YAKSHAHEALTHAPP 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:

1. **Login/Registration**: For user authentication.
2. **Dashboard**: Main page displaying patient records.
3. **Patient Records**: Add, edit, delete, and view records.
4. **Appointment Scheduling**: Manage patient appointments.
5. **Search/Filter**: Filter records by various parameters (e.g., doctor, date).

Project Information:

- **Use Case**: Aimed at simplifying EMR management, enhancing healthcare record accessibility, and enabling easy interaction with patient data for healthcare providers
-

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

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

Playwright Automation Test Cases

Test Case No.	Test Case Name	Preconditions	Steps	Expected Result
1	**Page Object Model (POM)**	Page objects for elements like login button and input fields are defined at https://healthapp.yaksha.com/Home/Index#/.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/. 2. Use POM structure to locate elements. 3. Perform actions like entering credentials (e.g., Username=admin, Password=pass123) and submitting the login form.	Login is successful, and the user is redirected to the dashboard.
2	**Modular Approach for Business Logic**	Reusable functions for managing patient data (e.g., add, edit, delete) are created for the registration page at https://healthapp.yaksha.com/Home/Index#/Patient/RegisterPatient/BasicInfo	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Patient/RegisterPatient/BasicInfo 2. Use a function (e.g., addPatient(name, age, etc.)) to add a patient (e.g., Name=John, Age=45, Doctor=Dr. Smith). 3. Validate patient data in the list.	Patient record for John is successfully added and visible in the patient list.
3	**Data-Driven Testing**	An external dataset (e.g., Excel file with patient details) is prepared for input testing.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/. 2. Load data from the file (e.g., Name=Jane, Age=30, Doctor=Dr. Brown). 3. Add records using the data. 4. Verify the records are correctly added to the system.	All records from the dataset are successfully added, and their details match the input.
4	**Apply Assertions for Validation**	Assertions are defined for required fields (e.g., Username, Password) and optional fields (e.g., Remember Me).	1. Navigate to the login page at https://healthapp.yaksha.com/Home/Index#/. 2. Use hard assertions to confirm the presence and functionality of mandatory fields (Username, Password). 3. Use soft assertions to test optional fields like the Remember Me checkbox. 4. Submit the login form with valid credentials and validate navigation.	Assertions confirm the presence of mandatory fields, and optional fields like Remember Me toggle correctly.
5	**Execution Status & Test Reporting**	Test reporting functionality is configured in Playwright.	1. Open the terminal or IDE where the Playwright framework is configured. https://healthapp.yaksha.com/Home/Index#/. 2. Generate a detailed report with test case status (e.g., Login=Pass, Add Patient=Fail). 3. Analyze results for any failed tests. 2. Run all test cases for the application by executing the Playwright test command (e.g., npx playwright test). 3. Wait for test execution to complete. 4. Generate a test execution report, showing the pass/fail status of each test case.	A detailed test report is generated, listing the pass/fail status of all test cases.
6	**Error Handling & Logging**	Logging is enabled in Playwright's test configuration.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/. 2. Perform an invalid operation, such as entering incorrect login credentials (e.g., Username=invalid_user, Password=wrong_pass).	Errors like "Invalid username" are logged with sufficient details for debugging.

