Unit Test Plan **Introduction**

This plan describes the testing approach that will serve as the fundamental strategy for a single app. The purpose of the app is to provide the user with a list of handmade teddy bears that they can purchase with simple actions. It will be built only for web use.

The product requirements for the MVP are as follows:

- A list view page, showing all items available for sale.
- A single product page (using URL query parameters), which will dynamically show the item selected by the user, display a description and price in dollars, and allow users to personalise the product and add it to their cart.
- A cart page (using the localStorage JavaScript functionality), showing a summary of products in the cart, the total price, and a form with which to submit an order.
- An order confirmation page, thanking the user for their order, showing the total price and the order ID returned by
  the server.

The existing code is located in this repository. After cloning the repository, run `npminstall` from within the project folder, and then run the server with `node server` (requires Node installed on your machine and run locally).

Features to Test

Once the repo has been cloned or downloaded, run node server to start the back-end. For this project, I used the 'Live Server' capability within the Visual Code application to start the local server @ port 3000 to display the front end.

1. A user can access the list view page, showing all items available for sale.

Frontend: User can load page and view all available items. The URL is http://127.0.0.1:5500/index.html, and if the back-end is not running, then you can expect to see an error message displayed underneath the main heading advising of a connection error.

2. A user can view a single product page, which will dynamically show the item selected by the user.

Frontend: When clicking on the CTA - 'Product Details' (example URL, http://127.0.0.1:5500/product.html?id=5be9c8541c9d440000665243) the user will see the details for the item that they have selected. If there is a connection issue, an error message will display advising of this, just underneath the main heading.

Otherwise, the user can expect to see displayed - a description, price and allow for the user to personalise the product (select a colour).

3. A user can add an item to the cart

Frontend: User can click on call to action to add item to cart, they can then view the items in the cart. To make the order, they must fill out the form. The form will not validate unless the individual input fields are filled out correctly. Placeholder text has been used to illustrate the correct format. A title attribute has been used to offer help if the input is not validated. This is accessed by hovering over the information icon on the far-right of the input field. (Regex patterns were used for validation for all inputs bar email)

4. A user receives confirmation of order upon detail validation being completed

Frontend: Confirmation page displays total price and a unique order ID. If there is a connectivity issue, a server error will be displayed on the page.