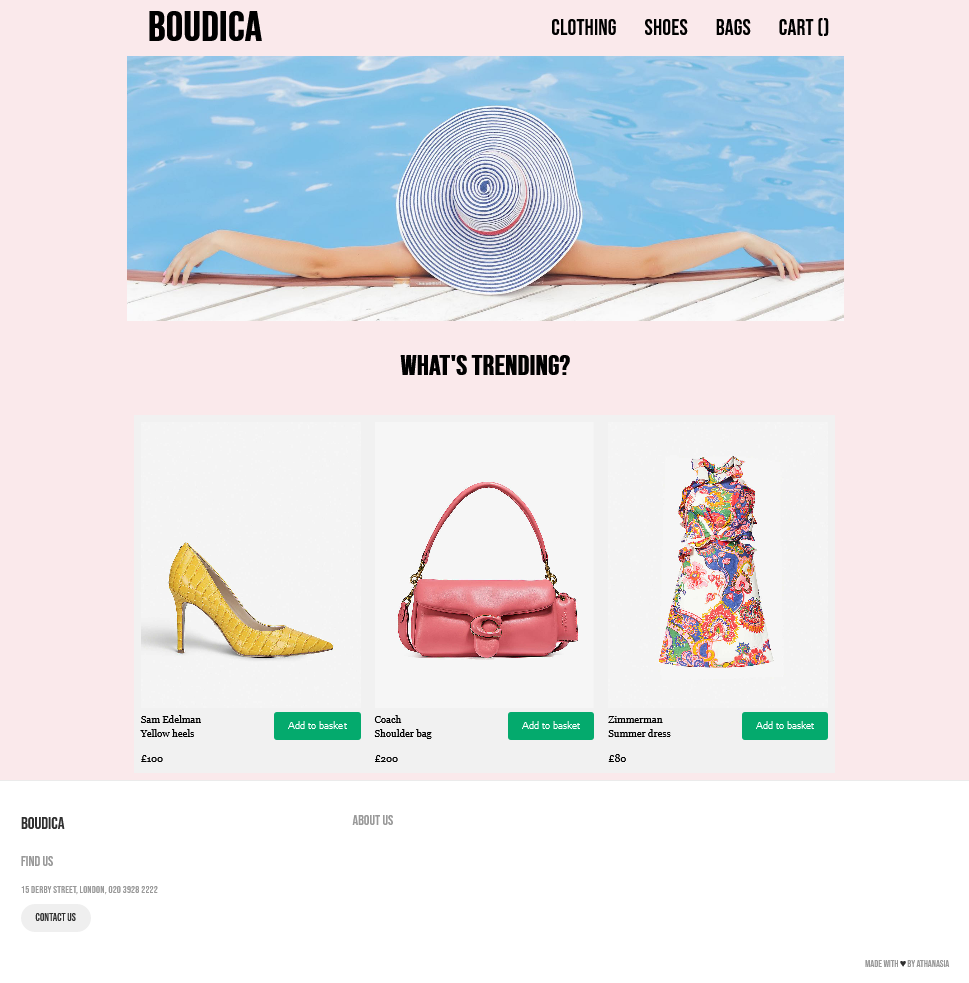
**Code Report**

The website that I built as part of this assignment is an online clothing shop. The website consists of 4 pages, the main home page, the product page, the Contact Us page and the About Us page. The main home page is the landing page where it provides some information of the content that is available in the website. The product page includes all the products that are available for sale in the website. The Contact Us page contains a form where users can submit their details and their message to the company. The last page is the About Us page where it contains information about the people who work for the company by using a mock API key to retrieve this information. Regarding the project structure, there many proposals among the online community, however I decided to keep it simple and use a single assets folder where I stored my images, CSS and JavaScript files.

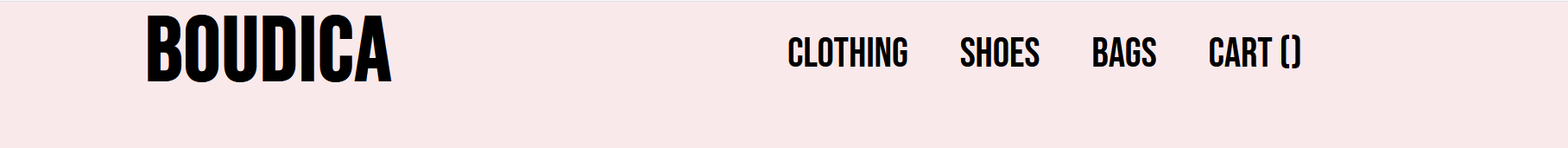
The basic colour for the website is a light pink set as background colour and black colour for all the text throughout the website apart from the footer section. The aim here was to choose a colour combination that is minimal and does not cause eye strain to the user. Most luxury brands such as Net-a-porter, Farfetch and Selfridges choose white as background colour with black text, however high contrast could cause eye strain as Connie Wong (2021) explains in her article. For this reason, the main colour for the website is light pink with black text and to differentiate the footer section from the main body, I chose white as background colour with grey text (Image 1). Also, I have used a minimal font ‘Bebas Neue’ from Google fonts.



