Testing Plan for Orinoco Teddy Bears

**Prepared for: Sheila, Orinoco Teddy Bears**

**Prepared by: Joseph Joslin**

**GIT repo:** <https://github.com/jjoslin07/Project-5>

**Site Link:** <https://jjoslin07.github.io/Project-5/index.html>

**Introduction**

*The test plan describes the approach used to test the functionality of the Orinoco Teddy Bears website. The purpose of the site is to display all the products for sale, allow the user to add them to a shopping cart, and fil in a form to submit an order.*

**The requirements are as follows:**

* The homepage will display all the products for sale, with a link to view more details on a single item.
* The single item page will display more details as well as allowing the user to select a color and add it to the cart.
* The cart will display all the items added, allow the user to remove items, and show the total price. In addition, a form will be provided to submit an order.
* The confirmation page will thank the user for the order by name, provide an order ID and show the total price.

**Features to Test**

1. User can open the home page
   1. Frontend:
      1. All the item details and images are displayed.
   2. Backend:
      1. Making an API request returns the correct data.
   3. Edge Case:
      1. If the API request fails – Display error for the user.
      2. If the server doesn’t respond – Display error for the user.
2. User can click the link to open the single item page.
   1. Frontend:
      1. Clicking the link opens the corresponding single item page.
      2. All the item details and images are displayed.
   2. Backend:
      1. Making an API request returns the correct data.
   3. Edge Case:
      1. If the API request fails or server crashes – Display error for the user.
3. User can select colors from the drop-down menu.
   1. Frontend:
      1. Dropdown display all available colors requested.
   2. Backend:
      1. Colors are stored in local storage.
4. User can add the item to the cart, and it correctly displays with colors.
   1. Frontend:
      1. Confirmation message is displayed.
      2. The cart shows the correct items added.
5. The total price is calculated correctly.
   1. Frontend:
      1. Price is divided by 100 to show correct price.
6. User can remove item from the cart.
   1. Frontend:
      1. When user clicks to remove an item it is removed from the cart and total price updates.
   2. Backend: Item is removed from local and session storage when removed from cart.
7. The order form correctly validates user input.
   1. Frontend:
      1. The first name, last name, and city are between 1-32 characters.
      2. The address is between 1-32 characters.
      3. The email is of the correct format.
      4. No fields are left empty.
      5. Green and red borders correctly show errors.
      6. Error messages are displayed.
      7. User can’t submit if any errors are detected.
8. The order is submitted.
   1. Frontend:
      1. Successfully making an order opens the conformation page.
   2. Edge case:
      1. Incorrect object send to the backend returns error and is displayed for the user.
9. The confirmation page displays the order ID and total price correctly.
   1. Frontend:
      1. The confirmation page shows user First name and thanks them.
      2. The confirmation page shows Total Price.
      3. The confirmation page shows order ID.
   2. Edge Case:
      1. If navigation to confirmation page is made without placing order, Null is shown.

**Features not to test**

The features that will not be tested are the static elements and styling that does not affect the interactivity of the site. As well as the validity of the data sent from the backend. These features do not affect the functionality of the API request and the shopping cart, therefore do not need to be tested.

**Static elements and styling**

The elements such as the headings and paragraphs, as well as styling such as the colors and fonts will not be tested. These features do not have any effect on the API requests for JSON data from the backend, or any other interactive part of the site and do not need to be tested.

**Validity of data requested**

The data received from the backend will not be validated on the frontend, so if the site displays incorrect colors, price or has a typo in the description, it will not be rectified on the frontend.