Front-End Development

Gourmet HAVEN

Block 1

By Toseef Ashraf Parveen

Table of Contents

Introduction:

The website was requested by an owner of a restaurant that wants to expand his business by introducing a website so he could be more reachable. Another reason why he wants to make a website is because he also wants to add an online service for booking a table so it is more automatized so he has more time to attend the customers and to cook for them, he also wants to add a service so customers can order online and make more profit out of it.

The web technologies:

For the makeup of this website, there were the involvement of several technologies which include HTML which stands for Hypertext Markup Language, this language was used due to its simplistic language, and it is used to make the structure of the website. Another technology used is CSS (Cascading Style Sheet), this was used to style the structure of the website. To add interactivity and functions to the website, JavaScript was used. To make the logo of the business and the layout of the website Canva was used, Canva is an online graphic design platform that is used to create social media graphics and presentations. For the requirement engineering, forms were used to make a simple questionnaire to see what the vison of the client is or what could be improved. It was used because it was simple to use, and the client was familiar with it. The website has 8 different sections including a collapsible navbar, a home section, an about and our story section, the menu section which including the reservation section are the most important for this project because they are one the requested requirements. There is also a section about the team

members and their positions, a testimonial carousel of customers reviews and finally there is also a footer.

The website was made responsive using bootstrap. The reason to use bootstrap is because it is a very wide used tool to not only make the website responsive to different screen sizes but it also helps to design the website itself as it has styles that splits the width of a container up to 12 different boxes and it can be personalised to adapt into different sizes or like the about section in this website it can split the screen in half without the trouble to style it on CSS which is time consuming and very tedious, instead with bootstrap it can be easily be done by adding a class and assigning the value col-md-6, which means to split the page in half for medium size screens. Bootstrap was used for every section of this website to not only make it responsive but to also style the different components of the sections.

All the sections of this website are included in the home page for ease of scroll but there are also hyperlinks in the navbar and footer connecting to each of these sections separately for customer experience so they can navigate faster to the desired destination.

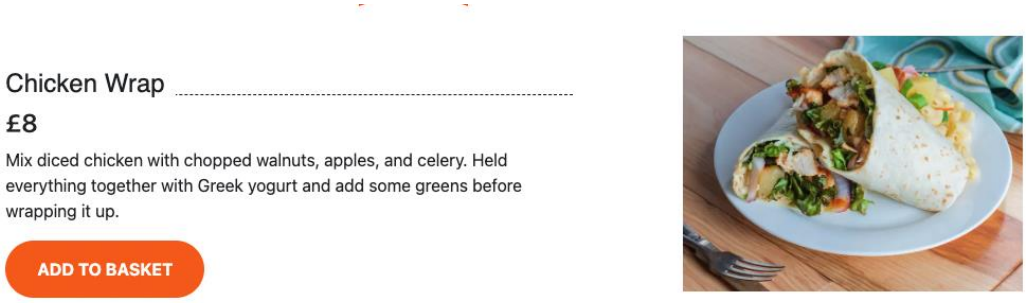


Figure 1

Because of the requirement of the client an e-commerce functionality was added in the menu section to purchase items. The reason the client wanted this feature was because he

wanted to increase the revenue of his restaurant and maybe expand it. This implementation was done in the menu next to each of the items so the customers know what they are purchasing like shown in figure 1, and to add an item to the basket to purchase the customers would have to click a button that says add to basket.

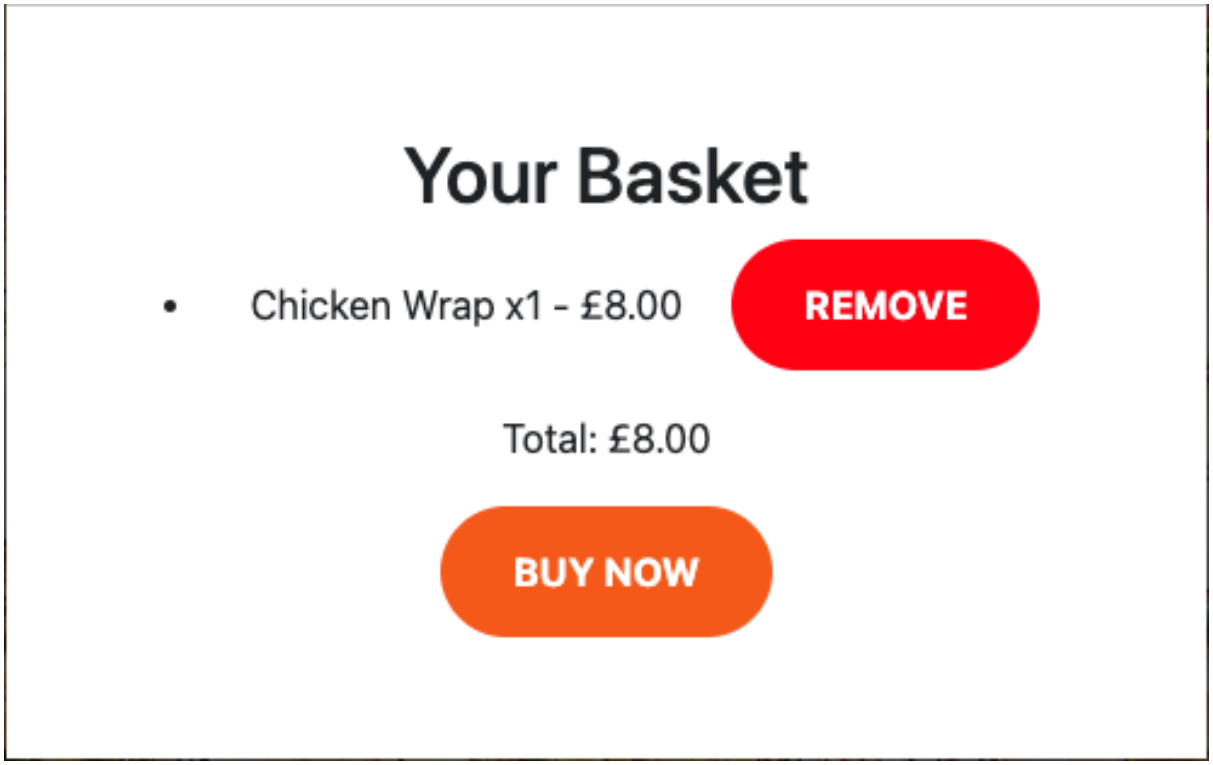


Figure 2

This will add the following item to the basket that is below the menu and customers have the liberty to add or remove any item they would like as shown in figure 2.

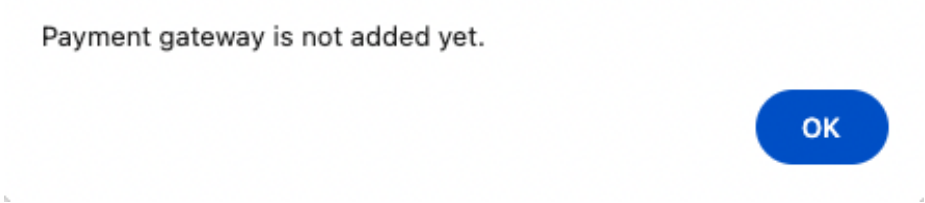


Figure 3

To finish the purchase the customers would have to click the buy button that would make them go to a payment gateway but that feature is not implemented yet so they will receive an alert displaying a message that the payment gateway is not added yet.

Figure 5

Another required element on the website is a form validation because there is a reservation form to get the customers information to book a table as shown in figure 5, and the client doesn't want to receive wrong information and it also help with the customer experience as it would alert if the information provided is wrong and it will not let it submit like in figure 6.

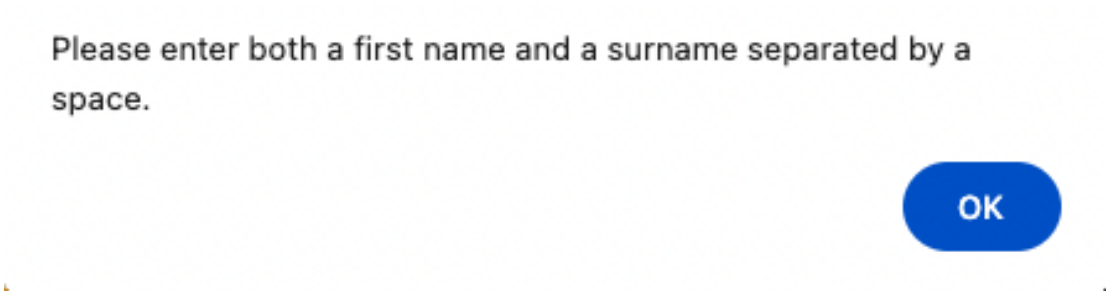


Figure 6

After the customers input the correct information and submit it, they will receive an alert displaying all the information they provided and if anything is wrong, they can contact the restaurant to change it as shown in figure 7.

