Web Application Development Project Submission 2024

Name: Dylan Boyle .....

Student ID: G00438786 .....

Date: 19/05/2024.....

Home page/index page/start page (eg., page user should open first): Index page (Home page) will be the first page of the website. .....

Using the site - information: This is a Clothing Website specifically on Streetwear clothes. You will be on the home page (Index.ejs) when you open the website, with logo, navigation bar, profile icon and shopping cart icon at the top of the page. Within the navigation bar, are home, about, menu, products, review, contact and blogs section, you can click any and it will bring you to that section. This part is a single page website. Beside the navigation bar are profile and shopping cart. When you click on the profile icon, it will redirect you to the sign-in and registration section. Click either button (sign-up button will switch over to the registration section, sign-in will bring you back to the sign-in section).

To use username and password root, go to registration and fill in the username, email and password and it will redirect you to a welcome page with your username and email, it will not show password for security reasons. After that you can click back to the profile icon and fill in the sign-in section and it will redirect you to a welcome back page, with your username and email. It will bring you back to the home page. In the shopping cart, there are three items to choose from, when you add or minus the quantity you want in each item, it will generate a total price, with discounted price when you reach over £100 get 20% off and £200 or over for 30% off. You can then click the close button to close the shopping cart tab or click checkout and it will redirect you to the checkout section.

In the checkout section, there are the delivery address you have to fill in, the number of items you want and payment method to pick, in order to place an order. After clicking the place an order button, you will be redirected to a thank you page. You can also give ratings and comments, in the review section. At the bottom of the review section, there is a button to give a review and it will redirect you to a review page. After a few seconds of loading time, you can use the slider to give star ratings, and underneath are the comments to write out for us to read your opinion and experience. After giving a comment and rating, you will be redirected to a thank you review page, and it will be brought back to the home page after clicking the back to home button. .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Project Requirements Implementation

ITEM 1	Reference
Allow the customer to enter their login details:	Customers can create an account in the registration section, and fill in the username, email and password fields. And login after registering their account.
Login details validated (via a login screen) before receiving a summary of the order:	(I was unable to fill in that feature, but delivery address are require to fill in order to make a ‘purchase’)
Username set to “user”	user
Password set to “pass”	pass
Brief description of implementation details:	In the checkout.js, the code handles the submission of an order form on a website. When the user clicks the "Place Order" button, it gathers the form data, performs client-side validation to ensure all required fields are filled, constructs an object containing the order details, and sends it to the server via a POST request to the '/order' endpoint. If the submission is successful, the user is redirected to a thank you page; otherwise, an error message is displayed. The functions `displayError`, `clearError`, and `displayAlert` are used for managing error messages and alerts during the submission process.

ITEM 2	Reference
Perform form validation through JavaScript or HTML to ensure that text fields are not empty, and a valid email address is entered	Validation through JavaScript is used a lot in the project. For registration, login, and checkout. Fields cannot be empty, have to use email, and certain number of characters are allow to proceed.
Brief description of implementation details:	1. Validation Conditions: It checks if any of the address, street, city, county, eircode, or country fields are empty. If any field is empty, it displays an error message to prompt the user to fill in all fields. (checkout.js) 2. Name Validation: It checks if the name field is empty. If it is, it displays an error message prompting the user to enter their name. (login.js) 3. Email Validation: It checks if the email field is empty. If it is, it displays an error message prompting the user to enter their email. It also validates the email domain to ensure it belongs to Gmail, Outlook, or Yahoo domains. 4. Password Validation: It checks if the password field is empty. If it is, it displays an error message prompting the user to enter their password. It also checks if the password length is less than 4 characters and displays an error message if it is. 5. Halt Submission: If there are any validation errors (i.e., if any of the fields are empty or if the email or password format is invalid), it sets a flag (`hasErrors`) to true and halts the form submission process. 6. Return Statement: If there are any validation errors (`hasErrors` is true), the function returns, preventing the form from being submitted to the server.

ITEM 3	Reference
Include a slideshow or carousel which displays a different image each time the page is loaded;	I have only use slideshow in the index page (home page), that show different images after certain seconds.
Brief description of implementation details:	1. HTML Structure: The slideshow container is defined with the class "slideshow-container". Inside this container, there are individual slides, each contained within a div with the class "mySlides". Each slide contains an image tag `<img>` with a source attribute pointing to an image file. (index.ejs) 2. Slideshow Function: The JavaScript function `showSlides()` is responsible for displaying the slideshow. It starts by getting all elements with the class "mySlides" and hides them by setting their display style to "none". (script.js) 3. Loop through Slides: It then iterates through each slide element and hides them. After that, it increments the `slideIndex`

	variable, which keeps track of the current slide. If `slideIndex` exceeds the total number of slides, it resets to 1. 4. Display Current Slide: It then displays the slide corresponding to the current `slideIndex` by setting its display style to "block". 5. Automatic Slideshow: Finally, it sets a timeout using `setTimeout()` to call itself again after a specified interval (in this case, every 10 seconds). This creates the effect of an automatic slideshow where the images change every 10 seconds.

