# Orinoco

The test plan

Competed and compiled by: Mihail Tudos

# Introduction

This document outlines the test plan for the MVP developed during the project five. The Orinoco MVP consists of a single app with vintage cameras as the main product promoted and sold by Orinoco.

The MVP concludes around four main pages as listed below:

* A product list page, showing all items available for sale (**index.html**), where the items are provided via the backend API and fetched in async mode by a XMLHttpRequest.
* A single product page whose product id is passed using URL query parameters, parsed, and then used to dynamically show the item selected by the user, display its description, price in dollars, and allow users to personalize the product but also to add the item to their cart once the item is personalized (**product.html**).
* A cart page (using the localStorage JavaScript functionality), showing a summary of products in the cart, the total price, and a form that collects users contact details, validates user’s input and which is used to submit an order (**cart.html, checkout.html**).
* An order confirmation page, thanking the user for their order, showing the total price and the order ID returned by the server (**confirmation.html**).

The product requirements for the To-Do app are as follows:

* The user must be able to see all products
* The user must be able to view product details by clicking on a product
* The user must be able to customize a product
* The user must be able to add an item into their cart
* The user must be able to see an overview of their cart
* The user must be able to fill a form to place an order
* The user must be able to see a confirmation of the order being placed

# Features to test

1. The user must be able to see all products

* Frontend
  + - The user can view the product list
    - The user can click on any product and be redirected to the product page
  + Backend
    - Calling API endpoint that returns the products in JSON format

1. The user must be able to view product details by clicking on a product

* Frontend
  + - The user can view the product details
  + Backend
    - Product custom options are listed dynamically

1. The user must be able to customize a product

* Frontend
  + - The user can view the product custom options
    - The user can customize the product
  + Backend
    - The user must customize a product before adding to the cart (validation)

1. The user must be able to add an item into their cart

* Frontend
  + - The user can add only customized products to the cart
  + Backend
    - When the item is added into the cart it persists throughout the sessions and can be viewed at any point (localStorage)

1. The user must be able to see an overview of their cart

* Frontend
  + - The user can click on the cart and be redirected to a cart page
    - The user can view the details of their cart on the page
    - The user can view the total cost of their cart
  + Backend
    - The cart is fetched and parsed from the local storage

1. The user must be able to fill a form with their contact details to place an order

* Frontend
  + - The users must fill their contact details to place an order
    - The user can view the details of their cart before submitting the order
  + Backend
    - The cart is fetched and parsed from the local storage

1. The user must enter valid data before submitting an order

* Frontend
  + - The users must fill their contact details and receive feedback as the input is valid/invalid
    - As the user clicks to submit the form, filled with valid input, they will be redirected to the order confirmation page
  + Backend
    - If provided details are valid a post request will be sent to the API endpoint which will return and order confirmation
    - Returned order is saved in browser localStorage

1. The user must be able to see a confirmation of the order being placed

* Frontend
  + - As the user gets redirected to the order confirmation page a user thanking message will be displayed also showing cart details and cart total cost in $
  + Backend
    - The order is retrieved from browser localStorage
    - The order is parsed and displayed along user thanking message