**Shoppers Stop Website Automation Report**

**Automation Framework: Playwright with JavaScript/TypeScript  
Framework: Playwright with TypeScript**

**Goal: Validate homepage load, navigation menus, banners, footer links, responsiveness, and product sections.**

**1. Verify homepage loads successfully**

* Navigates to homepage and checks URL and header visibility.

Code:

await page.goto('https://www.shoppersstop.com/');

await expect(page).toHaveURL(/shoppersstop.com/);

await expect(page.locator('header')).toBeVisible();

2. Verify main menu (Men, Women) is visible

* Checks if main category links "Men" and "Women" are visible.

Code:

const categories = ['Men', 'Women'];

for (const category of categories) {

await expect(page.locator(`a:has-text("${category}")`).first()).toBeVisible();

}

3. Check if promotional banners display and rotate

* Confirms visibility of promotional banners.

Code:

const banner = page.locator('.slick-track, .carousel-item');

await expect(banner).toBeVisible();

4. Clicking the logo redirects to homepage

* From a category page, clicks the logo and verifies homepage URL.

Code:

await page.goto('https://www.shoppersstop.com/category/kids');

await page.getByAltText('logo').first().click();

await expect(page).toHaveURL('https://www.shoppersstop.com/home');

5. Verify footer links work (About Us, Contact Us, Help/FAQs, Delivery Policy, Careers)

* Checks each footer link redirects correctly.

Code:

const footerLinks = ['ABOUT US', 'CONTACT US', 'HELP/FAQS', 'DELIVERY POLICY', 'CAREERS'];

for (const linkText of footerLinks) {

await page.locator(`footer a:has-text("${linkText}")`).click();

await expect(page).toHaveURL(new RegExp(linkText.toLowerCase().replace(/[ \/]/g, '-')));

await page.goBack();

}

6. Check “Help & Support” link in header

* Clicks "Help & Support" and verifies FAQ page loads.

Code:

await page.locator('header a:has-text("Help & Support")').click();

await expect(page.locator('text=Help/FAQ')).toBeVisible();

7. Test language or region switcher (if available)

* Selects alternate language/region if dropdown exists.

Code:

const langDropdown = page.locator('select.language-switcher');

if (await langDropdown.count() > 0) {

await langDropdown.selectOption({ index: 1 });

await expect(page).toHaveURL(/lang=/);

}

8. Verify site responsiveness

* Checks header visibility in different viewport sizes.

Code:

await page.setViewportSize({ width: 375, height: 667 }); // mobile

await expect(page.locator('header')).toBeVisible();

await page.setViewportSize({ width: 1024, height: 768 }); // desktop

await expect(page.locator('header')).toBeVisible();

9. Check nearby store functionality

* Clicks nearby store icon, selects city, and verifies directions page.

Code:

wait page.locator('button[aria-label="Store Locator"]').click();

await page.locator('select#city').selectOption('Mumbai');

await expect(page).toHaveURL(/store-locator/);

10. Verify product sections under WOMEN menu

* Hovers on WOMEN menu, clicks categories, and checks product listings.

Code:

await page.locator('a:has-text("WOMEN")').hover();

await page.locator('#category-women-1').click();

await expect(page.locator('h1.page-header')).toBeVisible();

**Module: Login and Logout Functionality**

**Overview**

The tests cover scenarios from user login with OTP, registration, error handling, session persistence, to logout verification.

**Test Cases and Details**

**31. Login with Valid Credentials**

**Purpose:** Verify that users can log in successfully with valid email and OTP.

**Steps:**

* Navigate to homepage.
* Click **Login**.
* Enter registered email address.
* Click **PROCEED**.
* Wait for **VERIFY OTP** button to enable.
* Click **VERIFY OTP**.

**Code snippet:**

await page.getByText('Login').click();

await page.locator('input[type="text"]').nth(1).fill('srinivaslakkoju2002@gmail.com');

await page.getByText('PROCEED').click();

const btnHandle = await page.locator('button:has-text("VERIFY OTP")').elementHandle();

if (btnHandle) {

await page.waitForFunction(btn => !btn.hasAttribute('disabled'), btnHandle);

await btnHandle.click();

} else {

throw new Error('VERIFY OTP button not found');

}

**32. Attempt Login with Wrong Password (OTP)**

**Purpose:** Ensure error message appears for invalid OTP during login.

**Steps:**

* Open login form.
* Enter email.
* Click **PROCEED**.
* Click **VERIFY OTP** without valid OTP.
* Verify error message *“Please enter a valid OTP”* is shown.

**Code snippet:**

await page.getByText('Login').click();

await page.locator('input[type="text"]').nth(1).fill('ajaytanneru07@gmail.com');

await page.getByText('PROCEED').click();

const btnHandle = await page.locator('button:has-text("VERIFY OTP")').elementHandle();

if (btnHandle) {

await page.waitForFunction(btn => !btn.hasAttribute('disabled'), btnHandle);

await btnHandle.click();

}

const errorMsg = await page.locator('.mt-1').nth(1).textContent();

expect(errorMsg).toBe('Please enter a valid OTP');

**33. Register a New User**

**Purpose:** Verify new user registration flow including OTP verification.

**Steps:**

* Click dropdown → **Sign up**.
* Enter name and phone number.
* Click **PROCEED**.
* Wait for and click **VERIFY OTP**.