Landing page mock-up

**Navigation bar**

Since navigation is an important part of the design process, I placed the navigation bar at the top of the page as according to Burell and Sodan (2006) navigation bars that are placed at the top and left of the page are the most popular among users. Therefore, the name of the website is placed at the far left of the navigation bar following the links to the products and the shopping cart at the far right. (Image 2).

Image 2: Top Navigation Bar

Boudica is the logo and also a link that takes you back to the main page. There are three options on the navigation bar, Clothing, Shoes and Bags that contain the products for each category and the last option is the Cart that counts the items that users have added to their shopping cart. I designed the navigation bar based on the example I found on the internet[[1]](#footnote-1). I chose this layout because I liked the top bar menu but I wanted to change the look and feel to a more minimal style. The main challenge was to make the top bar appear centred on the screen while keeping the title and the menu on the sides of the main container. I achieved this by declaring the .main-container element. I was also able to change the font and the size of the title and the menu to match my desired design.

The shopping cart was challenging for me to comprehend and develop. I created the shopping cart based on Chris Achinga’s example[[2]](#footnote-2) but I used only the parts that I could understand and implement.

**Main body**

Under the navigation bar there is a banner image and underneath the title ‘What’s trending?’ and 3 of the most popular products. The layout of the main body is achieved by using a combination of flex-container and flex-item CSS classes. The trick here was that we want to show three items per row therefore on the default view we had to fit three divs whose total width is less than 1024px (from our main-container class). In case we have more than three items, they will be added to new rows based on the configuration of flex-container.

Regarding rendering the website in smaller screens, I used media queries which means that I applied different CSS class attributes based on the width of the screen. For example, if the home page is loaded in a screen which is less than 800 pixels, the flex-direction will change from row to column.

**Product page**

There are three different categories of products in this website, clothing, shoes and bags. My initial thought was to create three different pages, one for each category but I thought that the pages would be quite similar so I did some research to check if I could have one page for all the different categories. Based on the following example from Jenna Molby[[3]](#footnote-3), it is possible to change the content of the website dynamically based on URL parameters.

Each product page link on the header contains the URL parameter category.

 <li><a href="product.html?category=clothing">Clothing</a></li>

When a page is being loaded, the JavaScript code will read this parameter and based on its value, I am appending the product to the container or not.

In order for this to work, I am declaring a global list of product objects with the following attributes:

 id: 9,

      imageSource: "",

      price: "",

      name: "",

      description: "",

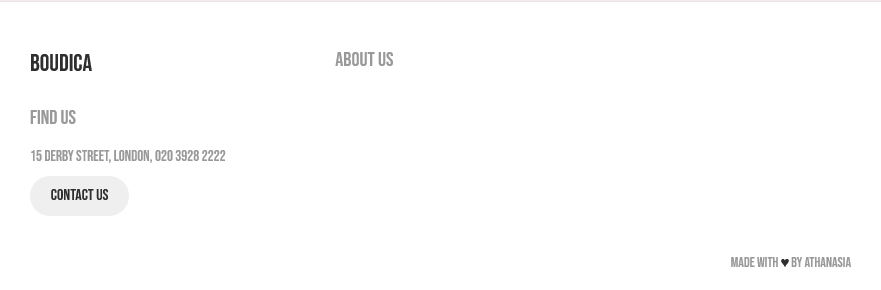
      taxonomy: "",

      popularity: "",

I am using these attributes to dynamically populate the product lists while filtering them based on taxonomies and popularity.

**Footer**

At the bottom of the page, I have placed the footer (Image 3). The footer consists of the name of the company, address, a Contact Us button and an About Us link. Also, there is a small footnote at right bottom of the footer that indicates the author of the page.

 Image 3: Footer

It was interesting to experiment with the kind of information I should include in the footer. Based on the assignment’s requirements, I decided to add the Contact Us and the About Us page. Studying the industry, it looks like the footer usually contains information about the business and sharable content with the customers. The main challenge was to keep the footer at the bottom of the page but not fixed. I manage to achieve this by using the attributes on the footer CSS class. In particular, I had set the position as absolute and the footer was fixed at the bottom of the page and it was always visible. By changing the position to relative, the footer was only visible when you scroll down to the bottom of the page.