Mymedic automation using playwright

Test Case No.	Test Case Name	Preconditions	Steps	Expected Result
			3. Validate that an error message is displayed. 4. Capture the error and log details, including the input, timestamp, and error message.	
7	**Locator Strategies with XPath & CSS**	User is logged in at https://healthapp.yaksha.com/Home/Index#/.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/. 2. Use XPath to locate dynamic elements, such as the search bar (//input[@name='searchField']). 3. Use CSS selectors for static elements, like buttons (button.submit). 4. Perform actions on these elements, such as entering text or clicking buttons. 5. Validate the results of these actions.	Elements are accurately located using XPath and CSS selectors. Actions like entering text into the search bar or clicking the submit button work as expected, and the correct results are displayed.
8	**Element Spying & Verification**	User is logged in, and Playwright inspection tools are enabled.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/. 2. Use Playwright's inspection tools (e.g., page.locator) to spy on element properties like id, class, and textContent. 3. Verify these attributes match the expected values (e.g., ID=usernameField, Class=input-text). 4. Validate the visibility and interactability of these elements.	All inspected elements have correct attributes and are visible and interactable. For example, usernameField has the expected ID and is editable without errors.
9	**Manage Web Element Interactions**	The Billing page at https://healthapp.yaksha.com/Home/Index#/Billing/BillReturnRequest is accessible, and multiple fiscal year options are available in the dropdown.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Billing/BillReturnRequest. 2. Locate the Fiscal Year dropdown. 3. Click the dropdown to display available options (e.g., 2022-2023, 2023-2024). 4. Select a fiscal year (e.g., 2023-2024). 5. Verify that the selection updates relevant data or UI (e.g., records for the selected fiscal year are displayed).	The Fiscal Year dropdown is functional. Selecting a year (e.g., 2023-2024) Dropdown selections and other interactions like checkbox toggles update the UI and related data as expected.
10	**Implement Wait Strategies**	The Bill Review page at https://healthapp.yaksha.com/Home/Index#/ClaimManagement/BillReview is accessible, and the page contains dynamic loading elements like spinners.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/ClaimManagement/BillReview. 2. Apply explicit waits for specific elements, such as loading spinners or tables, to ensure they are fully loaded. 3. Validate that all required elements (e.g., Review Insurance Bill List table) are visible and functional after loading completes.	Wait strategies prevent timeouts, and content loads successfully within the specified time.

Test Case No.	Test Case Name	Preconditions	Steps	Expected Result
11	**Switching Between Pages & Windows**	The user is logged in, and both the Bill Review page (https://healthapp.yaksha.com/Home/Index#/ClaimManagement/Bill	1. Navigate to https://healthapp.yaksha.com/Home/Index#/ClaimManagement/BillReview. 2. Validate that the Bill Review page loads	Navigation between the Bill Review page and Dashboard works seamlessly. The pages load fully, and no errors

Test Case No.	Test Case Name	Preconditions	Steps	Expected Result
		Review) and the Dashboard (https://healthapp.yaksha.com/Home/Index#/Dashboard) are accessible.	successfully. 3. Click on the Dashboard navigation link/button. 4. Validate redirection to https://healthapp.yaksha.com/Home/Index#/Dashboard. 5. Navigate back to the Bill Review page using the navigation menu or browser back button.	occur during navigation.
12	**Automate Complex Data Input**	User is logged in and on the The Create Appointment page at https://healthapp.yaksha.com/Home/Index#/Appointment/CreateAppointment is accessible, and valid patient and doctor data exists in the system.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Appointment/CreateAppointment. 2. Search for an existing patient (e.g., Jane Doe) in the patient search bar. 3. Fill in the appointment details: - Date: 2023-11-20 - Time: 11:00 AM - Doctor: Dr. Adams. 4. Save the appointment by clicking the Create Appointment button. 5. Verify that the appointment is created and appears in the appointment list.	The appointment for Jane Doe is successfully created with the correct details: Date=2023-11-20, Time=11:00 AM, and Doctor=Dr. Adams. The appointment appears in the appointment list.
13	**File Upload/Download Handling**	The Patient Registration Report page at https://healthapp.yaksha.com/Home/Index#/Reports/Patient/Patient/RegistrationReport is accessible, and patient registration data exists in the system.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Reports/Patient/Patient/RegistrationReport. 2. Select the desired filters for the report (e.g., Date Range: 2023-11-01 to 2023-11-12). 3. Click the Generate Report button. 4. Validate that the report displays patient registration data for the selected date range. 5. Export the report (e.g., as a PDF or Excel) and verify its contents.	The Patient Registration Report is generated successfully, displaying accurate data for the selected filters. The exported report (PDF/Excel) matches the displayed data.
14	**Parallel Test Execution**	Playwright is configured for parallel execution.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/. 2. Run tests like Login, Search, and Add Record simultaneously. 3. run in different bowser check the execution time 3. Monitor execution time and validate test results.	All tests run in parallel, reducing total execution time, and results are validated without conflicts.
15	**Verify Add/Remove Elements**	Add/remove functionality is enabled for list elements at https://healthapp.yaksha.com/Home/Index#/Settings/DepartmentsManage/Department is accessible, and the user has permissions to manage departments.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Settings/DepartmentsManage/Department. 2. Click the Add Department button. 3. Fill in the department details: - Name: Cardiology - Code: CARD001 - Description: Handles heart-related cases. 4. Save the new department and validate its presence in the department list. 5. Locate an existing department (e.g., Neurology) and click the Remove button. 6. Validate that the department is no longer	The new department (Cardiology) is successfully added and appears in the department list. The removed department (e.g., Neurology) is no longer displayed in the list.