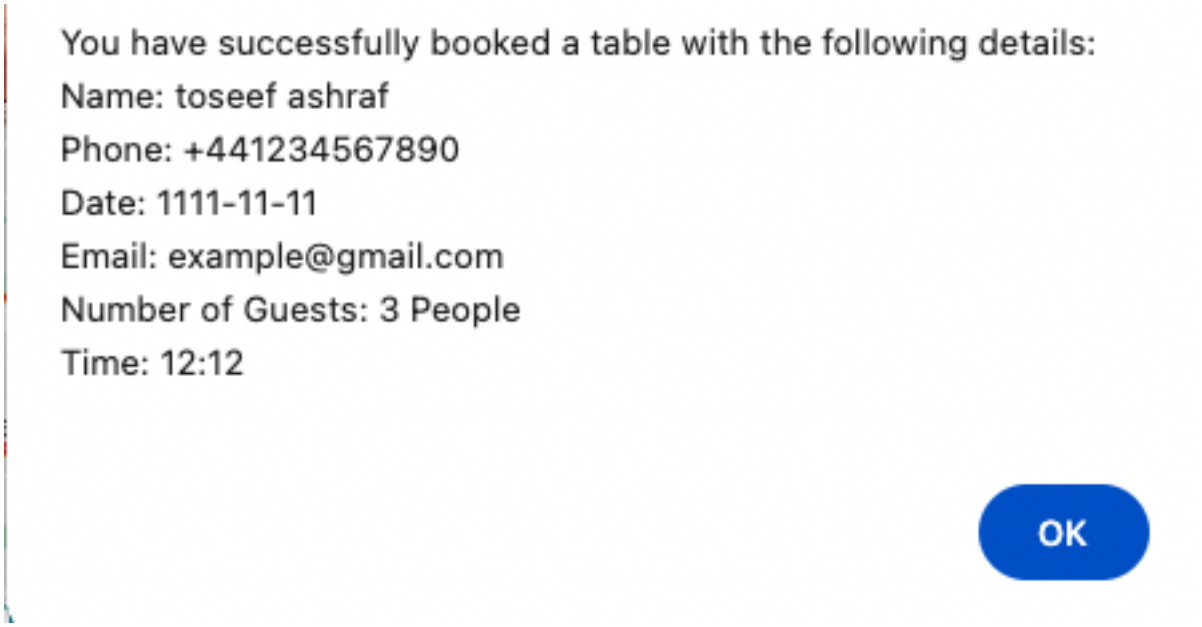
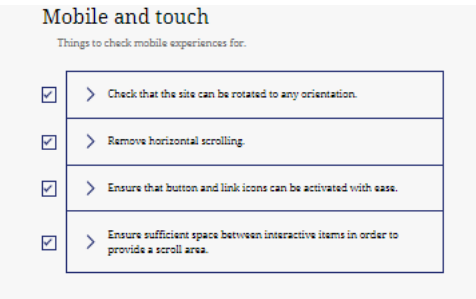


Figure 7

Accessibility:

After the website was developed it now had to go through an accessibility check to see if it meets the accessibility standards of world wide web. Another reason why accessibility is need is to make customer experience better, it doesn't just help those that are less abled, but it also must have legal compliance. It also improves the SEO (Search Engine Optimization), so it makes your website more discoverable and customer friendly. The checklist used was from a website called The A11Y Project. The following images shows the checklist:



Forms

Forms allow people to enter information into a site for processing and manipulation. This includes things like sending messages and placing orders.

- [x] All inputs in a form are associated with a corresponding `label` element.
- [x] Use `fieldset` and `legend` elements where appropriate.
- [x] Inputs use `autocomplete` where appropriate.
- [x] Make sure that form input errors are displayed in list above the form after submission.
- [x] Associate input error messaging with the input it corresponds to.
- [x] Make sure that error, warning, and success states are not visually communicated by just color.

Content

Content is the most important part of your site.

- [x] Use plain language and avoid figures of speech, idioms, and complicated metaphors.
- [x] Make sure that `button`, `a`, and `label` element content is unique and descriptive.
- [x] Use left-aligned text for left-to-right (LTR) languages, and right-aligned text for right-to-left (RTL) languages.

Global code

Global code is code that affects your entire website or web app.

- [x] Validate your HTML.
- [x] Use a `lang` attribute on the `html` element.
- [x] Provide a unique `title` for each page or view.
- [ ] Ensure that viewport zoom is not disabled.
- [x] Use landmark elements to indicate important content regions.
- [x] Ensure a linear content flow.
- [x] Avoid using the `autofocus` attribute.
- [x] Allow extending session timeouts.
- [x] Remove `title` attribute tooltips.

Keyboard

It is important that your interface and content can be operated, and navigated by use of a keyboard. Some people cannot use a mouse, or may be using other assistive technologies that may not allow for hovering or precise clicking.

- [ ] Make sure there is a visible focus style for interactive elements that are navigated to via keyboard input.
- [ ] Check to see that keyboard focus order matches the visual layout.
- [ ] Remove invisible focusable elements.

Appearance

How your website app content looks in any given situation.

- [x] Check your content in specialized browsing modes.
- [x] Increase text size to 200%.
- [x] Double-check that good proximity between content is maintained.
- [x] Make sure color isn't the only way information is conveyed.
- [x] Make sure instructions are not visual or audio-only.
- [x] Use a simple, straightforward, and consistent layout.

Animation

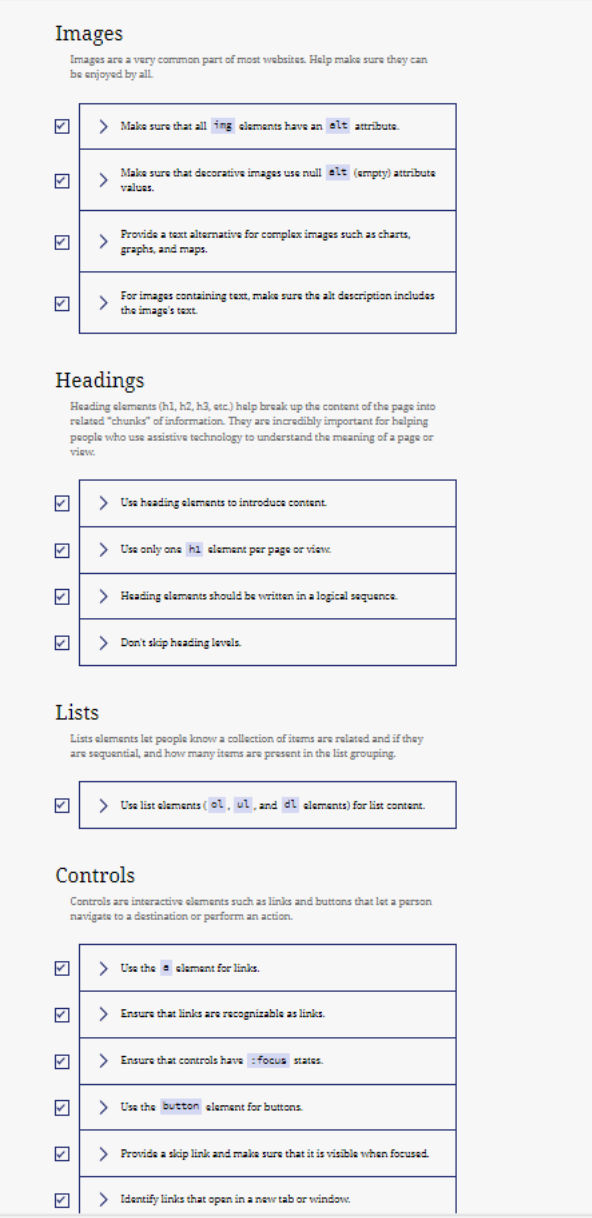
Content that moves, either on its own, or when triggered by a person activating a control.

- [x] Ensure animations are subtle and do not flash too much.
- [ ] Provide a mechanism to pause background video.
- [x] Make sure all animation obeys the `prefers-reduced-motion` media query.

Color contrast

Color contrast is how legible colors are when placed next to, and on top of each other.

- [x] Check the contrast for all normal-sized text.
- [x] Check the contrast for all large-sized text.
- [x] Check the contrast for all icons.
- [x] Check the contrast of borders for input elements (text input, radio buttons, checkboxes, etc.).
- [x] Check text that overlaps images or video.
- [x] Check custom `::selection` colors.



Some of the boxes where not ticked because they were of element that I didn't have in my website. The html was also validated at W3C validator. At the first the home had 177 total errors but after further inspection most of the errors where related to images and duplicate ids and classes and then the rest were missing closing tag.

