**JioMart Web Automation Testing Project**

**Overview**

This project is an end-to-end automated testing suite for the JioMart web application. It uses **Playwright** framework with **TypeScript** to implement functional UI tests, structured via the Page Object Model (POM) pattern for maintainability and reusability.

**Project Structure**

* **Page Objects**: Encapsulate UI elements and actions for different pages such as HomePage, LoginPage, CartPage, SearchResultsPage, ProfilePage, WishlistPage, CategoryPage.
* **Test Specs**: Located in the jiomart.spec.ts file, covering various user flows and functional validations.
* **Fixtures**: Custom Playwright fixtures are defined to manage page instances and browser context.
* **Config**: Configuration settings, including URLs and test data, are managed centrally for flexibility.
* **GitHub Workflow**: Continuous Integration configured to run tests on each push or pull request, ensuring code quality and test stability.

**Key Components**

**1. Page Object Model**

Each page class contains methods to interact with the UI and verify key states, e.g.:

1. **HomePage.ts**

Navigation, searching products, changing delivery location, verifying page elements.

**Locator Method Used**

| **Method Used** | **Example from code** |
| --- | --- |
| page.locator() | page.locator('#btn\_minicart') |
| page.getByText() | page.getByText('Agree') |
| page.getByRole() | page.getByRole('heading', { name: 'Contact Us' }) |

**Functionalities & Related Test Cases**

* + Homepage Navigation & Verification of url and title
  + Login & User Profile
  + Location Management
  + Cart Functionality
  + Product Search & Shopping List
  + Category Navigation & Validation
  + Wishlist Management
  + UI Components Visibility Checks
  + Navigation Link Redirection

**Expect Assertions Used**

| **Assertion** | **Assertion Description** |
| --- | --- |
| expect(this.page).toHaveURL('https://www.jiomart.com/') | Verifies current URL matches expected one. |
| expect(this.page).toHaveTitle("JioMart: India's online shopping destination") | Validates page title matches expected string. |
| expect(this.profileIcon).toBeVisible() | Checks that a UI element is present and visible to the user. |
| expect(this.deliveryLoc).toContainText('691501') | Ensures the element’s text content includes the expected string. |
| expect(matchFound).toBeTruthy() | Verifies that a boolean expression evaluates to true. |
| expect(this.page.url()).toContain(element) | Ensures the URL contains specific keywords, useful for search validations. |

1. **LoginPage.ts**

Inputting login details, OTP handling, and login verification.

**Locator Method Used**

| **Method Used** | **Selector Type** |
| --- | --- |
| page.locator() | ID, Attribute selectors |
| page.getByText() | Text-based locator |

**Expect Assertions Used**

| **Assertion** | **Description** |
| --- | --- |
| expect(this.page).toHaveTitle('Retail Account') | Validates that the login page title is "Retail Account". |
| expect(this.otpHeading).toBeVisible() | Confirms that the OTP verification heading is visible after clicking continue. |

**Functionalities & Related Test Cases**

* Verify Login Page Load
* Enter Phone Number and Trigger OTP
* Wait for OTP Input
  + 1. **CartPage.ts**

Cart validations including empty cart verification.

**Locator Methods Used**

| **Method Used** | **Selector Type** |
| --- | --- |
| page.locator() | CSS selector (tag.class) |
| page.locator() | Attribute selector ([classname=...]) |

**Expect Assertions Used**

| **Assertion** | **Description** |
| --- | --- |
| expect(this.page).toHaveURL(...) | Validates that the user is on the cart page URL. |
| expect(this.cartHeading).toHaveText("My Cart") | Confirms that the heading on the cart page is correct. |
| expect(this.cartEmptyText).toBeVisible() | Checks that the "empty cart" message is shown. |

**Functionalities & Related Test Cases**

* URL Validation
* Heading Validation
* Empty Cart Check
  + 1. **SearchResultsPage.ts**

Product search results, add to cart and wishlist button validations.

**Locator Method Used**

| **Method Used** | **Selector Type** |
| --- | --- |
| page.locator() | CSS class selectors, attribute selectors |
| locator.nth() | Indexed locator to pick specific elements |
| locator.locator() | Nested locator within a parent element |

**Expect Assertions Used**

| **Assertion** | **Description** |
| --- | --- |
| expect(currentURL).toContain(config.item) | Verifies the current URL contains the search item keyword from config. |
| expect(visibleCount).toBe(productCount) | Confirms the number of visible Add to Cart buttons matches the number of product cards. |
| expect(this.addToCartBtns.nth(0)).toBeEnabled() | Ensures the first Add to Cart button is enabled (clickable). |
| expect(homePage.wishListBtns.nth(0)).toHaveAttribute("class", /selected/) | Verifies the heart icon changes class attribute to reflect wishlisting (selected state). |
| expect(this.outOfStockFilterBox).toBeVisible() | Confirms the out-of-stock filter checkbox is visible before clicking. |
| expect(this.appliedFilterContent).toHaveText("Include out of stock") | Validates the text of applied filter content matches "Include out of stock". |
| expect(found).toBe(true) | Asserts that at least one product with out-of-stock label is found after applying the filter. |

**Functionalities & Related Test Cases**

* Verify Search Results URL
* Check Visibility of Add to Cart Button
* Verify Count and Visibility of Add to Cart Buttons vs Product Cards
* Verify Add to Cart Button is Enabled
* Verify Wishlisting Changes Heart Icon Colour
* Apply Out of Stock Filter
* Verify Out of Stock Filter Application
  + 1. **ProfilePage.ts**

Profile editing, address management, navigation between user account sections.