**Code snippet:**

await page.getByAltText('chevron-down').first().click();

await page.getByText('Sign up').click();

await page.locator('#Name').fill('asdh jbdsj');

await page.locator('#Phone Number').fill('9398178316');

await page.getByText('PROCEED').click();

const btnHandle = await page.locator('button:has-text("VERIFY OTP")').elementHandle();

if (btnHandle) {

await page.waitForFunction(btn => !btn.hasAttribute('disabled'), btnHandle);

await btnHandle.click();

}

**34. Try Registering with Existing Email**

**Purpose:** Test system response when trying to register with an email already in use.

**Steps:**

* Open signup form.
* Fill name and phone number fields.
* Click **REGISTER NOW**.
* Attempt OTP verification.

**Code snippet:**

await page.getByAltText('chevron-down').first().click();

await page.getByText('Sign up').click();

await page.locator('#Name').fill('asdh jbdsj');

await page.getByLabel('Phone Number').fill('9398178316');

await page.locator('p').getByText('REGISTER NOW').click();

const btnHandle = await page.locator('button:has-text("VERIFY OTP")').elementHandle();

if (btnHandle) {

await page.waitForFunction(btn => !btn.hasAttribute('disabled'), btnHandle);

await btnHandle.click();

}

**Product Page Functionality**  
Goal: Validate product details, image browsing, add to cart/wishlist, size selection, delivery estimate, reviews, and navigation features.

**1. Verify product title, price, description, and stock**

* Checks if the title, price, and description elements are visible.

Code:

const title = page.locator('//a[@data-testid="product-card"]/div/following-sibling::div/div').first();

const price = page.locator('text=₹').first();

const description = page.locator('.mt9.line-clamp-2').first();

expect(title).toBeVisible();

expect(await price.textContent()).toContain('₹');

expect(description).not.toBeNull();

2. Check size and color selection functionality

* Verifies size and quantity options are visible.

Code:

const size = page.locator('button.inline-flex p div').nth(2);

const quantity = page.locator('p.pl-1').nth(3);

expect(size).toBeVisible();

expect(quantity).toBeVisible();

3. Browse through product images

* Clicks on the second image thumbnail in the image slider.

Code:

await page.locator('.slick-slider.slick-vertical div.slick-slide img').nth(1).click();

4. Click “Add to Cart” and check if cart updates

* Tries to click Add to Cart and waits for UI update.

Code:

const addToCart = page.locator('//div/button/p/div').first();

await addToCart.click();

await page.waitForTimeout(2000);

5. Click “Add to Wishlist” without login

* Adds product to wishlist and verifies login prompt appears.

Code:

await page.getByText('WISHLIST').first().click();

await expect(page.getByText('Sign in to Shoppers Stop')).toBeVisible();

6. Add to wishlist after login

* Logs in with email, waits for OTP button, then adds product to wishlist.

Code:

await page.getByText('Login').click();

await page.locator('input[type="text"]').nth(1).fill('your-email@example.com');

await page.getByText('PROCEED').click();

const btn = await page.locator('button:has-text("VERIFY OTP")').elementHandle();

await page.waitForFunction(btn => !btn.hasAttribute('disabled'), btn);

await btn.click();

await page.getByText('WISHLIST').first().click();

7. Remove from wishlist

* After logging in, clicks the "WISHLISTED" button to remove item.

Code:

await page.getByText('WISHLISTED').click();

8. Check wishlist is empty

* Navigates to wishlist page and verifies empty message.

Code:

await page.getByAltText('wish-list').first().click();

await expect(page.getByText('You have no wish-listed items yet')).toBeVisible();

9. Verify related products section appears

* Checks that “Similar Products” section is visible.

Code:

expect(await page.getByText('Similar').first()).toBeVisible();

expect(await page.getByText('Products').first()).toBeVisible();

10. Check for reviews/rating visibility

* Verifies that the "RATINGS & REVIEWS" section is visible.

Code:

expect(await page.getByText('RATINGS & REVIEWS').first()).toBeVisible();

11. Check estimated delivery on entering pincode

* Enters a pincode and checks if delivery estimate is shown.

Code:

const pincode = page.locator('input[placeholder="Enter your PIN code"]');

await pincode.fill('535002');

await page.keyboard.press('Enter');

await page.waitForTimeout(2000);

expect(await page.locator('text=Standard Estimated Delivery in').first()).toBeVisible();

12. Verify breadcrumb or product navigation section

* Verifies visibility of navigation buttons like “About The Brand” or “Product Details”.