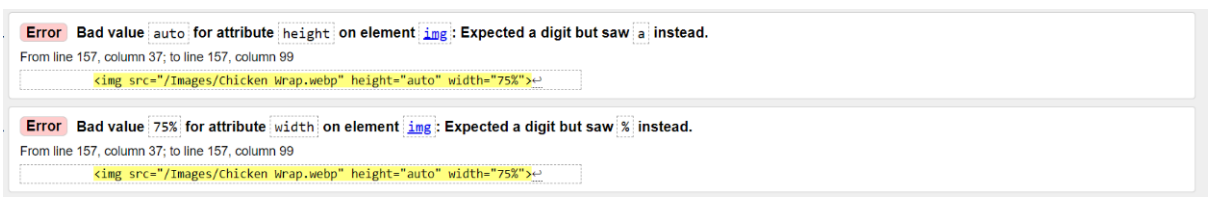


Figure 1

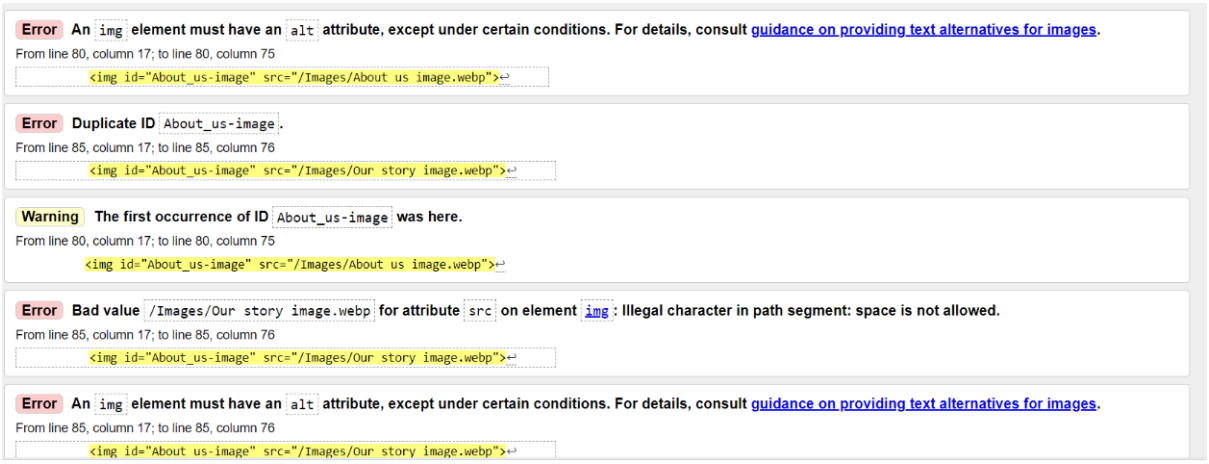


Figure 2

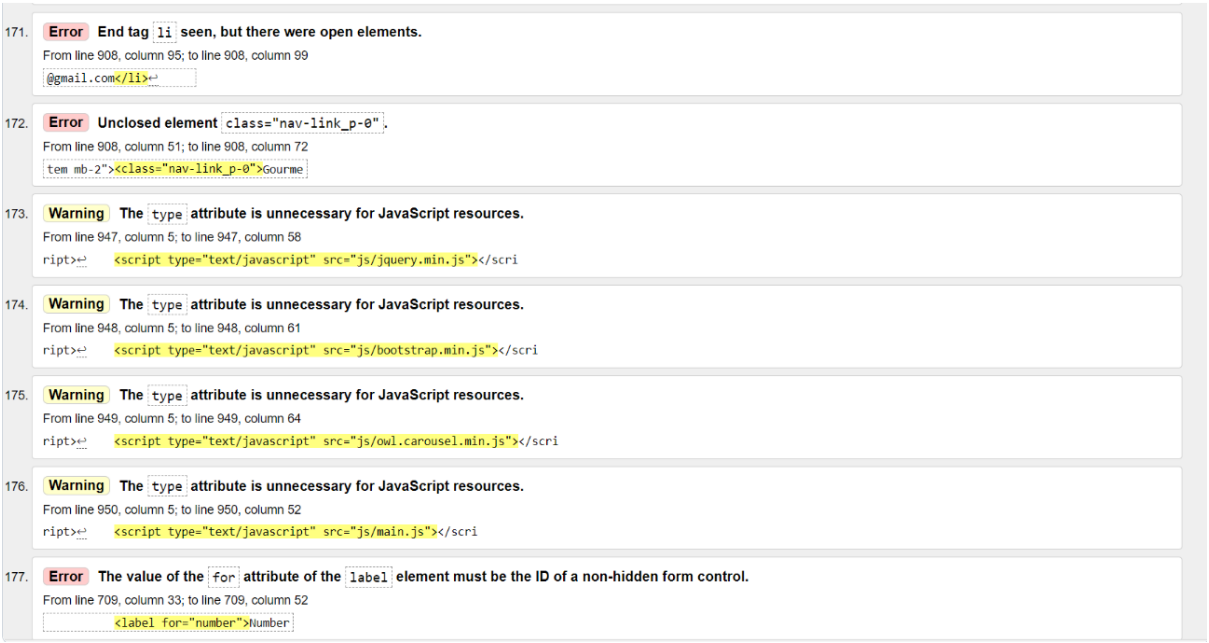


Figure 3

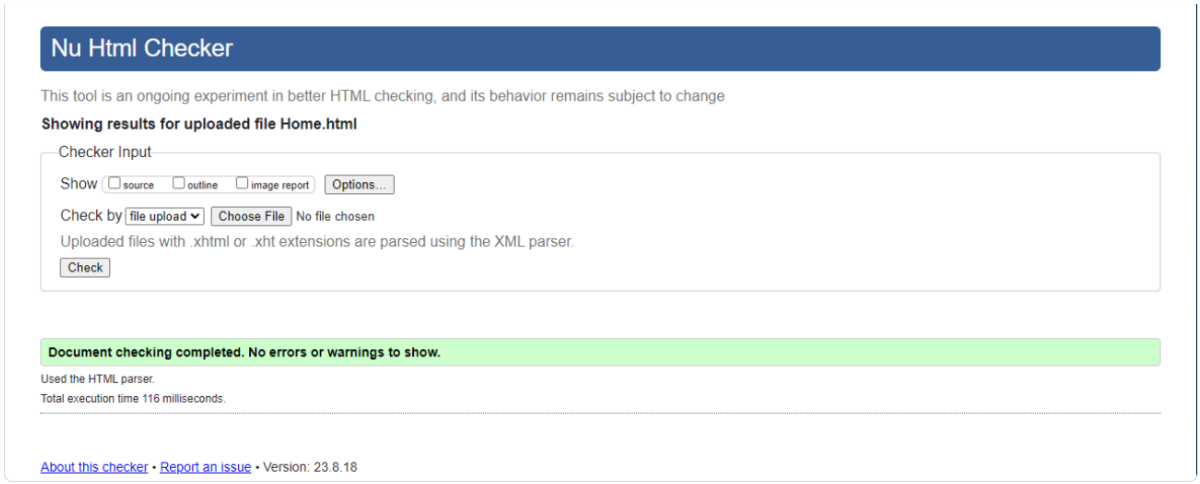


Figure 4

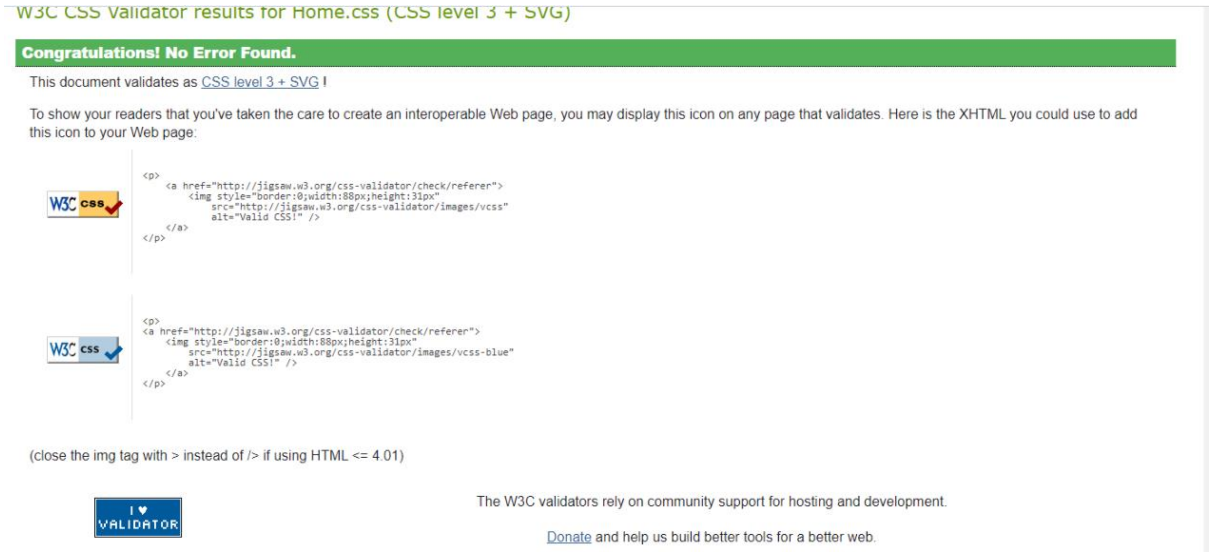


Figure 5

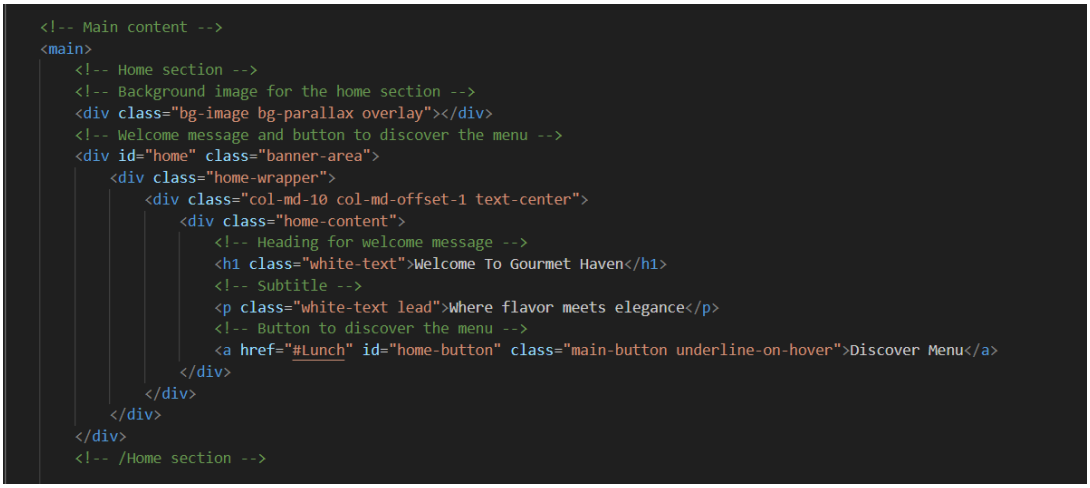
Figure 1, 2 and 3 shows all the errors I had, all other errors where a repeat of these because there was about 30 images and all images had spaces in between the name which were changed later with a dashed line. The images also needed an alt in case it stopped working and, they had a height and width, but the attribute assigned was not which was later changed and put in the style sheet instead. There were a lot of duplicate ids which were changed into classes, there was also 2

different class tags inside 1 element, one of them was removed and put inside the other one.

Figure 4 and 5 shows the end check which displays that all errors where fixed and all code was up to standard.

HTML Structure:

