Solution of beyonnex Assessment

Test Strategy

Following the important key factors involved in test strategy.

1) Introduction:

The Weather Shopper project involves the testing of a web application that simulates an eCommerce platform. The testing approach encompasses end-to-end testing using the Cypress framework, with a focus on functionality related to temperature-based product recommendations, cart management, and payment processing.

2) Scope:

The scope of testing covers the entire application, with a focus on critical user flows, including:

- Temperature-based product selection.
- Cart interaction and validation.
- Payment processing using the Stripe gateway.
- Integration with third-party APIs.

3) Testing Objectives:

The primary objectives of testing include:

- Validate the functionality of the Weather Shopper web application.
- Ensure accurate temperature-based product recommendations.
- Verify seamless cart management and item addition.
- Confirm the success of the payment processing functionality.

4) Test Environment:

- **Testing Framework**: Cypress
- **Continuous Integration**: Jenkins
- **Containerization:** Docker

5) Test Types and Test Cases:

Functional Testing:

Functional testing will ensure that the Weather Shopper application meets its specified requirements. It includes:

- Temperature-based product selection.
- Cart functionality.
- Payment processing.

Test Case ID	WS_01	**Test Case Description**	Verify Temperature-based Product Selection		
Created By	Raza	**Reviewed By**		**Version**	

QA Tester's Log

Tester's Name	Raza	**Date Tested**	25-11-2023	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	Prerequisites:
1	Ensure the Weather Shopper website is accessible.
2	
3	
4	

S #	Test Data
1	Use the temperature values provided by the Weather Shopper application.
2	

Test Scenario

Validate that the correct products are displayed based on the temperature.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Open the Weather Shopper website.	User should able to redirect to website	User is performing request and redirect the shopper website sucessfully.	Pass
2	Read the temperature from the UI.	User is able to read the temperature from UI.	User should be able read the temperature from UI.	Pass
3	Click on the "Buy Moisturizers" button if the temperature is below 19 degrees.	User is able tobuy the moisturizers successfully.	User should able to buy moisturizers if temperature below 19 degrees.	Pass
4	Verify that only moisturizers are displayed in the product list.	User is able to verify the moisturizers product list.	User should able to validate the moisturizers product list	Pass
5	Click on buy screens button and verify product list.	User is able to click and verify product list.	User should able to validate the product list of sunscreens.	Pass

- **Test Case: Verify Cart Functionality**

  Title : Ensure seamless interaction with the shopping cart.

  Steps:

- Add a moisturizer to the cart.
- Add a sunscreen to the cart.
- Click on the cart icon to view the cart.
- Verify that both products are listed in the cart.
- Remove one product from the cart.
- Verify that the cart is updated accordingly.

  Expected Result:

- Cart functions as expected with accurate product addition and removal.

  Actual Result:

- Cart function is working as expected and its matches the expected requirement.

- **Test Case: Verify Payment Processing**

  Title: Confirm successful payment processing using the Stripe gateway.

  Steps:

- Proceed to the cart and click "Pay with Card."
- Enter valid payment details (card number, expiration, CVV, email, and zip code).
- Intercept the Stripe request and validate its details.
- Click the submit button in the payment modal.
- Wait for the Stripe request interception to complete.
- Verify the payment confirmation page is displayed.

Actual and Expected Results: Payment is successfully processed, and the confirmation page is displayed.

Integration Testing

Integration testing will focus on the interaction between different components.

- **Test Case: Verify Third-party API Integration**

Title: Verify and Confirm successful integration with the Stripe payment gateway.

Steps:

- Perform a transaction using valid payment details.
- Intercept the Stripe request and validate its details.
- Confirm that the payment is successfully processed.
- Verify the accuracy of payment-related data in the application.

Actual and Expected Results: Successful integration with the Stripe payment gateway.

- **Test Case: Verify Interaction between Components**

Title: Verify and ensure seamless interaction between temperature conditions and product recommendations.

Steps:

- Manually set the temperature to a specific value in the application backend.
- Verify that the correct products are displayed based on the adjusted temperature.
- Reset the temperature and ensure default functionality is restored.

Actual and Expected Results: Components interact seamlessly, and product recommendations are accurate.

- **End-to-End Testing**

  End-to-end testing will cover complete user journeys from product selection to payment confirmation.

Test Case: Verify Complete User Journey

Title: Verify and confirm the successful execution of the entire user journey.

Steps:

- Execute the temperature-based product selection steps.
- Add products to the cart.
- Interact with the shopping cart.
- Proceed to payment and complete the transaction.

Actual and Expected Results: The entire user journey is completed without errors, and the payment is successful.

Regression Testing:

Confirm that new updates or features do not negatively impact existing functionalities.

- **Test Case: Re-run After Updates**
  Re-run functional tests after an update to ensure ecommerce functionalities are working fine.

- **Test Case: Verify Existing Functionalities**
  Verify that existing functionalities work as expected after introducing new features.

6) Testing tools for Automation:

I have used below tools for automation.

- Cypress
- Node.JS NPM
- Docker containers
- Mocha report

Key feature includes in this,

- Dynamic element interaction based on temperature conditions.
- Intercepting and validating network requests, specifically Stripe payments.
- Dockerization for consistent and reproducible test environments.

7) Test Data:

Constants such as card number, expiration, CVV, email, and zip code are used for data-driven testing.

8) Configuration file:

- cypress.json: Configuration file for Cypress settings.
- Dockerfile: Docker configuration for building a consistent testing environment.
- Jenkinsfile: Jenkins pipeline for continuous integration (Next Plan).

9) Conclusion:

The Test Strategy outlined above ensures comprehensive coverage of the Weather Shopper application, leveraging automated testing to maintain efficiency and reliability in the testing process.

Thanks!