YAKSHA HEALTHAPP WITH JAVASCRIPT AND CYPRESS

Yaksha health app test automation
using cypress

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:** Cypress (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

Test Case No.	Test Case Name	Precondition	Steps	Expected Result
1	Verify Login with Valid Credentials	User is on the login page https://healthapp.yaksha.com	1. Navigate to https://healthapp.yaksha.com/Home/Index#/. 2. Enter username as admin. 3. Enter password as pass123. 4. Click the login button.	The user should be successfully logged in and redirected to the dashboard or homepage after login.

Test Case No.	Test Case Name	Precondition	Steps	Expected Result
2	Verify Page Navigation and Load Time for Billing Counter	User logged in	1. Navigate to /Utilities/ChangeBillingCounter. 2. Measure page load time. 3. Assert that page loads within acceptable time.	The billing counter page should load within the acceptable time frame.
3	Patient Search with Valid Data	User logged in and on Patient Search page	1. Navigate to /Appointment/PatientSearch. 2. Enter valid patient data. 3. Submit search and verify results.	The correct patient search results should be displayed.
4	Activate Counter in Dispensary	User logged in and on Dispensary page	1. Navigate to /Dispensary/ActivateCounter. 2. Click "Activate Counter". 3. Verify success message.	The counter should be successfully activated, and a confirmation message should appear.
5	Purchase Request List Load	User logged in	1. Navigate to /ProcurementMain/PurchaseRequest/PurchaseRequestList. 2. Verify that the purchase request list loads properly with no missing elements.	The purchase request list should load with all the relevant data displayed.
6	Lab Dashboard Data Validation	User logged in and on Lab Dashboard page	1. Navigate to /Lab/Dashboard. 2. Verify that the dashboard data (e.g., number of tests) is displayed and matches expected values.	The lab dashboard should display accurate and up-to-date data.
7	Handle Alert on Billing Counter	User logged in and on Billing Counter page	1. Navigate to /Utilities/ChangeBillingCounter. 2. Perform an action that triggers an alert. 3. Handle the alert using Cypress dialog handling.	The alert should be handled successfully without causing test failure.
8	Data-Driven Testing for Patient Search	User logged in and Patient search test data in excel	1. Read patient search data from an external excel file. 2. Use data to perform search for multiple patients. 3. Validate search results for each patient.	Patient search should work correctly for all data sets from the excel file.
9	Error Handling and Logging in Purchase Request List	User logged in	1. Navigate to /ProcurementMain/PurchaseRequest/PurchaseRequestList. 2. Intentionally cause an error by entering incorrect filters. 3. Log errors and exceptions.	Errors should be logged with detailed information without disrupting test flow.
10	Keyword-Driven Framework for Appointment Search	User logged in	1. Create keywords like SearchPatient, VerifyResults. 2. Use these keywords to perform the search operation. 3. Assert results based on keyword output.	The keyword-driven test should successfully search for a patient and verify results.
11	Modular Script	User logged in	1. Create a reusable function for patient search.	The reusable

Test Case No.	Test Case Name	Precondition	Steps	Expected Result
	for Patient Search		2. Call this function in multiple tests to search for different patients. 3. Assert search results for each test case.	function should streamline the test and return correct results for each patient.
12	Verify Assertion for Counter Activation	User logged in and on Dispensary page	1. Navigate to /Dispensary/ActivateCounter. 2. Assert with expect that the counter is activated. 3. Perform soft assertion for validation message.	The counter should be activated, and the success message should appear as expected.
13	Test Execution Reporting for Billing Counter Activation	User logged in	1. Execute test for billing counter activation. 2. Capture and generate test execution report with status. 3. Validate report for pass/fail.	The test execution report should correctly reflect the status of the billing counter activation test.
14	Verify Locator Strategy for Appointment Search	User logged in and on Appointment Search page	1. Use CSS selectors to locate search bar and filters. 2. Perform a search and assert that the correct patient list is displayed.	The correct locators should be used, and the patient list should be accurately displayed.
15	Element Inspection and Handling Alerts in Lab Dashboard	User logged in and on Lab Dashboard page	1. Navigate to /Lab/Dashboard. 2. Inspect elements on the page using Cypress Inspector. 3. Trigger an alert and handle it.	Elements should be correctly identified, and the alert should be handled without issues.
16	Navigation Exception Handling on Dispensary Page	User logged in and on Dispensary page	1. Navigate to /Dispensary/ActivateCounter. 2. Simulate a timeout or slow page load. 3. Handle the navigation exception and retry.	Navigation exceptions should be handled gracefully, and the page should load after retrying.
17	Web Element Handling for Dropdowns in Purchase Request	User logged in and on Purchase Request page	1. Navigate to /ProcurementMain/PurchaseRequest/PurchaseRequestList. 2. Select a filter from a dropdown menu. 3. Assert that the correct list is displayed.	The correct data should be displayed based on the selected filter from the dropdown menu.
18	Form Authentication and Error Messages	User is on the login page	1. Navigate to the login page. 2. Submit the login form with incorrect credentials. 3. Verify that the error message is displayed.	The proper error message should be displayed when incorrect credentials are entered.
19	Verify Handling	User is logged	1. Navigate to https://healthapp.yaksha.com/Home/Index#/SSU/PatientList.	The test should

Test Case No.	Test Case Name	Precondition	Steps	Expected Result
	of Frames on Patient List Page	in and navigates to the Patient List page.	2. Identify and switch to the frame containing the patient list. 3. Perform actions like filling in the search form or clicking a button within the frame. 4. Assert that the correct elements are interactable within the frame.	successfully locate the frame, interact with elements inside it, and perform the expected actions like searching for a patient or clicking buttons.
20	Verify Handling of Tabs for Billing Reports and Other Reports	User is logged in and navigates to the Reports section.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Reports. 2. Click on the link that opens Billing Reports in a new tab. 3. Switch to the new tab and wait for it to load. 4. Interact with elements on the Billing Reports tab (e.g., verify report data). 5. Repeat for other reports like Appointment, Radiology, Lab, Doctors, and Patient reports.	The test should successfully switch between tabs, interact with elements on each report tab, and verify that the correct report data is displayed.
21	Verify Handling of Popups for Changing Billing Counter	User is logged in and navigates to the Billing Counter page.	1. Navigate to https://healthapp.yaksha.com/Home/Index#/Utilities/ChangeBillingCounter. 2. Perform an action that triggers a popup (e.g., selecting a counter). 3. Capture and handle the popup using Cypress dialog handling. 4. Accept or dismiss the popup.	The popup should be successfully captured and handled. The test should either accept or dismiss the popup as required.

EXPECTATIONS:

- Learners will gain experience in writing reliable, maintainable automated tests using Cypress.
- They'll learn how to interact with web elements, perform assertions for error prevention, and validate overall application functionality

With **Cypress**, 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

Yaksha health app test automation
using cypress

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 Cypress** by installing it in the project:

   npm install cypress --save-dev

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

3. **Run the Tests**:

   Open Cypress Test Runner
   If you want to run tests interactively in the Cypress Test Runner, use the following command:

   npx cypress open

   This will open the Cypress Test Runner GUI. From here, you can:
   - **Select the testing environment (E2E Testing).**
   - **Click on your test file (e.g., health_app.spec.js) to run the test interactively.**