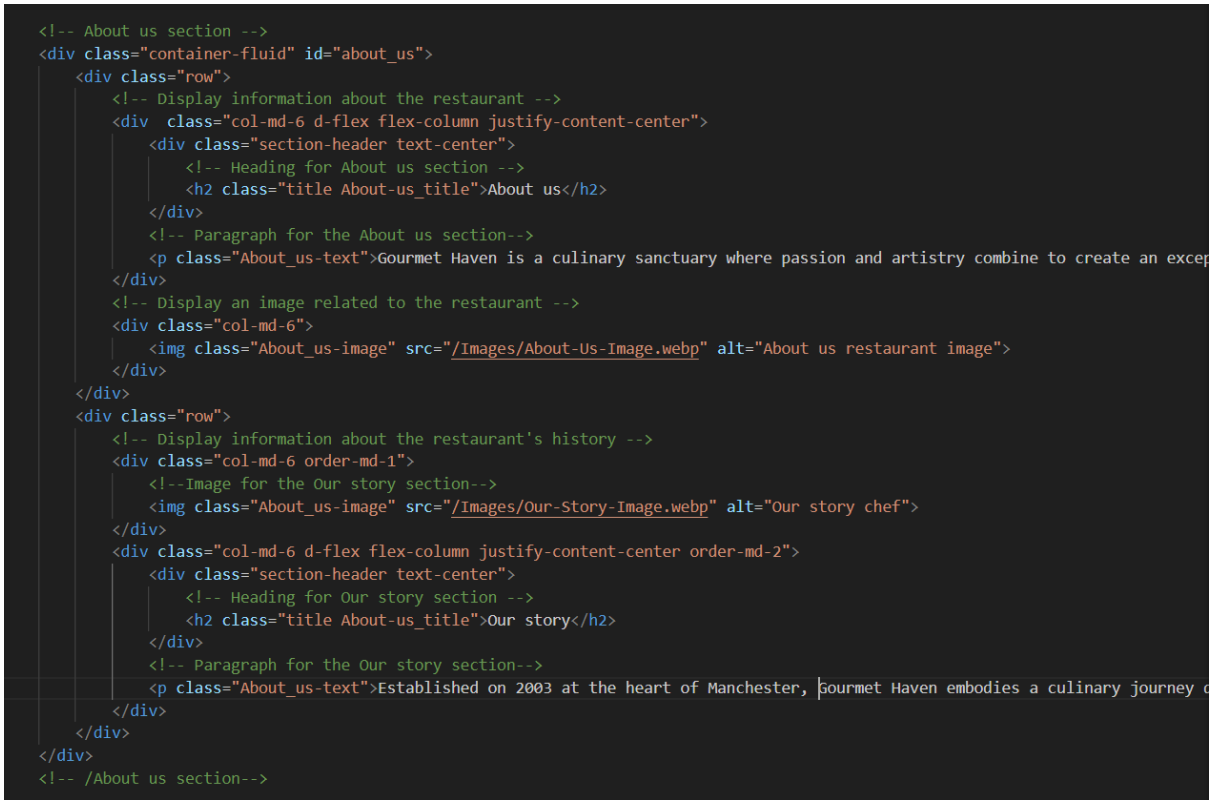
There are 4 webpages that include the sections that were mentioned in the project overview, these pages are the home page, the about page, the menu page, the reservation page. The navbar, the footer and the carousel were from the bootstrap website. The menu and the reservation form were also from a different website. All of these were modified according to the website especially the menu and the reservation form, both had a lot of problems and their JavaScript had to be changed a lot. Each of these have the footer. The home page has all the sections, so the customers have access to it without clicking something so nothing must load.



```
<!-- Main content -->
<main>
    <!-- Home section -->
    <!-- Background image for the home section -->
    <div class="bg-image bg-parallax overlay"></div>
    <!-- Welcome message and button to discover the menu -->
    <div id="home" class="banner-area">
        <div class="home-wrapper">
            <div class="col-md-10 col-md-offset-1 text-center">
                <div class="home-content">
                    <!-- Heading for welcome message -->
                    <h1 class="white-text">Welcome To Gourmet Haven</h1>
                    <!-- Subtitle -->
                    <p class="white-text lead">Where flavor meets elegance</p>
                    <!-- Button to discover the menu -->
                    <a href="#Lunch" id="home-button" class="main-button underline-on-hover">Discover Menu</a>
                </div>
            </div>
        </div>
    </div>
    <!-- /Home section -->
```

Figure 1

Figure 1 shows the home sections which contains the welcome message. It has an image overlay on top of the background image so the text is more visible, and it would be more pleasant to the customers. The content of the section is simple, it has 1 heading including the title and 1 sub-title having a paragraph. It also has a button that links to the lunch in the menu. It also has bootstrap to make it more responsive with different screen sizes.



```
<!-- About us section -->
<div class="container-fluid" id="about_us">
    <div class="row">
        <!-- Display information about the restaurant -->
        <div  class="col-md-6 d-flex flex-column justify-content-center">
            <div class="section-header text-center">
                <!-- Heading for About us section -->
                <h2 class="title About-us_title">About us</h2>
            </div>
            <!-- Paragraph for the About us section-->
            <p class="About_us-text">Gourmet Haven is a culinary sanctuary where passion and artistry combine to create an excep
        </div>
        <!-- Display an image related to the restaurant -->
        <div class="col-md-6">
            <img class="About_us-image" src="/Images/About-Us-Image.webp" alt="About us restaurant image">
        </div>
    </div>
    <div class="row">
        <!-- Display information about the restaurant's history -->
        <div class="col-md-6 order-md-1">
            <!--Image for the Our story section-->
            <img class="About_us-image" src="/Images/Our-Story-Image.webp" alt="Our story chef">
        </div>
        <div class="col-md-6 d-flex flex-column justify-content-center order-md-2">
            <div class="section-header text-center">
                <!-- Heading for Our story section -->
                <h2 class="title About-us_title">Our story</h2>
            </div>
            <!-- Paragraph for the Our story section-->
            <p class="About_us-text">Established on 2003 at the heart of Manchester, Gourmet Haven embodies a culinary journey c
        </div>
    </div>
</div>
<!-- /About us section-->
```

Figure 2

Figure 2 contains the about us section which has two components each containing a title, a paragraph and an image. One component is about the story of the restaurant and the other is about the type of restaurant it is. This was made responsive by adding bootstrap which

also made it easy to position the components as the about us section is divided into two rows each containing two columns that are 6 boxes, which is half of the page for each section of the components.

```
<!-- Menu section-->
<div id="menu" class="section">

    <!-- Background Image for the Menu section -->
    <div class="bg-image bg-parallax overlay" style="background-image:url(/Images/Background-Image.jpeg)">
    </div>
    <!-- Container for the Menu section -->
    <div class="container">

        <!-- row -->
        <div class="row">

            <!-- Section Header for the Menu section -->
            <div class="section-header text-center">
                <!-- Subtitle for the menu section -->
                <h4 class="sub-title">Discover</h4>
                <!-- Title for the menu section -->
                <h2 class="title white-text">Our Menu</h2>
            </div>

            <!-- Menu Navigation for different menu categories -->
            <ul class="menu-nav">
                <li class="active">
                    <!-- Menu item for Lunch with a link -->
                    <a data-toggle="tab" class="underline-on-hover" href="#Lunch" aria-expanded="true">Lunch</a>
                </li>

                <li class="">
                    <!-- Menu item for Dinner with a link -->
                    <a data-toggle="tab" class="underline-on-hover" href="#Dinner" aria-expanded="false">Dinner</a>
                </li>

                <li class="">
                    <!-- Menu item for Drinks with a link -->
                    <a data-toggle="tab" class="underline-on-hover" href="#Drinks" aria-expanded="false">Drinks</a>
                </li>
```

Figure 3

Figure 3 has the menu section which has 4 tabs including the lunch tab, the dinner tab, the drinks tab, and the dessert tab. Each of these have their own unique items. It also has a title and a sub-title describing the section.

```
<!-- Lunch -->
<div id="Lunch" class="tab-pane fade active in">
    <!-- Chicken Wrap -->
    <div class="row item1">
        <div class="col-md-6 d-flex flex-column justify-content-center">
            <div class="single-dish">
                <div class="single-dish-heading">
                    <!-- Name of Item -->
                    <h4 class="name">Chicken Wrap</h4>
                    <!-- Price of Item -->
                    <h4 class="price">£8</h4>
                </div>
                <!-- Description of Chicken Wrap -->
                <p>Mix diced chicken with chopped walnuts, apples, and celery. Held everything together with Greek yogurt and add some g
                <!-- Button to add 'Chicken Wrap' dish to the basket with a price of £8.00 -->
                <button class="main-button underline-on-hover" onclick="addToBasket('Chicken Wrap', 8.00)">Add to Basket</button>
            </div>
        </div>

        <!-- Chicken Wrap image -->
        <div class="col-md-6">
            <div class="Menu-image">
                <img src="/Chicken-Wrap.webp" alt="Chicken Wrap">
            </div>
        </div>
        <!-- /Chicken Wrap image -->
    </div>
    <!-- /Chicken Wrap -->
```

Figure 3

Figure 3 displays how the item is coded into the website. Each item has its own row which includes 2 columns, one containing the name of the item, the price and description of the item and it also include a button to add it to cart which is explained earlier in web technologies. The other container has the item image. This is how all 20 of the items in the menu are coded but after 5 items it just switches containers from lunch, dinner, drinks and desserts.

```
<!-- Basket-->
<div class="Basket">
    <!-- Heading for the shopping basket -->
    <h2>Your Basket</h2>
    <ul id="BasketItems">
        <!-- Basket items will be displayed here -->
    </ul>
    <!-- Display the total cost of items in the basket -->
    <p>Total: £<span id="BasketTotal">0.00</span></p>
    <!-- Button to initiate the purchase -->
    <button class="main-button underline-on-hover" onclick="buy()">Buy now</button>
</div>
<!-- /Basket-->
```

Figure 4

Figure 4 displays how the basket is implemented. It situated just after the menu so people can see what they added and delete if they don't want anything. The code consists of a title and description of the total amount and the button to buy.

```
<!-- Reservation section -->
<div id="reservation" class="section">
    <!-- Background image-->
    <div class="bg-image" style="background-image: url(/Background-Image.jpeg)"></div>
    <!-- Container-->
    <div class="container">
        <!-- Row-->
        <div class="row">
            <!-- Column for map display on medium and small screens -->
            <div class="col-md-6 col-md-offset-1 col-sm-10 col-sm-offset-1">
                <!-- Google Maps embedded using an iframe -->
                <iframe id="map"
                    src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d75993.0900249875!2d-2.30603445897900 27!3d53.4723191699000 9
                    allowfullscreen loading="lazy" referrerpolicy="no-referrer-when-downgrade"></iframe>
            </div>
```

Figure 5

Figure 5 displays the implementation of the reservation and the map. This section has a background image and a container which includes an iframe that has a map inside of it, the iframe was used so the customers would go outside of the website to another because maybe they wont return. The map feature was added from the official google maps which was embedded on this website, it is used so customers can find the location of the restaurant with ease.

```
<!-- Column for reservation form on medium and small screens -->
<div class="col-md-6 col-md-offset-1 col-sm-10 col-sm-offset-1">
    <!-- Reservation form within a row -->
    <form class="reserve-form row" id="reservationForm">
        <!-- Section header-->
        <div class="section-header text-center">
            <!-- Subtitle-->
            <h4 class="sub-title">Reservation</h4>
            <!-- Title-->
            <h2 class="title white-text">Book Your Table</h2>
        </div>
        <div class="col-md-6">
            <!-- Left half of the form -->
            <!-- Form group for name input -->
            <div class="form-group">
                <!-- Label for the name input field -->
                <label for="name">Name:</label>
                <!-- Text input for the name with placeholder and name attribute -->
                <input class="input" type="text" name="name" placeholder="Miguel Haven" id="name" autocomplete="name">
            </div>
            <!-- Form group for phone input -->
            <div class="form-group">
                <!-- Label for the phone input field -->
                <label for="phone">Phone:</label>
                <!-- Tel input for phone number with placeholder and name attribute -->
                <input class="input" type="tel" name="phone" placeholder="+44 1234 567890" id="phone" autocomplete="tel">
            </div>
            <!-- Form group for date input -->
            <div class="form-group">
```