**Locator Method Used**

| **Method Used** | **Selector Type** |
| --- | --- |
| page.locator() | Tag with attribute, class, nth-child, aria-label, CSS selectors |
| locator.getByText() | Text-based locator |
| page.getByRole() | Role-based locator with accessible name |

**Expect Assertions Used**

| **Assertion** | **Description** |
| --- | --- |
| expect(this.wishlistBtn).toBeVisible() | Checks that the Wishlist button is visible before clicking. |
| expect(this.editProfileHeading).toHaveText("Personal Info on JioMart") | Verifies the heading text on Edit Profile page. |
| expect(names[0]).toBe(config.firstName) | Confirms first name displayed matches expected first name. |
| expect(this.page).toHaveURL(...) | Validates the current page URL matches expected URLs for tabs (My Orders, My List, Coupons, etc.). |
| expect(addressBoxNumAfter).toBeGreaterThanOrEqual(addressBoxNum) | Confirms that the number of addresses after adding is greater or equal to before adding. |

**Functionalities & Related Test Cases**

* Wishlist Access
* Edit Profile Verification
* Edit User Name
* Verify Updated Name Display
* Navigate Tabs and Verify URLs
* Add New Address
* Edit Existing Address
* Navigate to PAN Card Information
* Verify Terms and Conditions Visibility
* Sign Out Functionality
* Payments Navigation
* Access JioMart Wallet Details
  + 1. **WishlistPage.ts**

**Locator Method Used**

| **Locator Name** | **Selector Explanation** |
| --- | --- |
| wishlistHeading | Finds the text element containing **"My Wishlist"** |
| productCard | Selects product cards under the wishlist heading container |
| productCardTitle | Targets the product title within the wishlist product list |

**Expect Assertions**

| **Assertion** | **Purpose** |
| --- | --- |
| expect(this.page).toHaveURL('https://www.jiomart.com/customer/wishlist') | Verifies the current page URL matches the Wishlist page URL |
| expect(this.wishlistHeading).toHaveText("My Wishlist") | Confirms the wishlist page heading text is correct |
| expect(prodNameFromWishlist.startsWith(productName)).toBe(true) | Checks the first product in the wishlist starts with the expected product name (partial match) |

**Functionalities**

* Verify User Is On Wishlist Page
* Verify Wishlist Heading
* Verify Item Present In Wishlist
  + 1. **CategoryPage.ts**

This class handles interactions and validations on the **Product Category Page** of the Jiomart site. It focuses on **filters**, **sort options**, and **product card validation**.

**Expect Assertions Used**

| **Assertion** | **Purpose** |
| --- | --- |
| expect(locator).toBeVisible() | Ensures that a UI element is displayed |
| expect(this.page).toHaveURL(...) | Confirms navigation didn't change after sorting |
| expect(currentPrice).toBeLessThanOrEqual(previousPrice) | Validates descending price order |
| expect(currentPrice).toBeGreaterThanOrEqual(previousPrice) | Validates ascending price order |

**Functionalities & Related Test Case Groups**

* Page Layout & Visibility Checks
* Filters visibility
* Subcategory section
* Sorting Functionality
* Default Sort – Popularity
* Price: High to Low Sorting
* Price: Low to High Sorting
* Discount High to Low Sorting

**Locator Methods Used**

| **Locator Method** | **Where Used** |
| --- | --- |
| page.locator() | Most locators |
| page.getByText() | Sorting buttons |

**2. Test Specifications**

Tests cover critical user journeys, including:

* Navigation flows (Home → Login, Home → Cart, Profile → Wishlist, etc.)
* User authentication with OTP
* Location update via pincode
* Product search and add-to-cart validations
* Wishlist operations with and without login
* Profile editing (name, address)
* Sorting and filtering products on category pages
* Viewing terms, coupons, wallet, and gift card details
* Sign-out functionality and verification of logged-out state
* UI element visibility and functional checks (dropdowns, logos, filters)

Tests are grouped logically with test.describe and configured to run in parallel for efficiency.

**3. Configuration and Utilities**

* Centralized config file storing test data like product lists for searches.
* Utility methods assist in reusable tasks and verification steps.
* Fixtures (jiomart.fixture.ts) to initialize page objects and context for each test.

**4. CI/CD Integration: GitHub Workflow**

* Automated test runs triggered on code pushes and pull requests.
* Ensures continuous validation of test cases.
* Helps maintain code quality and detect failures early.
* Workflow integrates with Playwright Test Runner and reports.

**Summary**

This project demonstrates a robust approach to web app testing by combining:

* Modern testing tools (Playwright + TypeScript)
* Page Object Model for scalable test design
* Comprehensive coverage of key user flows and edge cases
* Continuous Integration setup for reliable automated testing

It provides a solid foundation to ensure the quality of JioMart's user experience while enabling easy maintenance and future enhancements.