Test Case No.	Test Case Name	Preconditions	Steps	Expected Result
			displayed in the list.	
16	**Admin Login with Basic Auth**	Admin credentials (e.g., Username=admin, Password=pass123) are valid, and login functionality is accessible at https://healthapp.yaksha.com/Home/Index#/.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/. 2. Enter admin credentials and submit the login form. 3. Verify access to admin features like Reports.	Admin login is successful, granting access to restricted features (e.g., Reports).
17	**JavaScript Alert Handling**	The form page at https://healthapp.yaksha.com/Home/Index#/Patient/RegisterPatient/BasicInfo is accessible, and JavaScript alerts are triggered for invalid data submission.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Patient/RegisterPatient/BasicInfo. 2. Fill in invalid data in the registration form fields: - Name: Leave blank - Age: Enter a negative value (e.g., -5). 3. Click the Save or Submit button. 4. Handle the JavaScript alert triggered due to invalid input. 5. Verify the alert text (e.g., Invalid age entered) and dismiss the alert.	displayed with the correct Invalid age entered). The d, and the form remains on rrection.
18	**Secure File Download Verification**	Secure file download functionality is enabled for reports at https://healthapp.yaksha.com/Home/Index#/Reports.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Reports. 2. Download a secure file (e.g., report.pdf). 3. Verify file content and access control (e.g., login required).	Secure file downloads (e.g., report.pdf) are completed successfully with valid content and access control.

Test Case No.	Test Case Name	Preconditions	Steps	Expected Result
19	**Shadow DOM Element Handling**	The Appointment Details Report page at https://healthapp.yaksha.com/Home/Index#/Vaccination/Reports/AppointmentDetailsReport is accessible, and data exists for both "New" and "Follow-Up" appointment types.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Vaccination/Reports/AppointmentDetailsReport. 2. Locate the Appointment Type filter dropdown. 3. Select the New option and click the Generate Report button. 4. Validate that the report displays only "New" appointments. 5. Change the filter to Follow-Up and regenerate the report. 6. Validate that the report displays only "Follow-Up" appointments.	The Appointment Details Report accurately filters and displays data based on the selected appointment type ("New" or "Follow-Up" Elements inside the Shadow DOM are accessible, and interactions like updates or selections are reflected accurately.
20	**Verify Before/ After**	The Patient Registration page at https://healthapp.yaksha.com/Home/Index#/Patient/RegisterPatient/BasicI	1. Configure a before hook to navigate to the Patient Registration page (https://healthapp.yaksha.com/Home/Index	Hooks execute The before hook successfully initializes the test environment by navigating to the