Figure 6

```
        <label for="date">Date:</label>
        <!-- Date input for reservation date -->
        <input class="input" type="date" name="date" id="date">
    </div>
/div>
div class="col-md-6">
    <!-- Right half of the form -->
    <!-- Form group for email input -->
    <div class="form-group">
        <!-- Label for the email input field -->
        <label for="email">Email:</label>
        <!-- Email input for email address with placeholder and name attribute -->
        <input class="input" type="email" name="email" placeholder="example@gmail.com" id="email" autocomplete="emai
    </div>
    <!-- Form group for number of guests input -->
    <div class="form-group">
        <!-- Label for the number of guests input field -->
        <label for="Npeople">Number of Guests:</label>
        <!-- Dropdown select for choosing the number of guests -->
        <select class="input" id="Npeople" name="number">
            <option>Select the number</option>
            <option>1 Person</option>
            <option>2 People</option>
            <option>3 People</option>
            <option>4 People</option>
            <option>5 People</option>
            <option>6 People</option>
        </select>
    </div>
```

Figure 7

```
                        </div>
                        <!-- Form group for reservation time input -->
                        <div class="form-group">
                            <!-- Label for the time input field -->
                            <label for="time">Time:</label>
                            <!-- Time input for selecting reservation time -->
                            <input class="input" type="time" name="time" id="time">
                        </div>
                    </div>
                    <!-- Full-width column for the submission button -->
                    <div class="col-md-12 text-center">
                        <!-- Button for submitting the reservation form -->
                        <button class="main-button underline-on-hover" type="submit">Book Now</button>
                    </div>
                </form>
            </div>
        </div>
    </div>
  </div>
  <!-- /Reservation section -->
```

Figure 8

Figure 6, 7 and 8 shows the reservation form code to input information, there is a total of 6 inputs including name, email, number, number of guests, date and time. All of this have a

label, a type and an id. The number of guests also has a dropdown up to the number 6 so that is a limitation, customers would have to make a phone call if they were more than 6 people. All of this are under a title and a sub-title describing the section.

```
<!-- Team members -->
<div class="container">
    <!-- Section header-->
    <div class="section-header text-center">
        <!-- Subtitle-->
        <h4 class="sub-title" id="team-members">Team members</h4>
        <!-- Title-->
        <h2 class="title black-text">Our master chefs</h2>
    </div>
    <!-- Fluid container -->
    <div id="team-portrait" class="container-fluid">
        <!-- Row -->
        <div class="row">
            <!-- Column for a team member (1 of 4) -->
            <div class="col-xl-3 col-lg-6 col-md-6">
                <div class="white-box">
                    <!-- Featured image of the team member -->
                    <div class="featured">
                        <img class="white-box_image" src="/Chef-1.jpeg" alt="chef1">
                    </div>
                    <!-- Title for the team member -->
                    <div class="white-box_title featured">
                        <h2>Chef</h2>
                    </div>
                    <!-- Name of the team member -->
                    <div class="white-box_name featured">
                        <p>Miguel Ruiz</p>
                    </div>
                </div>
```

Figure 9

Figure 9 has the implementation of the team members section which includes bootstrap for responsiveness. The section has 1 container which contains title a sub-title and a row class. In the row class there's 4 columns which have a white box each which contains an image a name and a position of the member. After the row starts all columns have similar code but have different contents depending on the member.

```
<!-- Container for the carousel section -->
<div class="container">
    <!-- Section header-->
    <div class="section-header text-center">
        <h2 class="title" id="Review-title">Hear from our customers!</h2>
    </div>
    <!-- Owl Carousel container for testimonials -->
    <div id="owl-carousel" class="owl-carousel owl-theme mt-5">
        <!-- First carousel item -->
        <div class="owl-item">
            <div class="card">
                <!-- Image for the first customer -->
                <div class="img-card">
                    <img src="/Customer-1.jpeg" alt="Customer1">
                </div>
                <!-- Testimonial text for the first customer -->
                <div class="testimonial mt-4 mb-2">
                    My dining experience at Gourmet Haven was truly exceptional. The food was exquisite, the service was impeccable, and
                </div>
                <!-- Name of the first customer -->
                <div class="name">
                    Miguel Diaz
                </div>
            </div>
        </div>
        <!-- /First carousel item -->
```

Figure 9

Figure 9 displays the code for the testimonial carousel section. This section has a title and below it it's the carousel. The carousel starts with an id which style it and makes it a carousel because it uses bootstrap which has a library which contains all the styles and JavaScript, so it only has to be called as it already is linked on the website. First it has a class about owl-item and then card inside of it which makes like a box or layout for the content. The content consists of an image, the review and the name. All the other owl-items are similar but with different content. After the carousel the is a line of code closing the main content.

```
<!-- Footer section -->
<footer id="footer" class="py-4">
    <div class="container-fluid">
        <div class="row">
            <!-- Left column with navigation links -->
            <div class="col-md-4 col-md-2 mb-3 ml-2">
                <h5 class="footer-title">Gourmet Haven</h5>
                <ul class="nav flex-column">
                    <!-- Navigation link to Home section -->
                    <li class="nav-item mb-2 underline-on-hover"><a href="/Home.html" class="nav-link">Home</a></li>
                    <!-- Navigation link to About section -->
                    <li class="nav-item mb-2 underline-on-hover"><a href="/About.html" class="nav-link">About us</a></li>
                    <!-- Navigation link to Menu section -->
                    <li class="nav-item mb-2 underline-on-hover"><a href="/Our-menu.html" class="nav-link">Menu</a></li>
                    <!-- Navigation link to Reservation section -->
                    <li class="nav-item mb-2 underline-on-hover"><a href="/Reservation.html" class="nav-link">Reservation</a></li>
                </ul>
            </div>

            <!-- Middle column with contact information -->
            <div class="col-md-4 col-md-4 mb-3">
                <!--Title-->
                <h5 class="footer-title">Contact Us</h5>
                <ul class="nav flex-column">
                    <!--Address-->
                    <li class="nav-item mb-2">123 Street, Manchester, UK</li>
                    <!--Number-->
                    <li class="nav-item mb-2">+44 1234567890</li>
```

Figure 10

```
                <!--Email-->
                <li class="nav-item mb-2">Gourmethaven@gmail.com</li>
            </ul>
        </div>

        <!-- Right column with opening times -->
        <div class="col-md-4 col-md-2 mb-3">
            <!--Title-->
            <h5 class="footer-title">Opening Times</h5>
            <ul class="nav flex-column">
                <li class="nav-item mb-2">MONDAY-FRIDAY</li>
                <li class="nav-item mb-2">09AM-09PM</li>
                <li class="nav-item mb-2">SATURDAY-SUNDAY</li>
                <li class="nav-item mb-2">10AM-04PM</li>
            </ul>
        </div>
    </div>

    <!-- Subscription form -->
    <div class="col-md-12 d-flex flex-column flex-sm-row justify-content-center">
        <form id="newsletterForm">
            <!--Title-->
            <h5 class="footer-title">Subscribe to our newsletter</h5>
            <!--Sub-title-->
            <p>Monthly digest of what's new and exciting from us.</p>
            <div class="d-flex flex-column flex-sm-row w-100 gap-2">
                <!-- Email input field for subscription -->
                <label for="newsletterEmail" class="visually-hidden">Email address</label>
                <input id="newsletterEmail" type="text" class="form-control" placeholder="Email address">
```

Figure 11

```
                    <button class="btn btn-primary" type="button" id="subscribeButton">Subscribe</button>
                </div>
            </form>
        </div>
        <!-- /Subscription form -->

        <!-- Copyright information -->
        <div class="d-flex flex-column flex-sm-row justify-content-between my-2 border-top">
            <p>© 2023 Gourmet Haven, Inc. All rights reserved.</p>
        </div>
    </div>
</footer>
```

Figure 12

Figures 10, 11 and 12 contain the footer. Inside the footer container there's a row with 3 columns, each contain different type of information. The first has a title and has anchor tags with hyperlinks connecting to the other webpages for customer ease and better experience. The second column has a title as well and it has contact details about the restaurant. The third column has the opening and closing information. After the row the is another row containing a newsletter input with a title a button and a placeholder, this was implemented so when customers subscribe, they will receive emails about certain events or offers exclusive to them.

