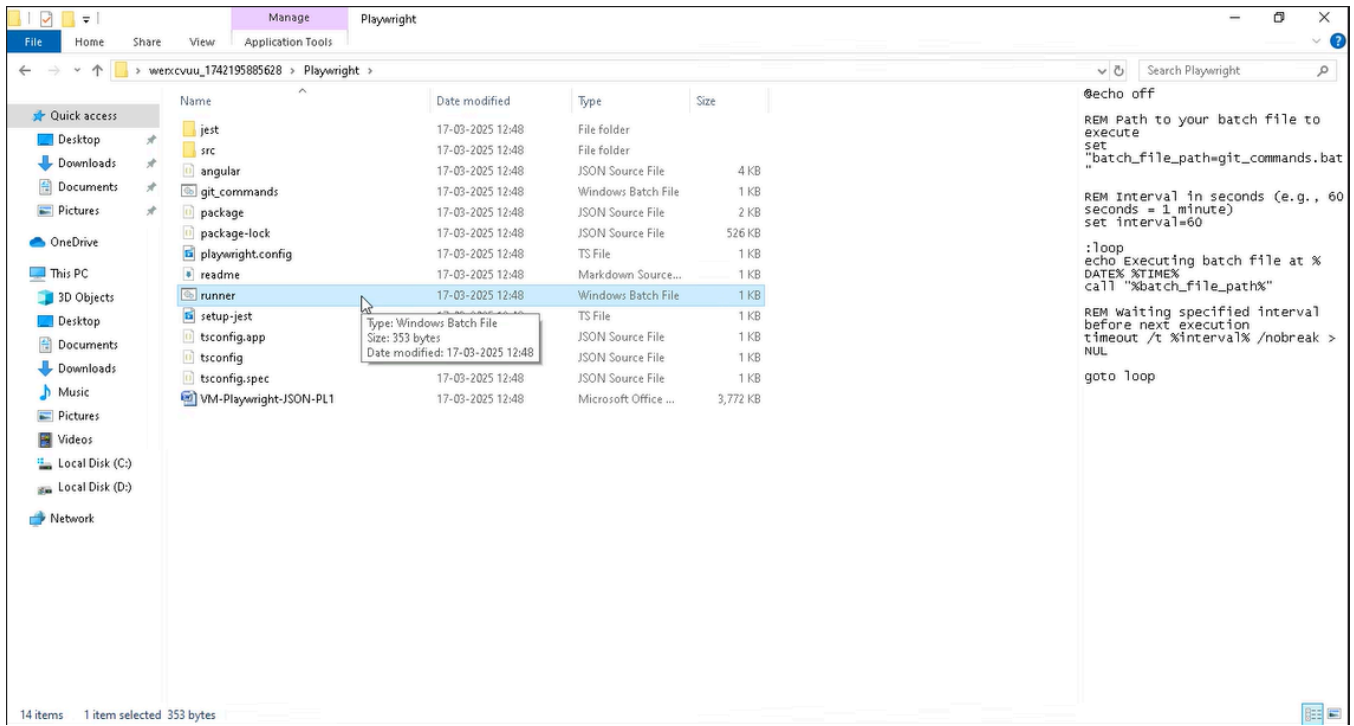
1. Associates should have written clean code that is readable.
2. Associates need to follow SOLID programming principles.

Execution Steps:

Steps for Execution:

1. Please open the folder created on desktop with the email name you used to login.

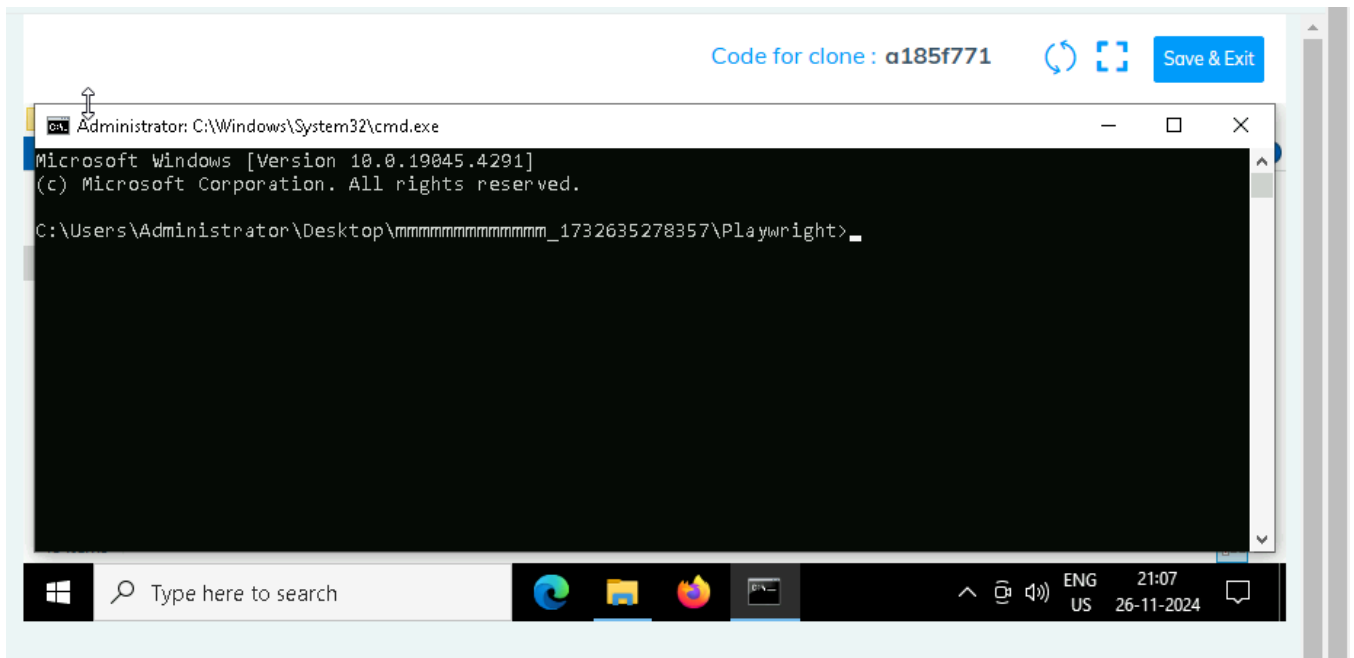
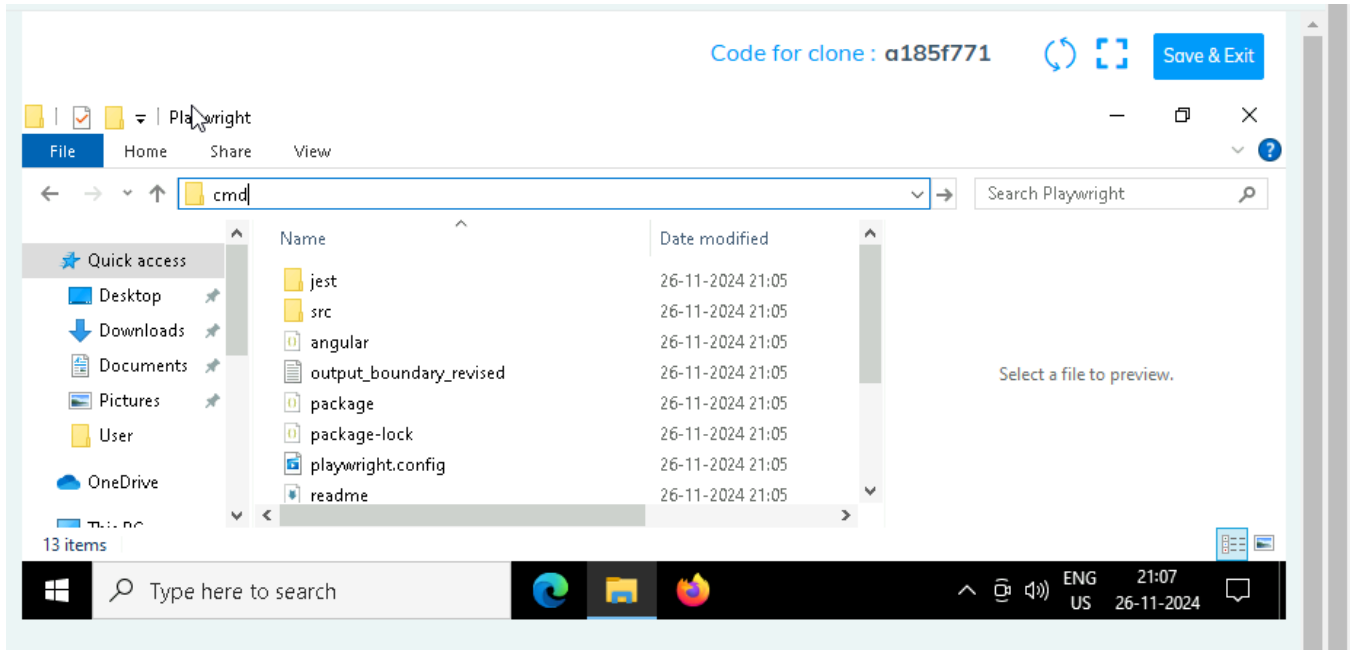
2. Go into the Playwright folder and execute this “runner” file. This will keep pushing the code at regular intervals.



