**Test Automation Strategy using Cypress**

1. **Confirm the Lowest Price in the Store**:

describe('Lowest Price Confirmation', () => {

it('Should confirm the lowest price is 7.99', () => {

cy.visit('/inventory.html');

cy.get('.inventory\_item\_price')

.invoke('text')

.then((pricesText) => {

const prices = pricesText.split('\n').map((price) => parseFloat(price.replace('$', '')));

const lowestPrice = Math.min(...prices);

expect(lowestPrice).to.equal(7.99);

});

});

});

1. **Test the Checkout Process**:

describe('Checkout Process', () => {

it('Should go through the checkout pages and complete the process', () => {

cy.visit('/inventory.html');

// Add items to the cart

cy.get('.btn\_inventory').eq(0).click(); // Add the first item

cy.get('.btn\_inventory').eq(1).click(); // Add the second item

// Go to the cart

cy.get('.shopping\_cart\_link').click();

// Start the checkout process

cy.get('.checkout\_button').click();

// Fill out checkout information

cy.get('#first-name').type('John');

cy.get('#last-name').type('Doe');

cy.get('#postal-code').type('12345');

// Continue to the next step

cy.get('.cart\_button').click();

// Finish the checkout

cy.get('.cart\_button').click();

// Verify the purchase is complete

cy.contains('THANK YOU FOR YOUR ORDER');

});

});

1. **Ensure the Order Item's Total is Showing Correctly**:

// Define a function to calculate the expected total

function calculateExpectedTotal() {

let total = 0;

cy.get('.cart\_item').each(($item) => {

const priceText = $item.find('.inventory\_item\_price').text();

const price = parseFloat(priceText.replace('$', ''));

total += price;

});

return total.toFixed(2); // Format total to two decimal places

}

describe('Order Item Total', () => {

it('Should verify the item total amount during checkout', () => {

cy.visit('/inventory.html');

// Add items to the cart

cy.get('.btn\_inventory').eq(0).click();// Add the first item

cy.get('.btn\_inventory').eq(1).click(); // Add the second item

// Go to the cart

cy.get('.shopping\_cart\_link').click();

// Calculate the expected item total based on added items

const expectedTotal = calculateExpectedTotal();

// Start the checkout process

cy.get('#checkout').click();

// Check that the item total matches the expected total

cy.get('.summary\_subtotal\_label').should('contain', expectedTotal);

});

});

1. **Ensure the Order Tax**:

// Define a function to calculate the expected total

function calculateExpectedTotal() {

let total = 0;

cy.get('.cart\_item').each(($item) => {

const priceText = $item.find('.inventory\_item\_price').text();

const price = parseFloat(priceText.replace('$', ''));

total += price;

});

return total.toFixed(2); // Format total to two decimal places

}

describe('Order Tax', () => {

it('Should verify the order tax is between 5% and 10%', () => {

cy.visit('/inventory.html');

// Add items to the cart

cy.get('.btn\_inventory').eq(0).click();// Add the first item

cy.get('.btn\_inventory').eq(1).click(); // Add the second item

// Go to the cart

cy.get('.shopping\_cart\_link').click();

// Calculate the expected item total and tax

const expectedTotal = calculateExpectedTotal();

const expectedTax = expectedTotal \* 0.05; // Assuming 5% tax

// Start the checkout process

cy.get('#checkout').click();

// Check that the tax is between 5% and 10%

cy.get('.summary\_tax\_label').invoke('text').then((taxText) => {

const actualTax = parseFloat(taxText.replace('$', ''));

expect(actualTax).to.be.at.least(expectedTax);

expect(actualTax).to.be.at.most(expectedTotal \* 0.10); // Assuming 10% tax

});

});

});

1. **Reusable Login and Logout Page Object Method**:

function login(username, password) {

cy.visit('/login.html');

cy.get('#user-name').type(username);

cy.get('#password').type(password);

cy.get('#login-button').click();

}

function logout() {

cy.get('#react-burger-menu-btn').click();

cy.get('#logout\_sidebar\_link').click();

}

describe('Login and Logout', () => {

it('Should log in and out successfully', () => {

login('your\_username', 'your\_password');

cy.contains('Products');

logout();

cy.contains('Login');

});

it('Should prevent access to /inventory.html if not logged in', () => {

cy.visit('/inventory.html');

cy.contains('Epic sadface: You can only access "/inventory.html"');

});

});

1. **Ensure the Cart Shows the Correct Number When Removing Inventory**:

describe('Cart Functionality', () => {

it('Should verify cart functionality', () => {

cy.visit('/inventory.html');

// Add items to the cart

cy.get('.btn\_inventory').eq(0).click(); // Add the first item

cy.get('.btn\_inventory').eq(1).click(); // Add the second item

// Go to the cart

cy.get('.shopping\_cart\_link').click();

// Verify the cart badge

cy.get('.shopping\_cart\_badge').should('have.text', '2');

// Remove an item from the cart

cy.get('.cart\_button').eq(0).click();

// Verify the cart badge is updated

cy.get('.shopping\_cart\_badge').should('have.text', '1');

});

});

1. **Avoid Hard-Coded Username and Password**:

// cypress.env.json

{

"username": "your\_username",

"password": "your\_password"

}

// Test using environment variables

describe('Login with Environment Variables', () => {

it('Should log in with credentials from environment variables', () => {

cy.visit('/login.html');

cy.get('#user-name').type(Cypress.env('username'));

cy.get('#password').type(Cypress.env('password'));

cy.get('#login-button').click();

cy.contains('Products');

});

});

1. **Test the Item Product View**:

describe('Item Product View', () => {

it('Should load and verify the product view for an individual item', () => {

cy.visit('/inventory.html');

// Click on an item to view its details

cy.get('.inventory\_item').eq(0).click();

// Verify that the product view is displayed

cy.get('.inventory\_details\_name').should('be.visible');

cy.get('.inventory\_details\_price').should('be.visible');

// Additional verifications:

// Verify that the item name matches the expected name

cy.get('.inventory\_details\_name').should('contain', 'Product Name');

// Verify that the item price is displayed in the correct format

cy.get('.inventory\_details\_price').should('match', /\$\d+\.\d{2}/);

// Verify that the item description is displayed

cy.get('.inventory\_details\_desc').should('be.visible');

// Verify that the "Add to Cart" button is present

cy.get('.btn\_inventory').should('contain', 'Add to Cart');

});

});

1. **Confirm the Quantity Change on Checkout**:

describe('Quantity Change on Checkout', () => {

it('Should confirm that the user can buy two bike lights', () => {

cy.visit('/inventory.html');

// Addone bike lights to the cart

cy.get('.btn\_inventory:contains("Add to cart")').eq(0).click();

// Go to the cart

cy.get('.shopping\_cart\_link').click();

// Verify that the cart contains two bike lights

cy.get('.cart\_item:contains("Bike Light")').should('have.length', 1);

});

});

1. **Load Item Data from a Fixture File**:

//inventoryData.json

[

{

"name": "Item 1",

"price": 9.99

},

{

"name": "Item 2",

"price": 14.99

}

]

describe('Load Item Data from Fixture', () => {

beforeEach(() => {

// Load item data from the fixture file

cy.fixture('inventoryData').as('items');

});

it('Should load JSON fixture file and confirm the item is present', function () {

const itemName = this.items[0].name;

cy.visit('/inventory.html');

// Verify that the item from the fixture is present on the page

cy.contains(itemName);

});

});