CSS Styles:

CSS was added to the website to style it. There are a lot of components like background image or background colour or text colour, it also was used to change the font itself, its size and make boxes around certain element like the menu or the carousel or the team members. There are a lot of these styles that have been added but there are also media query as well to make the website more responsive because bootstrap is not certainly 100% accurate.

```css
/* Media query for screens with a min width of 768px and max width of 1500px */
@media (min-width: 768px) and (max-width: 1500px) {
    /* Add padding-top to About Us images */
    .About_us-image {
        padding-top: 25%;
    }
}

/* Media query for screens with a min width of 480.6px and max width of 575.5px */
@media (min-width: 480.6px) and (max-width: 575.5px) {
    /* Adjust name label positioning in Owl Carousel */
    .owl-carousel .owl-item .card .name {
        left: 24%;
    }
}

/* Media query for screens with a max width of 360px */
@media (max-width: 360px) {
    /* Adjust name label positioning in Owl Carousel */
    .owl-carousel .owl-item .card .name {
        left: 30%;
    }
}
```

Figure 13

```css
/* Media query for screens with a max width of 480px */
@media only screen and (max-width: 480px) {
    /* Hide the map on small screens */
    #map {
        display: none;
    }
}

/* Media query for screens with a max width of 576px */
@media (max-width: 576px) {
    /* Adjust Basket margin for responsiveness */
    .Basket{
        margin-left: 0%;
        margin-right: 0%;
        width: 100%;
    }
}

/* Media query for screens with a min width of 576px and max width of 720p
@media (min-width: 576px) and (max-width: 720px) {
    /* Adjust BAsket margin for responsiveness */
    .Basket {
        margin-left: 20%;
        margin-right: 20%;
    }
}
```

Figure 14

```
    .home-content h1 {
        font-size: 36px;
    }
    /* Display copyright text centered */
    .copyright {
        display: block;
        text-align: center;
    }
}

/* Media query for screens with a max width of 767px */
@media only screen and (max-width: 767px) {
    /* Display contact navigation */
    .contact-nav {
        display: block;
    }
    /* Adjust home padding and column order */
    #home {
        padding-left: 0%;
    }
    .order-md-1 {
        order: 2; /* Rearrange the first column to appear second */
    }
    .order-md-2 {
        order: 1; /* Rearrange the second column to appear first */
    }
}
```

Figure 15

```
/*-------------------------------------*\
    Responsive
\*-------------------------------------*/

/* Media query for screens with a max width of 1201px */
@media only screen and (max-width: 1201px) {
    /* Hide the contact navigation */
    .contact-nav {
        display: none;
    }
    /* Adjust About Us text padding for responsiveness */
    .About_us-text {
        padding-left: 10%;
        padding-right: 10%;
    }
}

/* Media query for screens with a min width of 1200px */
@media (min-width:1200px) {
    /* Adjust About Us text font size */
    .About_us-text {
        font-size: large;
    }
}

/* Media query for screens with a max width of 991px */
@media only screen and (max-width: 991px) {
    /* Adjust home content heading font size */
    .home-content h1 {
```

Figure16

From Figures 13 to 16 its all about the media query which changes how different elements are seen in different screen sizes.

JavaScript Code:

```
/*-----------------------------------*\
    Menu
\*-----------------------------------*/

// Wait for the HTML document to be fully loaded and interpreted
document.addEventListener("DOMContentLoaded", function() {
    // Find the default 'Lunch' tab link and its content
    const lunchTabLink = document.querySelector('a[href="#Lunch"]');
    const lunchTabContent = document.querySelector('#Lunch');

    // Mark the default 'Lunch' tab link and tab content as 'active' and 'show'
    lunchTabLink.classList.add('active'); // Mark the default tab link as active
    lunchTabContent.classList.add('active', 'in', 'show'); // Mark the default tab content as active and shown

    // Add click event listeners to handle tab switching for all menu tab links
    document.querySelectorAll('.menu-nav a[data-toggle="tab"]').forEach(function(tabLink) {
        tabLink.addEventListener('click', function(event) {
            // Prevent the default behavior of the link (e.g., navigating to a URL)
            event.preventDefault();

            // Remove the 'active', 'in', and 'show' classes from all tab links and tab contents
            document.querySelectorAll('.menu-nav li, .tab-pane').forEach(function(item) {
                item.classList.remove('active', 'in', 'show');
            });

            // Find the corresponding tab content based on the clicked tab link
            const targetContent = document.querySelector(this.getAttribute('href'));

            // Add the 'active', 'in', and 'show' classes to the clicked tab link and its corresponding content
            this.parentElement.classList.add('active'); // Mark the clicked tab link as active
```

Figure 1

```
            // Find the corresponding tab content based on the clicked tab link
            const targetContent = document.querySelector(this.getAttribute('href'));

            // Add the 'active', 'in', and 'show' classes to the clicked tab link and its corresponding content
            this.parentElement.classList.add('active'); // Mark the clicked tab link as active
            targetContent.classList.add('active', 'in', 'show'); // Mark the corresponding tab content as active and shown
        });
    });
});
```

Figure 2

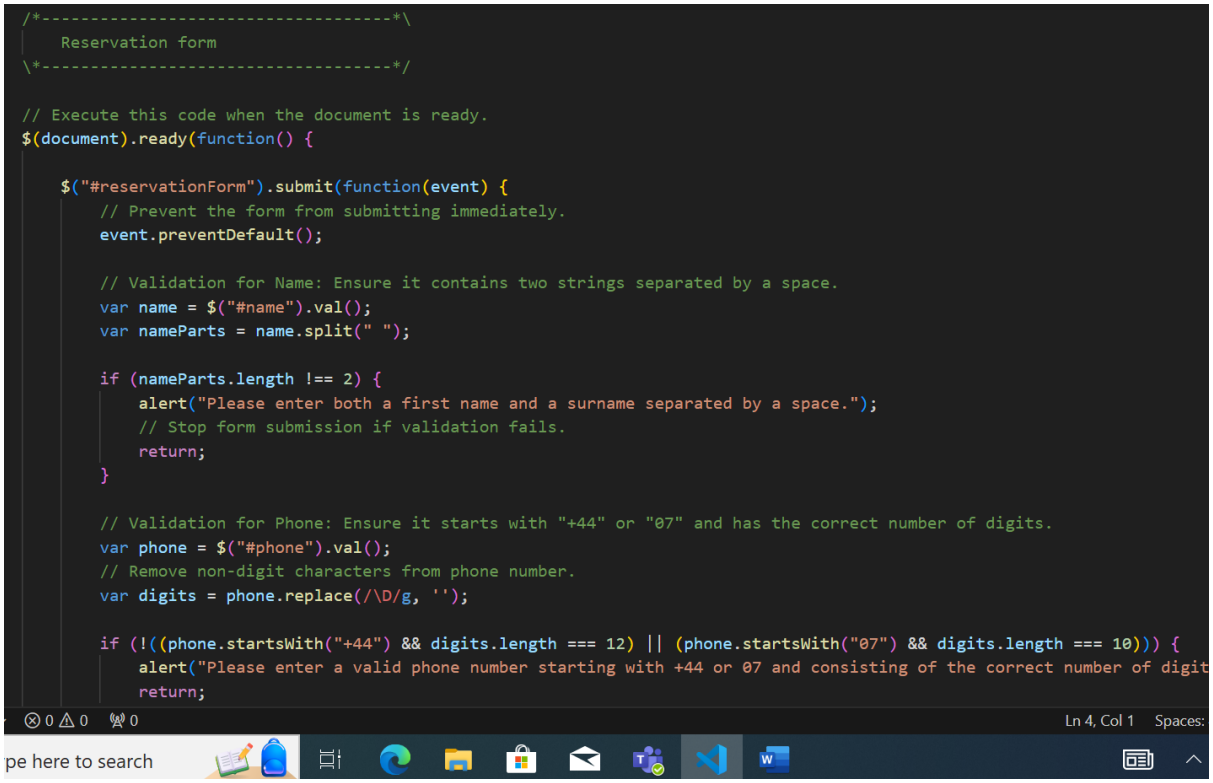


Figure 3

```
if (!((phone.startsWith("+44") && digits.length === 12) || (phone.startsWith("07") && digits.length === 10))) {
    alert("Please enter a valid phone number starting with +44 or 07 and consisting of the correct number of digits.");
    return;
}

// Validation for Date: Ensure it's not empty.
var date = $("#date").val();
if (date === "") {
    alert("Date is required. Please fill it in.");
    return;
}

// Validation for Email: Custom validation based on pattern.
var email = $("#email").val();

// Check if the email ends with "@gmail.com" or "@outlook.com."
if (!(email.endsWith("@gmail.com") || email.endsWith("@outlook.com"))) {
    alert("Please enter a valid email address ending with @gmail.com or @outlook.com.");
    return;
}

// Validation for Number of Guests: Ensure it's not the default "Select" value.
var number = $("#Npeople").val();
if (number === "Select the number" || number === "") {
    alert("Please select the number of guests.");
    return;
}
```

Figure 4

```
        // Validation for Number of Guests: Ensure it's not the default "Select" value.
        var number = $("#Npeople").val();
        if (number === "Select the number" || number === "") {
            alert("Please select the number of guests.");
            return;
        }

        // Validation for Time: Ensure it's not empty.
        var time = $("#time").val();
        if (time === "") {
            alert("Time is required. Please fill it in.");
            return;
        }

        // Display a confirmation message after successful form submission.
        var confirmationMessage = "You have successfully booked a table with the following details:\n" +
            "Name: " + name + "\n" +
            "Phone: " + phone + "\n" +
            "Date: " + date + "\n" +
            "Email: " + email + "\n" +
            "Number of Guests: " + number + "\n" +
            "Time: " + time;

        alert(confirmationMessage);

        // Reset the form after submission.
        $("#reservationForm")[0].reset();
    });
});
```

Figure 5

```
/*------------------------------------*\
    Basket
\*------------------------------------*/
// Initialize an empty array to represent the shopping basket
const Basket = [];

// Function to add a product to the shopping basket
function addToBasket(productName, price) {
    // Nested function to find a product in the basket based on its name
    function findProduct(item) {
        return item.productName === productName;
    }

    // Find a product in the basket using the findProduct function
    const product = Basket.find(findProduct);

    if (product) {
        // If the product is already in the basket, increment its quantity
        product.quantity++;
    } else {
        // If the product is not in the basket, add it with a quantity of 1
        Basket.push({ productName, price, quantity: 1 });
    }

    // Update the basket display and total cost
    updateBasket();
}
```

Figure 6

```
function removeFromBasket(productName) {
    // Find the index of the product to be removed in the basket
    const index = Basket.findIndex(item => item.productName === productName);

    if (index !== -1) {
        // If the product is found in the basket
        if (Basket[index].quantity > 1) {
            // If the product quantity is greater than 1, decrement the quantity
            Basket[index].quantity--;
        } else {
            // If the product quantity is 1, remove the product from the basket
            Basket.splice(index, 1);
        }

        // Update the basket display and total cost
        updateBasket();
    }
}

// Function to update the basket display and calculate the total cost
function updateBasket() {
    // Get the basket items container element from the HTML
    const BasketItems = document.getElementById("BasketItems");

    // Clear the basket display by setting its content to an empty string
    BasketItems.innerHTML = "";

    // Initialize a variable 'total' to keep track of the total cost
    let total = 0;
```