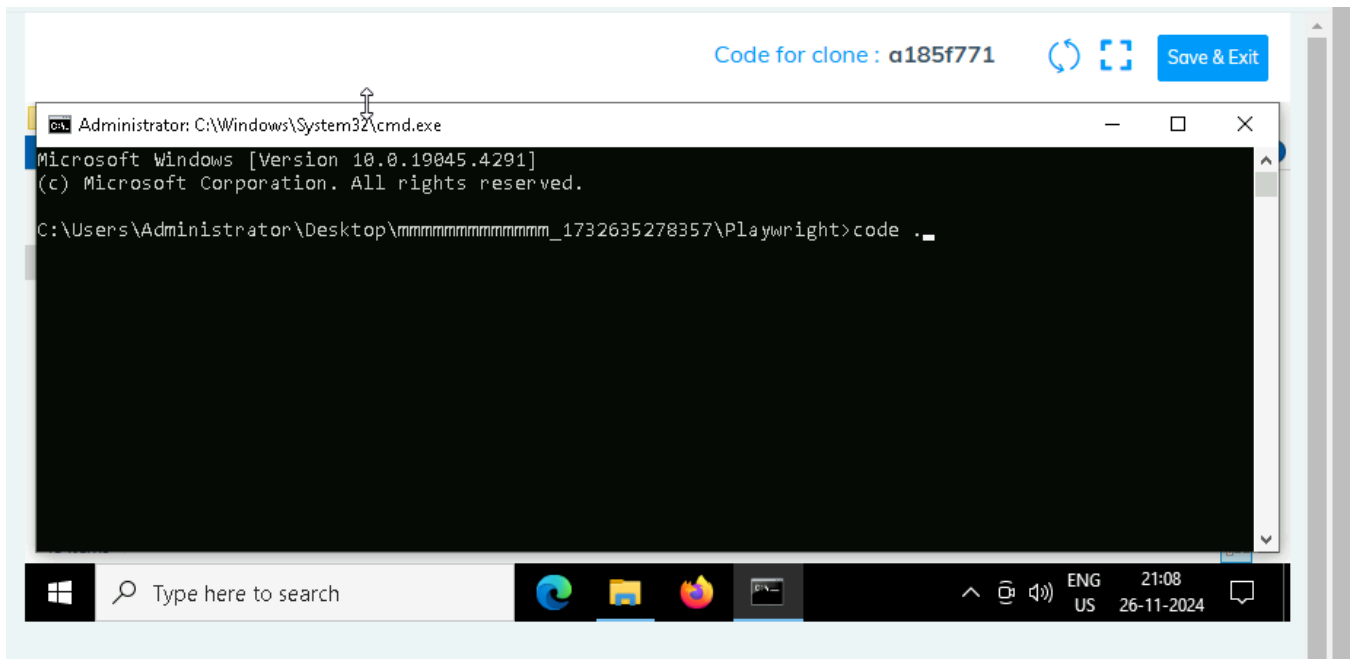
Mymedic automation using playwright

3. Open command prompt with it's location and use below command:

code .

