# Executive Summary

The Fashionista Boutique is small collaborative of fashion designers who come together to sell their clothing lines. Most of their designs are hand crafted and limited edition. The collaborative began in 2011 and has become extremely popular. At any given time there may be less than 30 products available, and while in other businesses this would be a disadvantage, the exclusivity and the quality of the products seems to drive the popularity. This idea was chosen as it is of interest to me and after thinking about the business, I was able to imagine how I would like that business to be portrayed and what that business would want from the website. The key was the ability to easily buy products and to be able to create an account with the business to more easily buy products in the future and to be kept up to date with the business’s news.

Bootstrap was used extensively throughout the project with jQuery bringing value to the project with its functions throughout. Below is a description of the different parts of code used per page of the website and a separate description of sections which occur throughout the whole website.

# Header

A header was placed at the top of each page using Bootstrap Columns. Headers help to prevent the page from seeming cluttered and gives space for logo’s and other links to be used instead if needed. The lack of clutter allows for the eye to be drawn directly to the nav bar in most cases. Within this Navbar I placed a logo which I made with canva.com.

# Footer

Within the footer, a contact form, a logo and the business address was placed. The aesthetic of this is not ideal and if there was more time it would be changed and the Contact form would be removed from the header and placed in it’s own page. The footer itself has a different colour background from the rest of the body. Button colour was changed and different padding placed around the objects. Rounded corners were also added to the labels through CSS. On the orders, register and login pages the colours of the footer were then inverted, this was done by using jQuery.

# Navbar

The navbar background colour and font colour of the navbar was changed through the CSS, the space between the links was also manipulated this way. The icons were taken from canva and the colours were edited to match the site. They were then placed in the navbar.

# Homepage

## Carousel

The banners were created in canva, the timing of the carousel was adjusted by manipulating the Bootstrap code with data-bs-interval="10000" for each slide to change approx. every 10 secs rather than approx. every 5 secs. The pictures in the carousel were each resized to be 1024 x 700 to make sure they fit yet weren’t too big. The carousel itself was far too large for the page, originally I had edited each picture to be a particular height and I had changed the height of the carousel, but this didn’t look right and messed with the responsiveness leaving a large gap between the carousel and the cards, the smaller the screen became. In the end I was helped by code received from [stackoverflow](https://stackoverflow.com/questions/38695846/change-height-of-bootstrap-carousel-keeping-it-responsive/38696655) which allowed me to change the height and keep the responsiveness. I then changed the width to auto so that the img wouldn’t stretch too much. Each img has a link which opens in a new tab.

## Cards

Bootstrap cards were used to display some of the clothing categories on the front page. Each of the photos within were resized to 223x260 to improve the optimisation. Buttons, paragraphs and headings were removed and the title of each Card with a link remains to keep these cards simple. The Card title and the pictures link to corresponding pages. Padding was also adjusted in CSS.

# Clothing Page

This page displays the clothing available in the website. The page is divided into sections- Most Popular, Dresses, Tops, Skirts, Trousers, Coats and Accessories. Each section has a nav above it so that the other sections are always easily accessed.

## Banner

The banner was created in Canva by splicing different pictures together. These were then mapped out using img <map> in HTML by finding the coordinates by opening the image in paint and mapping out the coordinates in the HTML. The different sections of the images were then linked to their corresponding sections.

## Cards with Modal

Each card holds a piece of clothing available for sale on the site. The imgs have all been resized for optimization, the headings have been used but their size has been changed with CSS. The container with the cards has also been centred on the page with CSS, the font size and alignment has also been adjusted there. When the “Buy” button is clicked it is linked to a jQuery function which holds each item of clothing as a JSON. The id of the button triggers which JSON is to be displayed, the information within that particular object is then displayed onto a modal. This JSON is then stored in the var tempObj. On the model a user can choose the quantity of pieces of clothing that they want, this is stored within the quantity var and within the tempObj JSON under selectedQuantity. After the user has selected the quantity, they can Add to Basket. Basket is an array and the addToBasket() pushes the tempObj into the array. It was at this stage where I came to an issue and was pointed in the right direction by Sam Cogan. I wanted to store this basket so that it was available in another page, but arrays don’t exist outside of the page. I was however able to store the array in localStorage by turning it into a string with Stringify and then parsing the JSON back when in orders with the displayBasket() function.

# Register

Form was created using HTML and Javascript. Corners were rounded and the width adjusted on the input through CSS. The background colour and the font colour was also changed. The form uses HTML5 types with email type and required and section keywords. Onsubmit triggers the Javascript function checkValidation() which prevents the page from refreshing on submit and also checks whether the email entered is already in the system and if the password includes capitalisation, lowercase letters, numbers and symbols by turning a String with the password and strings with the requirements into arrays through Array.from() and looping through them to check if the requirements match, if the requirements match the matching Boolean is then set to true, if all Booleans are true the new account can be created. The submit() function is then triggered which checks that the password valid and then add’s the account information into a JSON named users. An alert is then sent out telling the user they are signed in and the loggedIn() function is triggered which runs jQuery to hide the form and change the contents of the h1 and paragraph and to personalise it with the email entered.

If the email is already in the users array an alert is sent out asking the user to sign in. If the password is invalid an alert is sent out asking for a valid password. In reality the users array does not go beyond the page and so a dummy JSON was created so that these functions could be tested.

# Login

As with the Registration form, the Login form has HTML5 keywords used. Onsubmit at the top of the form triggers the login() function. This function checks if the email is within the user array, if it is not a message appears at the bottom of the form asking the user to create an account and giving a link using .link in Javascript. There is a bug in the code here where I was trying to get the password to be asked 3 times but originally when I had the while loop in the function it would either cause the alert to pop up 3 times in a row, another buggy scenario occurred. I then thought that it was possibly about getting the whole function to run three times but it seems the event.default within the function is preventing that but it could be something else either. I hope to get the time to go back and fix this, but if I don’t the code is commented out. Once the user is signed in, the form is hidden and the h1 and paragraph is changed and the personalised email is used.

Orders

Payment

Testing and Validation

Graphical user interface, text, application, email

Description automatically generated

Figure 1: HTML Checker Report

Graphical user interface, text, application

Description automatically generated