Figure 7

```
        removeButton.className = "remove";

        // Add an onclick event handler to remove the respective product
        removeButton.onclick = () => removeFromBasket(item.productName);

        // Display product name, quantity, and total price for each item
        BasketItem.textContent = `${item.productName} x${item.quantity} - £${(item.price * item.quantity).toFixed(2)}`;

        // Append(adding, attaching or concatenating one element or data) the "Remove" button to the list item
        BasketItem.appendChild(removeButton);

        // Append(adding, attaching or concatenating one element or data) the list item to the basket items container
        BasketItems.appendChild(BasketItem);

        // Calculate the total cost by summing the cost of each item
        total += item.price * item.quantity;
    });

    // Get the element for displaying the total cost
    const BasketTotal = document.getElementById("BasketTotal");

    // Display the total cost with two decimal places
    BasketTotal.textContent = total.toFixed(2);
}

// Placeholder function for implementing the buy process
function buy() {
    alert("Payment gateway is not added yet.");
}
```

Figure 8

```
/*-----------------------------------*\
    Newsletter
\*-----------------------------------*/

// Attach the following code to the document ready event
$(document).ready(function() {
    // Create an empty array to store subscribed email addresses
    var subscribedEmails = [];

    // Attach a click event handler to the "subscribeButton" button
    $("#subscribeButton").click(function() {
        // Get the email entered in the input field
        var email = $("#newsletterEmail").val(); // Get the value of the input field with ID "newsletterEmail"

        // Check if the email input is empty
        if (email === "") {
            // If empty, display an alert requesting the user to enter an email
            alert("Please enter your email address.");
        }
        // Check if the email does not end with either @gmail.com or @outlook.com
        else if (!(email.endsWith("@gmail.com") || email.endsWith("@outlook.com"))) {
            // If it doesn't, display an alert indicating that the email is invalid
            alert("Please enter a valid email address ending with @gmail.com or @outlook.com.");
        }
        // Check if the email is already in the list of subscribed emails
        else if (subscribedEmails.includes(email)) {
            // If it is, display an alert indicating that the user is already subscribed
            alert("You are already subscribed with this email address.");
        }
```

Figure 9

```
        // Check if the email does not end with either @gmail.com or @outlook.com
        else if (!(email.endsWith("@gmail.com") || email.endsWith("@outlook.com"))) {
            // If it doesn't, display an alert indicating that the email is invalid
            alert("Please enter a valid email address ending with @gmail.com or @outlook.com.");
        }
        // Check if the email is already in the list of subscribed emails
        else if (subscribedEmails.includes(email)) {
            // If it is, display an alert indicating that the user is already subscribed
            alert("You are already subscribed with this email address.");
        }
        // If the email is not empty and not in the list of subscribed emails, add it to the list
        else {
            subscribedEmails.push(email); // Add the email to the list of subscribed emails

            // Call a function to display a confirmation message
            displayConfirmationMessage();
        }
    });

    // Define a function to display the confirmation message
    function displayConfirmationMessage() {
        // Display a confirmation message in an alert
        alert("Thank you for subscribing to our newsletter!");

        // Clear the email input field
        $("#newsletterEmail").val(""); // Clear the value of the input field with ID "newsletterEmail"
    }
);
```

Figure 10

```
    Testimonial Carousel
\*----------------------------------*/

// Wait for the document to be fully loaded before executing any code
$(document).ready(function () {

    // Select the HTML element with the class "owl-carousel" and store it in the variable "silder"
    var silder = $(".owl-carousel");

    // Initialize the Owl Carousel on the selected element with various configuration options
    silder.owlCarousel({
        // Enable autoplay for the carousel
        autoplay: true,

        // Set the autoplay interval to 3 seconds (3000 milliseconds)
        autoplayTimeout: 3000,

        // Don't pause autoplay when hovering over the carousel
        autoplayHoverPause: false,

        // Show only one item at a time in the carousel
        items: 1,

        // Add space to the stage (container) on both sides
        stagePadding: 20,

        // Center the active item in the carousel
        center: true,
```

Figure 11

```
    // Hide navigation arrows (next/prev)
    nav: false,

    // Set the margin between items to 50 pixels
    margin: 50,

    // Show pagination dots
    dots: true,

    // Enable infinite looping of the carousel
    loop: true,

    // Define responsive settings based on the viewport width
    responsive: {
        // When viewport width is 0 or more, show 1 item
        0: { items: 1 },

        // When viewport width is 480px or more, show 2 items
        480: { items: 2 },

        // When viewport width is 575px or more, show 2 items
        575: { items: 2 },

        // When viewport width is 768px or more, show 2 items
        768: { items: 2 },
```

Figure 12

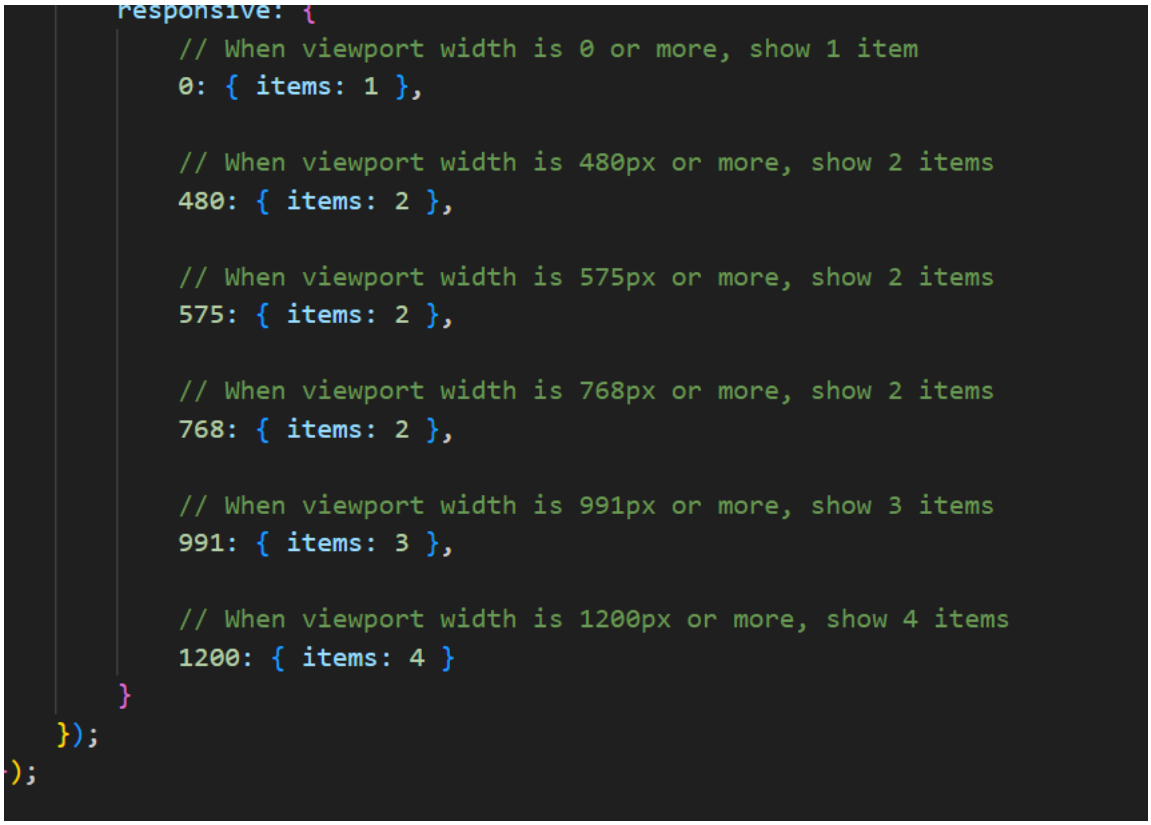


Figure 13

All these figures make up the interactivity of the website from changing tabs and displaying the content of that tab on the menu to the basket which handles the buying process from clicking on the button and the item appearing on the basket to removing the item with the remove button, and the button to the payment gateway which is not added yet.

There is also interactivity in the reservation form, the JavaScript prevents from submitting the form if all the if statements are not followed and they are the confirmation message will be alerted. This also affects the newsletter input.

There is also a code for the carousel which determines the scrolling horizontally with the mouse to the time it takes for each slide to change, it also has an autoplay.

Deploying the website:

```
10  Configuration file: none
11  To use retry middleware with Faraday v2.0+, install `faraday-retry` gem
12    Conversion error: Jekyll::Converters::Scss encountered an error while converting 'assets/css/style.scss':
13                      No such file or directory @ dir_chdir - /github/workspace/docs
14  /usr/local/bundle/gems/jekyll-sass-converter-1.5.2/lib/jekyll/converters/scss.rb:86:in `chdir': No such file or directory @
    dir_chdir - /github/workspace/docs (Errno::ENOENT)
```

Figure 1

The website was originally expected to be deployed in GitHub but due to certain errors it couldn't be deployed. Firstly, it was deployed how it was with all the sub-files, then at the second attempt the sub-files were deleted, and everything was in the main file, but it still failed. Then it was decided that it will be uploaded in another website called Netlify and it was successfully deployed there. It was decided that there won't be any sub-files so any errors don't appear.

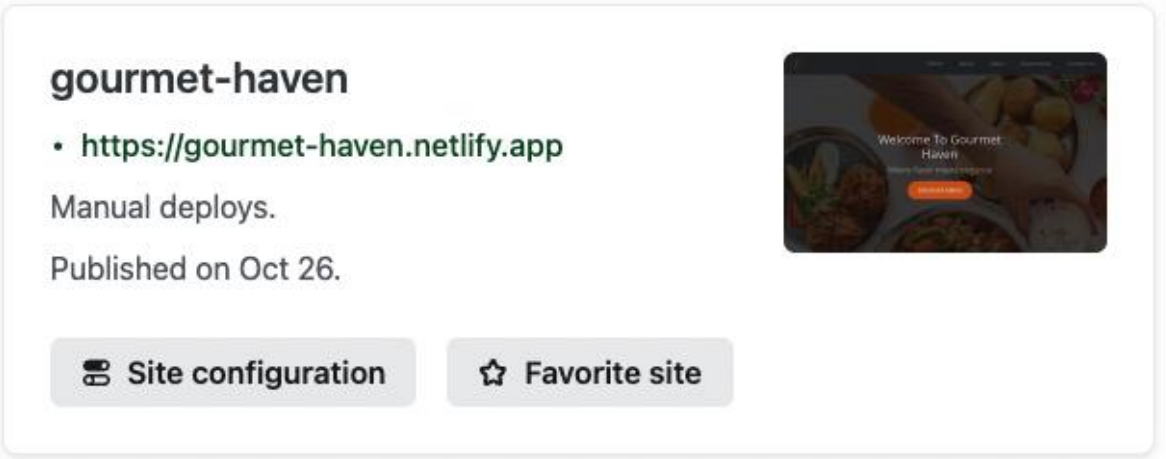


Figure 5

Figure 5 shows that it was successfully deployed in Netlify.