**Contact Us page**

The Contact Us page (Image 4) is includes a form where users can enter their personal details and their message and it was based on an example from W3schools[[4]](#footnote-4). The user can open the Contact Us page through a button in the footer section. I experimented with the visual result of rendering the form in different screen sizes and generally how the user input works using JavaScript. Currently, the form submission results in an alert box showing the information entered. In real applications, I could replace this alert with a backend service to store this information in a database.

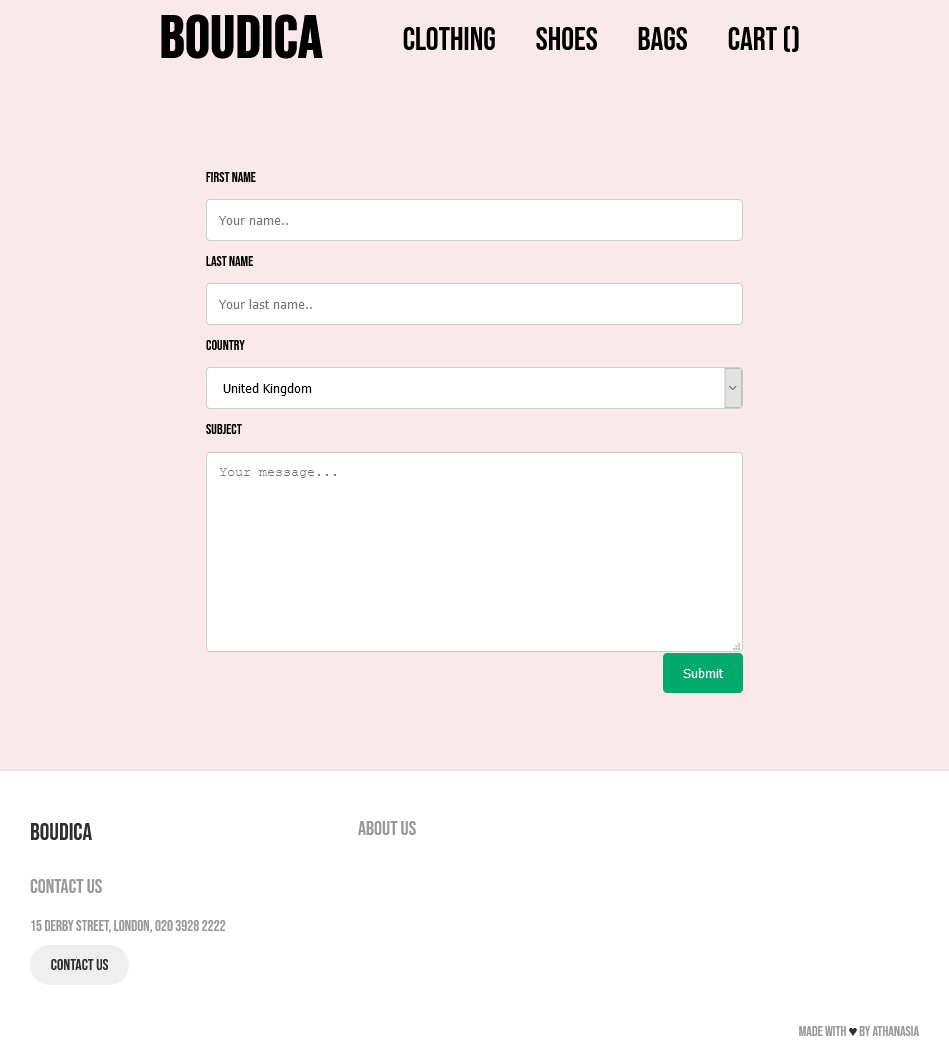


Image 4. Contact Us page mock up

**About Us page**

The About Us page contains the staff who work at Boudica. I was able to find a public API which returns a random list of names and photos. By doing an http request to <https://reqres.in/api/users> I was able to get a list of photos and names and populate my About Us page using an asynchronous request. The request is being implemented using the JQuery $.ajax function. An example of how to use this function was provided directly from the documentation at Reqres.in. My logic is implemented in the aboutus.js file.

I would like to explain further how I compile the flex items when I want to list some data. Initially, I had the products and the employees hardcoded in the html code. It took me a while to figure out that with JavaScript you can create html code as well. This is what I am trying to achieve with the personTemplate, the linkCreationTemplate and the productTemplate functions. I didn’t like the structure when I was appending html code within a loop and I decided to create those reusable functions and build the html code in a more readable way.

1. <https://codepen.io/tamayura/pen/aWqdxG/> [↑](#footnote-ref-1)
2. <https://codepen.io/chrisachinga/pen/MWwrZLJ> [↑](#footnote-ref-2)
3. <https://jennamolby.com/how-to-display-dynamic-content-on-a-page-using-url-parameters/> [↑](#footnote-ref-3)
4. <https://www.w3schools.com/howto/howto_css_responsive_form.asp> [↑](#footnote-ref-4)