**ORINOCO TEST PLAN**

**Introduction**

This plan describes the testing approach that will serve as the fundamental strategy for our Orinoco Website. The purpose of the site is to provide a virtual store where visitors pick the products of choice, personalize and purchase them.

The product requirements for the Orinoco Website are as follows:

A user can choose an item.

A user can personalize an item.

A user can add an item to the cart.

A user can purchase the item.

**Features to test**

**Test No. 1: Check the AJAX API query**

Make sure the "teddy" function gets a positive response from the API.

Comment : console.log(products);

Successful: If a 201 response contains the list of products.

**Test No. 2: Check the AJAX API query error.**

Make sure the "teddy" function gets an error in the API response.

How: alert ("We're sorry, the server isn't responding!");

Successful: if error code http 400, 404 or 500 in the console.

**Test No. 3: Check that Teddy's**  **information is well recovered.**

Check that The Teddys' information is well retrieved from the server to our e-commerce site.

Comment : console.log(products);

Successful: If the teddy board appears in the console.

**Test No. 4:Check that the price of Teddy is recorded.**

Make sure the Teddys are recording their price.

Comment : console.log(products[i].price);

Successful: if teddy with its price in the console.

**Test No. 5:**  **Check for items added to the index page.**

Check that the products.forEach() feature displays items in the 'index.html' page.

How: example for name => var name = document.creatElement ('h4');

name.textContent = teddy.name;

Successful: If the articles appear on the index.html page.

**Test No. 6:**  **Check the API AJAX query to get the product ID.**

Check that the "article" function gets a positive response from the API and contains product info.

Comment : Network => Name => Headers => StatusCode : 200 OK.

Successful: If a JSON response contains product info in the Preview network.

**Tests No.7:**  **Check the item info on the product page.**

Check that the "Product View" feature displays items in the 'products.html' page.

How: example for name => var name = document.creatElement ('h3');

name.textContent - teddy.name;

nom.id ' name';

Successful: If the articles appear on the 'products' page.html'.

**Test No. 8:**  **Basket sent to LocalStorage.**

Check that the "addPanier"() feature sends a table containing the products to LocalStorage when the user clicks the "Add to the cart" button.

Settings: localStorage.setItem ('panier', JSON.stringify (panier));

Successful: If in the LocalStorage there is a table containing the products put in the cart with the click of the "Add to the cart" button.

**Test No. 9:**  **Check the success message.**

Alert the customer that the item was in the basket when the "Add to the cart" button was clicked.

How: alert ("Article added to the cartt!");

Successful: If the alert appears on the browser.

**Test No. 10:**  **Check the display of items placed in the basket on the cart page.**

Check that the "affichagePanier() feature displays items in the 'cart.html' page.

How: example for name=> var name = document.creatElement ('h3');

name.textContent - teddy.name;

Successful: If items appear on the 'basket' page.html' as a small item sheet.

And if the basket is empty: display the phrase "Your basket is empty."

**Test No. 11:**  **Check the total cost of added items.**

Make sure the display of the total cost appears in the 'cart.html' page.

Settings: var total = localStorage.getItem ('PrixTotal');

var prixTotal = document.getElementById('total');

Successful: If the total price is displayed at the bottom of the 'cart.html' page.

**Test No. 12:**  **Check the delete button in the cart.**

Check the "deletebuttons" feature by removing a product from the cart.

How: By selecting the product name in the DOM. And removing it from the LocalStorage. Then remove it from the 'cart.html' page display.

Successful: If at the click of the "delete article" button, the article will delete. And by creating an alert on the customer browser. "Teddy's name has been removed from your cart."

**Test No. 13:**  **Check the order's POST query.**

Check that the achat() function does post the request when the form is valid.

Comment : Network => Name => Headers => StatutCode : 201 created.

Successful: If ORDER returns a 201 response.

**Test No. 14:**  **Check the order's POST query.**

Check that the achat() function does post a CONTACT object and a PRODUITS table with product IDs.

Comment : Network => Name => Headers => Request Payload : {Contact… {} : Products… []}.

Successful: If ORDER returns a 201 response and the object of the data sent.

**Test No. 15:**  **Check orderId and prices are well displayed on the order page.**

Check the "commande() function shows the order number generated by the POST query and the total cost in the order confirmation page in the 'order.html' page

Comment : var div = document.creatElement(‘div’) ;

div.textContent - "Thank you for your order..";

Successful: If the page indicates the order number and the total price of the order.

**Test No. 16:**  **Check that the form inputs are mandatory.**

Make sure that input data that users will fill out is mandatory. That the post query is stopped if an input is not filled.

How: by putting in the "required" input tag make the field mandatory. And adding to the JS file "return event.preventDefault(); This prevents the function from running if the field is not properly filled.

Successful: If at the click of the "validate command" button nothing happens if a form field is not filled out. (or poorly filled).

**Features to not test**

1. A user can use the links to navigate de website.
2. A user can send a mail to contact.

**INTEGRATION TESTS**

**Test No. 1: Check the error message in case of miscommunication with the API**

Check if an error message appears when the project miscommunications the API.

**Test No. 2: List of products in case of positive response from the API**

Check to see if the product list appears when the API returns a positive response.

**Test No. 3: Check that the clicked product appears on the product page**

Check to see if the product clicked on the index page appears on the product page.

**Test 4: A success check message appears when the product is added to the basket**

Check to see if a success message appears at the top of the screen when a product is added to the cart.

**Test No. 5: Check the display when the cart is empty**

Check to see if a message appears on the basket page when the cart is empty.

**Test No. 6: Check the display when the trolley is not empty**

Check to see if the product list appears on the basket page when the basket is not empty.

**Test No. 7: Check the error message during a productless confirmation attempt**

Check if an error message appears when the user wants to confirm the productless order in the cart.

**Test No. 8: Check the error message when the first name is missing or incorrect**

Check if an error message appears when the first name is missing or incorrect in the form while the user wants to confirm the order.

**Test No. 9: Check the error message when the name is missing or incorrect**

Check if an error message appears when the last name is missing or incorrect in the form while the user wants to confirm the order.

**Test No. 10: Check the error message when the email is missing or wrong**

Check if an error message appears when the email is missing or incorrect in the form while the user wants to confirm the order.

**Test No. 11: Check the error message when the address is missing or incorrect**

Check if an error message appears when the address is missing or incorrect in the form while the user wants to confirm the order.

**Test No. 12: Check the error message when the city is missing or wrong**

Check if an error message appears when the city is missing or incorrect in the form while the user wants to confirm the order.

**Test No. 13: Check order confirmation**

Make sure the user is redirected to the order confirmation page. But also, if the order number generated by the server, as well as the total cost of the order is included.