ITEM 4	**Reference**
Allow the user to 'purchase' items from the site;	In the checkout page, there are three items in the cart, you can choose any amount of quantity you can, and the discount price will be applied either the total price is above £100 for 20% or £200 for 30%. You must fill in the delivery field and payment method before clicking the place an order button, otherwise it will not proceed. JavaScript was heavily used for the quantity, price and discount price.
Brief description of implementation details:	1. DOMContentLoaded Event Listener: The code inside the event listener executes when the DOM content is fully loaded. This ensures that the JavaScript code runs after the HTML document has been completely parsed and loaded. (Checkout.js) 2. Event Listeners: Event listeners are attached to various elements such as quantity inputs and the place order button to handle user interactions. 3. Client-side Validation: The code validates the form fields before submitting the data to the server. It checks if all required fields (address, street, city, county, eircode, country) are filled in. If any field is empty, an error message is displayed using the `displayError()` function. 4. Form Data Collection: After successful validation, the form data is collected into a JavaScript object named `formData`. This object includes address details, cart items (product name, price, quantity), payment method, total price, and total quantity. 5. Submit Form Data: The `submitFormData()` function sends the form data to the server using a Fetch API POST request. If the request is successful (status code 200), the user is redirected to a thank you page. Otherwise, an error message is displayed. 6. Utility Functions: Utility functions such as `displayError()`, `clearError()`, `displayAlert()`, `calculateTotalQuantity()`, and `calculateTotalPrice()` are defined to handle error messages, clear errors, display alerts, and calculate the total quantity and price of items in the cart. 7. Checkout Page Route: When a GET request is made to '/checkout', this route retrieves menu items from the database using a query. If there's an error fetching menu items, it sends a 500 status error response. Otherwise, it renders the 'checkout' template and passes the retrieved menu items data to it. (App.js)

	8. Order Submission Route: When a POST request is made to '/order', this route handles the submission of order data. It extracts the order data from the request body, which likely includes details such as address, cart items, payment method, total price, etc. The cart items are converted to a JSON string before inserting into the database. Then, it inserts the order data into the 'orders' table in the database. If there's an error during insertion, it sends a 500 status error response. Otherwise, it sends a 200 status success response. These routes ensure the functionality of the checkout process by handling the rendering of the checkout page and the submission of order data to the database.

ITEM 5	Reference
Use an object or an array in JavaScript;	The project extensively employs both Objects and Arrays throughout the project.
Brief description of implementation details:	Checkout.js, login.js, script.js and review.js heavily used objects and arrays. (Script.js) 1. Objects: - Several objects are selected using methods like `document.getElementById`, `document.querySelector`, and `document.querySelectorAll`. These objects represent various HTML elements. - Event listeners are attached to HTML elements using methods like `addEventListener`, where callback functions are used to handle events. - Functions are defined to perform specific tasks, such as adding items to the cart, updating quantities, and toggling cart visibility. - The `window.location` object is used to redirect the user to different pages based on certain actions. 2. Arrays: - Arrays are used to store collections of HTML elements retrieved using `document.querySelectorAll`. - Arrays are iterated over using methods like `forEach` to attach event listeners to multiple elements simultaneously. - Functions like `updateTotalPrice`, `updateTotalPrice1`, and `updateTotalPrice2` iterate over arrays of HTML elements to update their properties or contents.

ITEM 6	Reference
Use at least one custom module in node;	I did not use any custom module in node. But only use necessary modules for this project.

Brief description of implementation details:	1. Express: Express is a web application framework for Node.js that provides a set of features for building web applications and APIs. It simplifies the process of handling HTTP requests, defining routes, and managing middleware. 2. Express Session: Express Session is a middleware for Express that provides session management functionality. It allows you to store user session data, such as user authentication state or shopping cart contents, across multiple HTTP requests. 3. Bcrypt: Bcrypt is a library used for hashing passwords securely. It provides functions for generating hashed passwords and comparing them with stored hashed passwords to verify user credentials. 4. Path: Path is a built-in module in Node.js that provides utilities for working with file and directory paths. It's commonly used for resolving and manipulating file paths in a platform-independent manner. 5. MySQL: MySQL is a popular open-source relational database management system (RDBMS) that is widely used for storing and retrieving data in web applications. The `mysql` module in Node.js allows you to interact with MySQL databases from your Node.js application. 6. Body Parser: Body Parser is a middleware for Express that parses incoming request bodies in various formats, such as JSON, URL-encoded, or multipart form data. It makes it easy to access request body data in your route handlers. These modules collectively provide the foundation for building a web application with Node.js, handling HTTP requests, managing user sessions, interacting with a MySQL database, and securing user passwords.