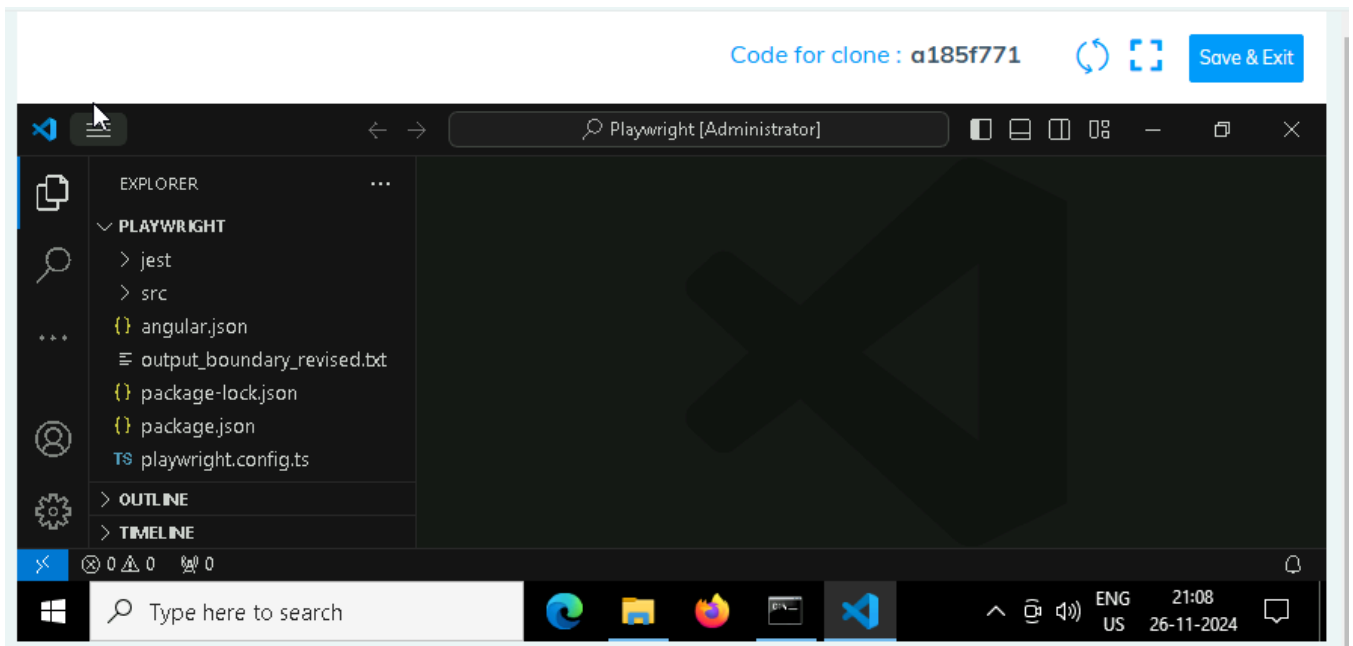


Mymedic automation using playwright



4. Once VsCode is open. Please open it's terminal by:

Mymedic automation using playwright



5. Install all dependencies using

npm install

6. Install playwright

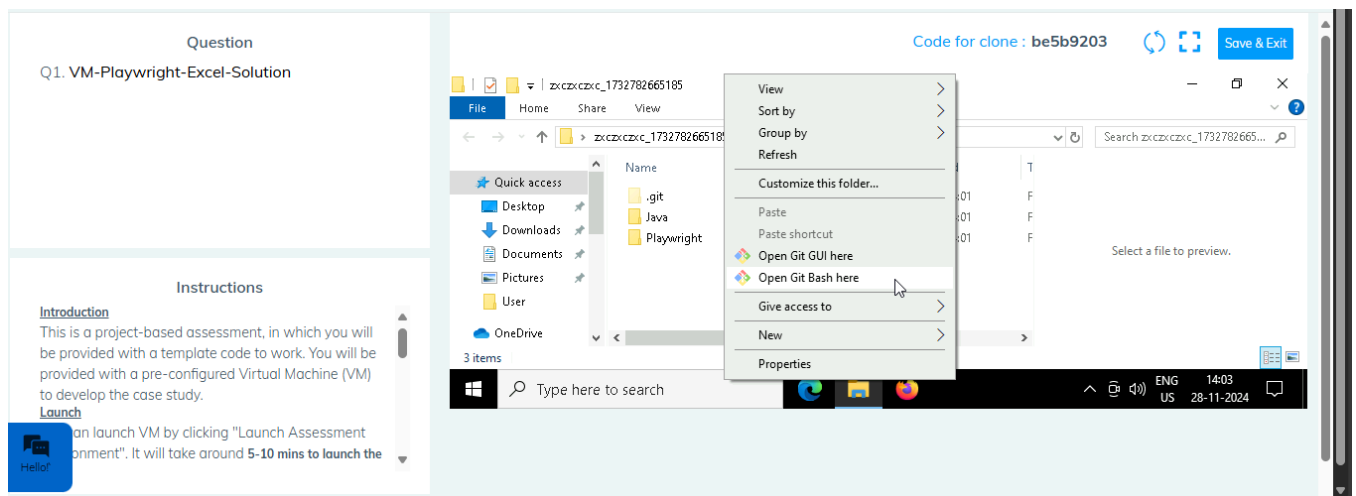
npx playwright install