Requirements engineering:

Elicitation techniques:

Elicitation techniques is the collecting information from a client. Requirements elicitation activities are a way to extract or otherwise collect requirements for a business solution.

The method used for this project was a mix of brainstorming and prototyping and a questionnaire. There were sessions 1 on 1 where ideas were brainstormed, and a prototype of the website was made on that week and following that every week features were added according to the brainstormed ideas and more prototypes were made until the client was satisfied with the final project. The advantages of these are that if the client doesn't know exactly what he wanted to do or didn't have a concrete vision of his website this will give the client ideas. The disadvantages are that it is time consuming and there must be a lot of facilitation for the client to come with ideas or to inspire the client to envision his end-product. There are a lot of other elicitation techniques here are some of them:

Requirements workshop:

A requirement workshop is a structured meeting with the various owners or stakeholders and in this you will produce a model. The advantages of these type of elicitation are that there will be a lot of in-depth ideas and every one can coincide with each other and come up with an organise plan or requirement.

Interviews:

In this method there will be conversations with the client or stakeholders to gain in-depth insights or understanding. The advantages are that it will be personalised, and any misunderstanding will be cleared. The cons are that it is time consuming, and it might be biased if there are more than 1 client.

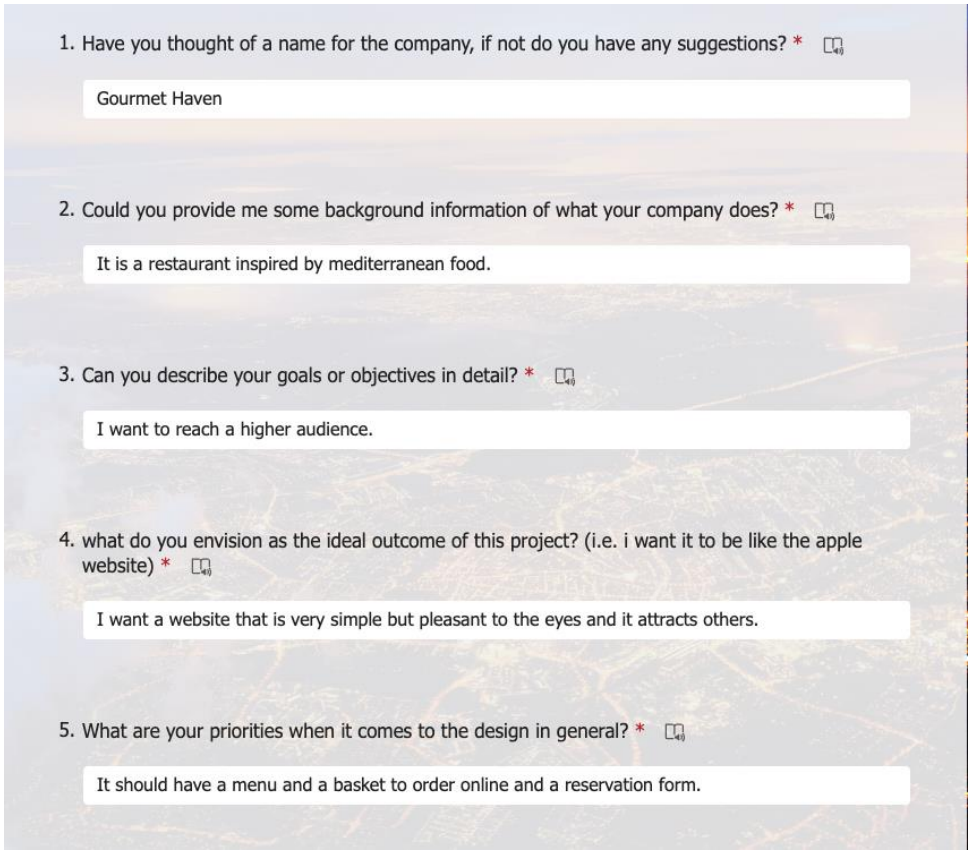


Figure 6

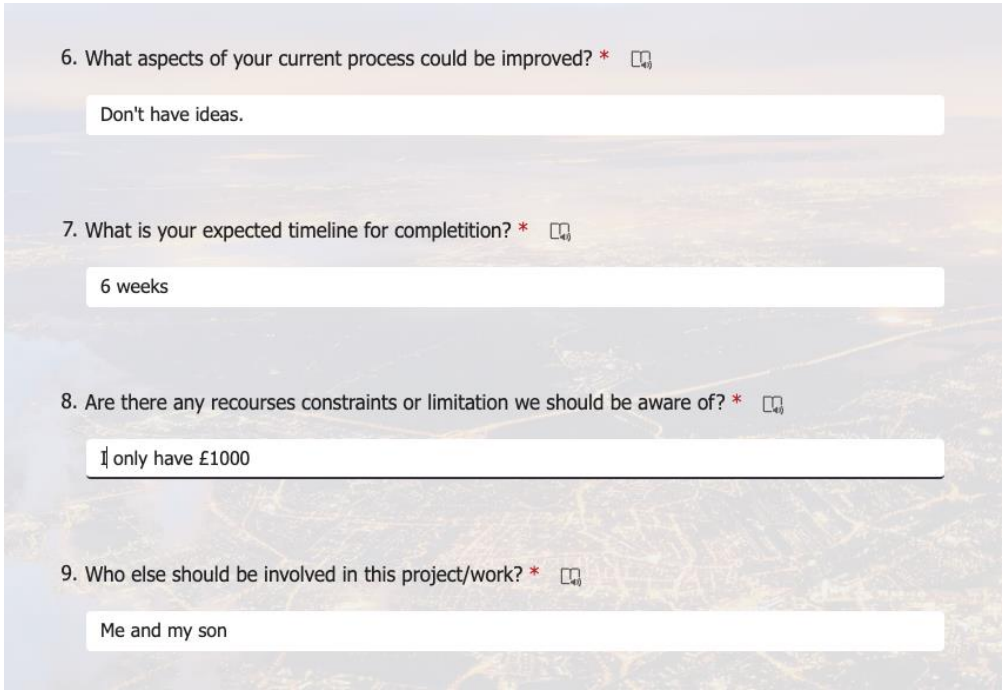


Figure 7

Figure 6 and 7 displays the questionnaire that was given to the client and its answers.

Reference list:

Nicholas, J. (2021). *9 Elicitation Techniques Used by Business Analysts - Tips and Guidance | BusinessAnalystMentor.com*. [online] businessanalystmentor.com. Available at: https://businessanalystmentor.com/elicitation-technique/.

Ahmad, S. (2017). *Restaurant HTML Template*. [online] themewagon.github.io. Available at: https://themewagon.github.io/risotto/ [Accessed 26 Oct. 2023].

Better Homes & Gardens. (n.d.). *11 Mediterranean Dessert Recipes to Satisfy Your Sweet Tooth*. [online] Available at: https://www.bhg.com/recipes/ethnic-food/mediterranean-dessert-recipes/ [Accessed 26 Oct. 2023].

Bharat (2014). *100 MOST POPULAR CELEBRITIES IN THE WORLD*. [online] IMDb. Available at: https://www.imdb.com/list/ls052283250/.

Friend, N. (2018). *Inside Danny Meyer's New NYC Restaurant, Manhattan*. [online] Food & Wine. Available at: https://www.foodandwine.com/news/danny-meyer-new-restaurant-manhatta [Accessed 26 Oct. 2023].

Manchester · Manchester, UK. (n.d.). *Manchester · Manchester, UK*. [online] Available at: https://www.google.com/maps/place/Manchester/@53.4723193 [Accessed 26 Oct. 2023].

Mitrokostas, S. (n.d.). *10 of the best things to eat for lunch on the Mediterranean diet*. [online] Insider. Available at: https://www.insider.com/best-things-to-eat-for-lunch-on-the-mediterranean-diet#try-a-salmon-nioise-salad-for-a-serving-of-vegetables-and-protein-9 [Accessed 26 Oct. 2023].

Nicholas, J. (2021). *9 Elicitation Techniques Used by Business Analysts - Tips And Guidance | BusinessAnalystMentor.com*. [online] businessanalystmentor.com. Available at: https://businessanalystmentor.com/elicitation-technique/.

Rense, S. (2020). *The 15 Most Popular Drinks to Order—or Fantasize About Ordering—at a Bar*. [online] Esquire. Available at: https://www.esquire.com/food-drink/drinks/g215/popular-bar-drinks-0609/.

Taste (2018). *Recipes, recipes and recipes - Taste*. [online] www.taste.com.au. Available at: https://www.taste.com.au/.

Thrillist (2015). *12 Reasons Why Being a Chef Is Way Harder Than You Think*. [online] Thrillist. Available at: https://www.thrillist.com/eat/nation/why-being-a-chef-is-way-harder-than-you-think.

TripAdvisor. (n.d.). *MAISON LAMELOISE, Chagny - Updated 2023 Restaurant Reviews, Menu, Prices, & Reservations*. [online] Available at: https://www.tripadvisor.co.uk/Restaurant_Review-g196538-d1082778-Reviews-Maison_Lameloise-Chagny_Saone_et_Loire_Bourgogne_Franche_Comte.html [Accessed 26 Oct. 2023].

W3.org. (2009). *The W3C CSS Validation Service*. [online] Available at: https://jigsaw.w3.org/css-validator/#validate_by_upload.

W3.org. (2013). *The W3C Markup Validation Service*. [online] Available at: https://validator.w3.org/#validate_by_upload.

W3Schools (2019). *What is Bootstrap*. [online] W3schools.com. Available at: https://www.w3schools.com/whatis/whatis_bootstrap.asp.