Code:

expect(await page.locator('button:has-text("About The Brand")')).toBeTruthy();

expect(await page.getByText('button:has-text("Product Details")')).toBeTruthy();

**Search Functionality**  
  
Goal**:** Validate the product search, autocomplete, filters, and sorting features.

**1. Search for a valid product and verify results**

* Enters "T-shirt", presses Enter, and checks if results are shown.

Code:

await page.locator('input[placeholder="What are you looking for?"]').fill('T-shirt');

await page.keyboard.press('Enter');

await expect(page.locator('.grid.justify-between a').first()).toBeVisible();

**2. Search for an invalid product and verify “No Results” message**

* Enters gibberish and confirms the “No Results Found” message is visible.

Code:

await page.locator('input[placeholder="What are you looking for?"]').fill('asdhkjashd123');

await page.keyboard.press('Enter');

await expect(page.locator('div:has-text("No Results Found")')).toBeVisible();

**3. Test search with partial product name**

* Types part of a product name (e.g., "Sho") and checks for suggestions.

await page.locator('input[placeholder="What are you looking for?"]').pressSequentially('Sho', { delay: 100 });

await page.keyboard.press('Enter');

await expect(page.locator('.grid.justify-between a').first()).toBeVisible();

**4. Use filter by brand and check filtered results**

* Searches for "T-shirt", applies the brand filter for "Adidas", and verifies it.

await page.keyboard.press('Enter');

await page.locator('button:has-text("Brands")').click();

await page.locator('input[placeholder="Search"]').fill('Adidas');

await expect(page.locator('#Adidas')).toBeVisible();

await page.locator('#Adidas').click();

**5. Use filter by price and validate results**

* Filters for ₹0–₹500, then verifies all prices are within range.

Code:

await page.locator('label[for="0-500"]').click();

const prices = await page.locator('[data-testid="product-card"] .text-black').allTextContents();

prices.forEach(price => {

const amount = Number(price.replace('₹', '').replace(/,/g, '').trim());

if (amount >= 500) test.fail();

});

**6. Use sorting option (Low to High)**

* Sorts products by "Price Low to High" and checks the sorting label.

Code:

await page.locator('button[role="combobox"]').click();

await page.getByText('Price Low to High').click();

expect(await page.locator('.mr-10.text-sm').textContent()).toBe('Price Low to High');

**7. Check autocomplete suggestions in search bar**

* Types "shi" and verifies that suggestion text is shown.

Code:

await page.locator('.w-full.bg-transparent.text-sm').fill('shi');

await expect(page.locator('//p[@data-testid="suggested-search"]').first()).toBeVisible();

**8. Click on a product from suggestions**

* Clicks the first suggestion that appears after typing.

Code:

await page.locator('//p[@data-testid="suggested-search"]').first().click();

**9. Verify display of trending searches**

* Clicks the search input and checks if “Trending” section appears.

Code:

await page.locator('.w-full.bg-transparent.text-sm').click();

expect(await page.getByText('Trending')).toBeVisible();

**10. Clear search input and re-search**

* Searches for one product, clears it, then searches another and verifies results.

Code:

await page.locator('input[placeholder="What are you looking for?"]').fill('Shoes');

await page.keyboard.press('Enter');

await expect(page.locator('.grid.justify-between a').first()).toBeVisible();

await page.locator('input[placeholder="What are you looking for?"]').fill('');

await page.locator('input[placeholder="What are you looking for?"]').fill('Shirt');

await page.keyboard.press('Enter');

await expect(page.locator('.grid.justify-between a').first()).toBeVisible();

**Page: Cart Functionality**  
  
Goal: Validate adding multiple products to cart, size selection, cart update, and item count verification.

**1. Search and add the first product to the cart**

* Searches for "T-shirt", selects the first product, picks a size if available, and adds to cart.

Code:

await page.goto('https://www.shoppersstop.com/');

await page.locator('input[placeholder="What are you looking for?"]').fill('T-shirt');

const search = page.locator('.hidden.w-full.items-center input');

await search.click();

await page.keyboard.press('Enter');

await page.getByTestId('product-card').nth(0).click();

const sizeOption = page.locator('button.inline-flex p div').nth(2);

if (await sizeOption.isVisible()) await sizeOption.click();

const addToCart = page.locator('//div/button/p/div').first();

await addToCart.click();

await page.waitForTimeout(2000);

await page.goBack();

**2. Add the second product to the cart**

* Repeats selecting the first product and adding it to the cart.

Code:

await page.getByTestId('product-card').nth(0).click();

if (await sizeOption.isVisible()) await sizeOption.click();

await addToCart.click();

await page.waitForTimeout(2000);

**3. Open the cart and verify multiple items**

* Clicks on the cart icon and checks if at least 2 items are present.

Code:

await page.locator('img[alt="shopping-cart"]').click();

await page.waitForLoadState();

const cartItems = await page.locator('[data-testid="cart-item"]').count();

expect(cartItems).toBeGreaterThanOrEqual(2);