7. Run the Tests:

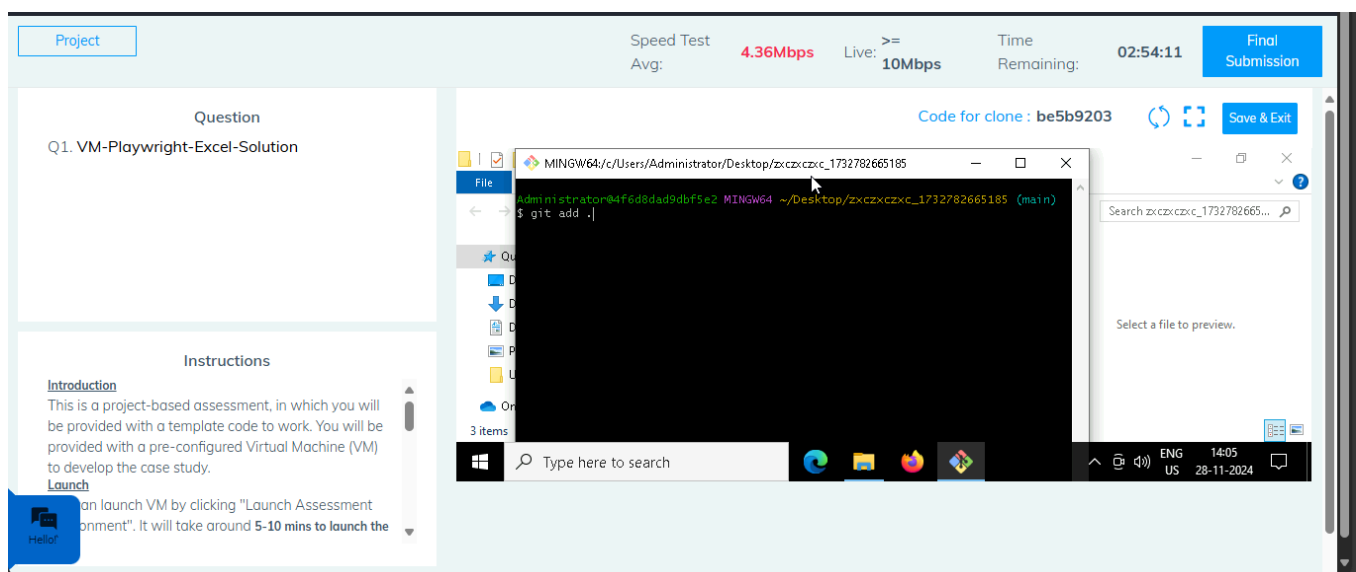
npm playwright test ./src/tests/PL1_testcases/yaksha.spec.ts

8. Once you have executed the test cases. Now it is must to push your code to git. For this, please go inside the folder created on desktop with the email id you have used to login and then:

1. Open gitbash

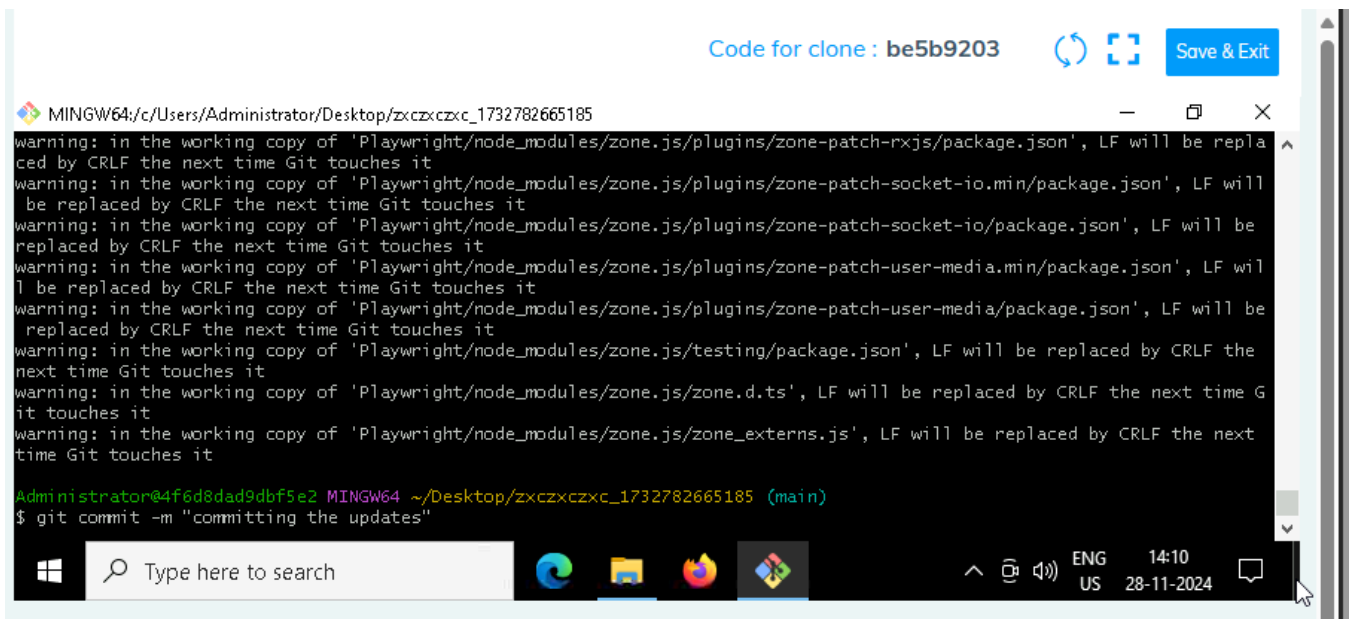


2. Add all files



Mymedic automation using playwright

3. Commit the changes



The screenshot shows a Windows terminal window with the title bar "MINGW64: c:/Users/Administrator/Desktop/zxczxczc_1732782665185". The terminal output displays several warnings about LF being replaced by CRLF in various package.json files. The user then enters the command `git commit -m "committing the updates"`. The terminal window has a taskbar at the bottom with the Windows logo, a search bar, and several application icons. The system tray shows the language as "ENG US", the time as "14:10", and the date as "28-11-2024".

```
Code for clone : be5b9203 [refresh] [copy] Save & Exit
```

