KANAP

Testing Plan for To-Do-App

Prepared for KANAP

Prepared by: AMBUROSE Ranjith Kumar, Web Developer

1st April 2022

Proposal no° : 1

AMBUROSE RANJITH KUMAR

KANAP

Introduction:

This test plan describes the testing approach and overall frameworks that will drive the testing of Kanap.
A good User Interface is the first thing that the user comes in contact with the website. A good user interface attratcs the user and helps to create a distinct identity of a website. Ecommerce sites can get very complicated but this list is a starting point when carrying out testing before a site launch.

Main Page:

Home page with products title, description, featured products. Displaying all the products of Kanap that are available for purchace.
I call on the API using fetch the local host url: http://localhost:3000/api/products , i create a function for loop to display all the products dynamically on my home page.

Product Page:

A “product” page which will (dynamically) display the details of the product that the user clicked on when on the homepage.
From the product page users will be able to select the required quantity and colour and then add the product to the cart.

The searchParams readonly property of the URL interface returns a URLSearchParams object allowing access to the GET decoded query arguments contained in the URL.

I create variables for the product page such as: product image, product title, product price, product description, colors & quantity.

I configure an event listener when the user clicks on add to cart.

The localStorage object allows you to save key/value pairs in the browser.

Function for the pop up when a product is added in the cart

Cart Page:

Recollect the products stored in the local storage. If the Cart is empty display the cart is empty.

Create a loop, depending on the length of the products in the local storage.

Add a event listener to modify the quantity in the cart.

Refresh the localStorage with the new data retrieved...

Alert the modification and update the totals

I delete a product in the cart, save the id and color selected by the delete button, send new data to localStorage.

Alert of deleted items and reload the page

I display the total of the items in the basket.

I calculate the total amount of the cart: item price x quantity.

Request User Information, add an event listener for post form and check input validation.

KANAP

Confirmation Page:

Create a function to get the order Id.

Features to Test:

1. A user can see a single item

   Frontend :

   User can click on view product to see single product.

   Backend:

   Calling an API to return the correct JSON data of the single item to show it to the single item view.

2. A user can add a quantity of an item to the cart from the single product page view.

   Frontend:

   User can select the color of the item and click on add to cart to add one(1) sigle item to the cart.

   Backend:

   Verify if the color is been selected.

   Calling API endpoint to return the correct JSON data of the single item to add it to the cart.

3. A user can add or remove a single count of an item from the cart from the cart page view.

   Frontend:

   User can select the quantity of the item and click to add or remove from the cart

   Backend:

   Verify if the quantity is being selected.

   Verify if the item with the colour selected are added or removed from the cart.

4. A user can update the count of an item from the cart view.
   Frontend :
      User can change count ot the item to update the quantity in the cart.
   Backend:
      Check if the items in the cart, in order to change the quantity.
      Update the quantity of the items in the cart.

5. A user can only remove all count of an item from the cart view.
   Frontend:
      User can click on remove item from cart view to remove all count items from the cart
      New cart view is shown with the new items.
   Backend:
      Verify if the item exist in the cart to remove it.
      Verify the quantity in order to remove it from the cart, if quantity is more that user had selected wont't let it to remove, it will only let remove the item if the quantity is less or equal to the initial quantity.

6. A user can submit the order from the cart view.
   Frontend:
      User can click on submit form and submit order.
   Backend:
      Check if all the elements in the form are filled out.
      Calling API order check if the contact form is filled and the product array exist to give back the order ID.

Features not to Test: User cannot see the order after submitted.