Test Case No.	Test Case Name	Preconditions	Steps	Expected Result
	Hooks	nfo is accessible, and test hooks are configured.	#/Patient/RegisterPatient/BasicInfo). 2. Configure an after hook to clean up test data after execution. 3. Execute the test: - Add a new patient with details: - Name: Jane Doe - Age: 30 - Address: 123 Health Street. - Save the patient record. 4. Validate that the patient is added successfully. 5. Ensure the after hook deletes the test patient data after the test completes.	registration page. The patient Jane Doe is added successfully during the test, and the after hook cleans up the test data, leaving no residual test records.
21	**Advanced Locator Strategy**	Attribute-based locators (e.g., data-id) are defined for the Patient Search page at https://healthapp.yaksha.com/Home/Index#/Doctors/OutPatientDoctor/NewPatient.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Doctors/OutPatientDoctor/NewPatient. 2. Use locators to identify the search bar. 3. Enter a search term (e.g., John Doe) and validate the results.	Attribute-based locators correctly identify elements, and the search results for John Doe are accurate and displayed as expected.
22	**Context Menu Interaction**	The Department Management page at https://healthapp.yaksha.com/Home/Index#/Settings/DepartmentsManage/Department is accessible, and departments exist in the system.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Settings/DepartmentsManage/Department. 2. Locate a department record (e.g., Cardiology). 3. Right-click on the department to open the context menu. 4. Select an action from the context menu (e.g., Edit Department). 5. Validate that the corresponding action (e.g., edit form) is triggered successfully..	The context menu is displayed upon right-clicking a department record. Selecting an action (e.g., Edit Department) triggers the corresponding functionality (e.g., opening the edit form).
23	**Dynamic Controls Verification**	The In-Patient List page at https://healthapp.yaksha.com/Home/Index#/Nursing/InPatient/InPatientList is accessible, with valid bed availability and ward data..	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Nursing/InPatient/InPatientList. 2. Locate a patient record (e.g., John Doe). 3. Allocate a bed to the patient using the Allocate Bed button. 4. Click the Consumption button and log items used (e.g., medication, supplies). 5. Perform a Ward Transfer for the patient to a different ward (e.g., transfer from General Ward to ICU). 6. Validate that all actions (bed allocation, consumption log, and ward transfer) are reflected correctly in the patient record.	The bed is allocated successfully, the consumption items are logged accurately, and the ward transfer updates the patient's record without errors.
24	**Entry/Exit Intent Ad**	Entry/Exit intent ads are configured on the Reports page at https://healthapp.yaksha.com/Home/Index#/Reports.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Reports/Dashboard. 2. Observe for any entry intent ads displayed	Entry intent ads display upon page load. Exit intent ads trigger when the user moves the cursor toward the browser's top. Ads are

Test Case No.	Test Case Name	Preconditions	Steps	Expected Result
	Handling		immediately upon page load. 3. Simulate exit intent by moving the cursor toward the top of the browser window. 4. Validate that the exit intent ad is displayed. 5. Close the ad using the close (X) button. 6. Confirm that the ad is dismissed and the page remains functional.	dismissible, and the page remains functional.
25	**Floating Menu Accessibility**	A floating menu is available on the Reports page at https://healthapp.yaksha.com/Home/Index#/Reports.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Reports. 2. Scroll down the page. 3. Verify the floating menu remains accessible and functional throughout.	The floating menu stays visible and functional during scrolling, ensuring smooth navigation.
26	**Infinite Scroll Verification**	The Patient Search page at https://healthapp.yaksha.com/Home/Index#/Patient/SearchPatient is accessible, and patient records are configured to load dynamically during scrolling.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Patient/SearchPatient. 2. Enter search criteria (e.g., John). 3. Scroll to the bottom of the results list to trigger the loading of additional patient records. 4. Validate that new patient records are loaded dynamically without refreshing the page. 5. Repeat scrolling multiple times to verify seamless loading of additional records.	Infinite scrolling dynamically loads new patient records without errors or delays. The total number of displayed records increases progressively with each scroll.

EXPECTATIONS:

Learners will gain experience in building strongly-typed applications using Angular.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

Mymedic automation using playwright

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. **Set up Playwright** by installing it in the project:

   ```
   npm install --save-dev playwright
   ```

2. **Create Test File** for each test case or combine them into one file for easier execution, e.g., healthapp-tests.spec.ts.

3. **Run the Tests**:

   ```
   npx playwright test
   ```