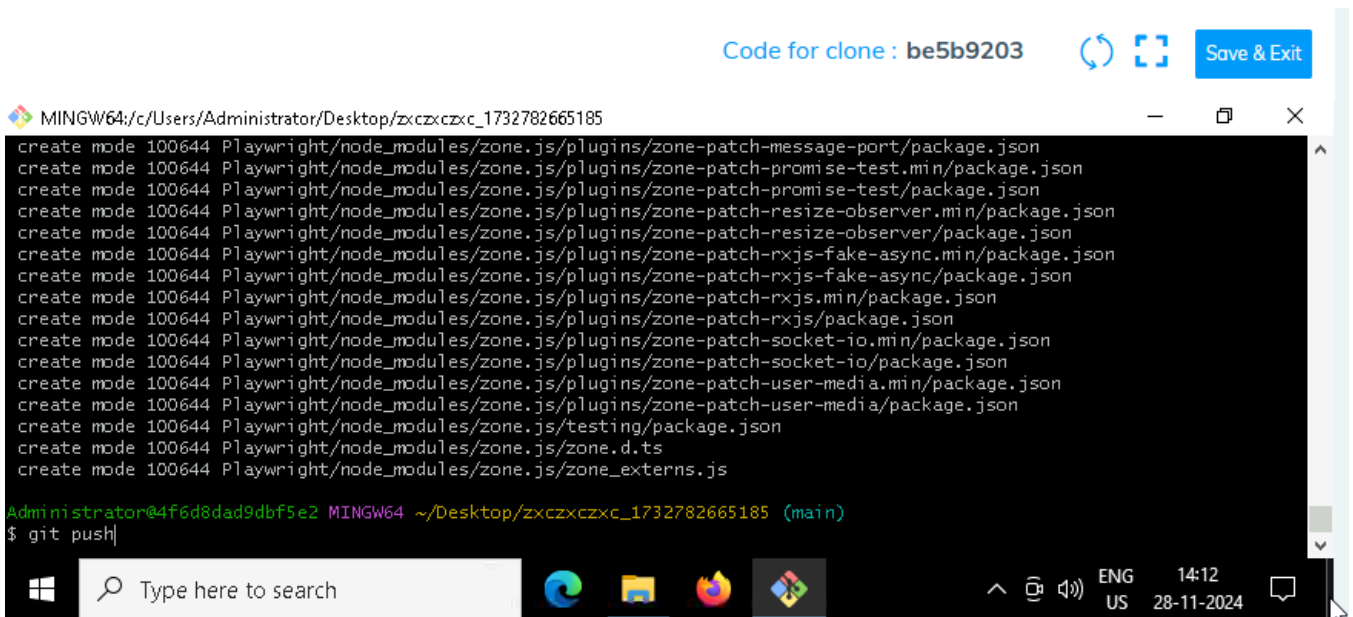
```
MINGW64: c:/Users/Administrator/Desktop/zxczxczc_1732782665185
```

```
warning: in the working copy of 'Playwright/node_modules/zone.js/plugins/zone-patch-rxjs/package.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Playwright/node_modules/zone.js/plugins/zone-patch-socket-io.min/package.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Playwright/node_modules/zone.js/plugins/zone-patch-socket-io/package.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Playwright/node_modules/zone.js/plugins/zone-patch-user-media.min/package.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Playwright/node_modules/zone.js/plugins/zone-patch-user-media/package.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Playwright/node_modules/zone.js/testing/package.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Playwright/node_modules/zone.js/zone.d.ts', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Playwright/node_modules/zone.js/zone_externs.js', LF will be replaced by CRLF the next time Git touches it

Administrator@4f6d8dad9dbf5e2 MINGW64 ~/Desktop/zxczxczc_1732782665185 (main)
$ git commit -m "committing the updates"
```

Windows taskbar: Type here to search, [icons], ENG US, 14:10, 28-11-2024

4. Push the changes



The screenshot shows a Windows terminal window with the title bar "MINGW64: c:/Users/Administrator/Desktop/zxczxczc_1732782665185". The terminal output displays a list of files being created, each with a mode of 100644. The user then enters the command `git push`. The terminal window has a taskbar at the bottom with the Windows logo, a search bar, and several application icons. The system tray shows the language as "ENG US", the time as "14:12", and the date as "28-11-2024".

```
Code for clone : be5b9203 [refresh] [copy] Save & Exit
```

```
MINGW64: c:/Users/Administrator/Desktop/zxczxczc_1732782665185
```

```
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-message-port/package.json
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-promise-test.min/package.json
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-promise-test/package.json
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-resize-observer.min/package.json
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-resize-observer/package.json
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-rxjs-fake-async.min/package.json
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-rxjs-fake-async/package.json
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-rxjs.min/package.json
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-rxjs/package.json
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-socket-io.min/package.json
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-socket-io/package.json
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-user-media.min/package.json
create mode 100644 Playwright/node_modules/zone.js/plugins/zone-patch-user-media/package.json
create mode 100644 Playwright/node_modules/zone.js/testing/package.json
create mode 100644 Playwright/node_modules/zone.js/zone.d.ts
create mode 100644 Playwright/node_modules/zone.js/zone_externs.js

Administrator@4f6d8dad9dbf5e2 MINGW64 ~/Desktop/zxczxczc_1732782665185 (main)
$ git push
```

Windows taskbar: Type here to search, [icons], ENG US, 14:12, 28-11-2024