ITEM 7	**Reference**
Include capability for handling post and get requests;	This project uses a good few POST and GET requests, allowing users to both submit data, such as orders and reviews, and retrieve information, like menu items and user profiles. Through these functionalities, users can interact with the application.
Brief description of implementation details:	1. Index Page Route: Retrieves menu items from the database using a SQL query. If there's an error fetching menu items, it logs the error and sends a 500 status with an error message. Otherwise, it renders the `index.ejs` template and passes the menu items data to it. 2. Checkout Page Route: Similar to the index page route, it retrieves menu items from the database and renders the `checkout.ejs` template with the menu items data. 3. Review Page Route: Renders the `review.ejs` template without passing any data. 4. Profile Page Route: Renders the `profile.ejs` template without passing any data.

	5. Order Submission Route: Handles POST requests for submitting orders. It retrieves order data from the request body, converts cart items to a JSON string, inserts the order data into the database, and sends an appropriate response. 6. User Registration Route: Handles POST requests for user registration. It logs the route being reached and the entire request body, then inserts the user into the database, sets the username and email in the session after successful registration, and redirects to the welcome page. 7. User Login Route: Handles POST requests for user login. It queries the database to find the user with the provided email and password, sets the username and email in the session upon successful authentication, and sends back the user information in the response. 8. Welcome Page Route: Retrieves username and email from the session or database and renders the `welcome.ejs` template with the username and email data. 9. Welcome Back Page Route: Retrieves username and email from the session and renders the `welcome_back.ejs` template with the username and email data. 10. Review Submission Route: Handles POST requests for submitting reviews. It receives review data from the request body, inserts the review data into the database, and sends an appropriate response. 11. Thank You Review Page Route: Renders the `thank-you-review.ejs` template with the rating data retrieved from the query parameters. 12. Add to Cart Route: Handles POST requests for adding items to the cart. It retrieves item details from the database based on the item ID, adds item details to the user's session cart data, and redirects to the homepage. 13. Thank You Page Route: Renders the `thank-you.ejs` template.

ITEM 8	**Reference**
Include both static and dynamic content;	Static content includes elements like the logo, navigation menu, and introductory text in the HTML code. Dynamic content, fetched from a database or backend server, allowing it to change dynamically based on user interactions or updates to the underlying data.

Brief description of implementation details:	Static content includes elements like the logo, navigation menu, and introductory text in the HTML code. Dynamic content is present in sections like "Menu". For example, the list of menu items displayed on your website may vary depending on database queries, making them dynamic elements. Additionally, features like form submission and cart management involve dynamic interactions with the server to process user input and update the display accordingly.

ITEM 9	**Reference**
Include the use of templates in Node;	They are used in some of the project, particularly in the ejs templates, checkout, thank you page, etc.
Brief description of implementation details:	This implementation includes dynamic rendering of cart items where the quantity can be adjusted using input fields. It also handles cases where items have discounted prices, displaying them accordingly. Additionally, it provides feedback to the user after submitting a review, displaying the rating and offering a button to return to the homepage. (Checkout.ejs, welcome.ejs, welcome back.ejs, and thank you review.ejs.)

ITEM 10	**Reference**
Include error messages to provide feedback to user sin case of issues or errors;	There are some codes used for error messages and validations to provide feedback to users from checkout.js and login.js.
Brief description of implementation details:	In login.js there is part of a form validation process for a sign-up form in a web application. 1. The code first clears any previous errors associated with the form. 2. Then it retrieves the input values for the name, email, and password fields. 3. It checks each field for validity, ensuring they are not empty and meet certain criteria (e.g., password length, email domain). 4. If any validation checks fail, it displays an appropriate error message next to the corresponding input field. 5. If there are no validation errors, the form submission process continues; otherwise, it halts submission, allowing users to correct their input. This process helps ensure that user-provided data meets the required criteria before it is submitted to the server.

	In checkout.js there is a code validation for a form, related to user address details. 1. It checks if any of the address fields (address, street, city, county, eircode, country) are empty. 2. If any field is empty, it displays an error message prompting the user to fill in all fields. 3. If all fields are filled, it creates a `formData` object containing the address details and initializes an empty array for cart items, indicating that the form data is ready for submission. This validation ensures that users provide complete address information before submitting the form.

ITEM 11	**Reference**
Connect to a database that contains relevant site information (eg., product info, prices) using NODE (your database name should be your ATU ID);	The database connection contains host; localhost, user; root, password; root and database; G00348786. In the database there are few tables, mainly menu_items (three items in shopping cart and checkout page) orders (when a user purchase something it send information such as quantity, price, payment method, etc.) accounts (when user create an account and login from that database that matched the email and password) and review table (contain ratings and comment, when a user submit a review).
Brief description of implementation details:	1. The `/` route retrieves menu items from the database and renders the `index.ejs` template with the menu items. 2. The `/checkout` route retrieves menu items from the database and renders the `checkout.ejs` template with the menu items. 3. The `/register` route handles user registration, inserting user data into the database and setting session variables for the username and email upon successful registration. 4. The `/login` route handles user login, querying the database for authentication and setting session variables upon successful login. 5. The `/submit-review` route handles the submission of reviews, inserting review data into the database.

ITEM 12	**Reference**
Use Bootstrap version 5 via CDN	Bootstrap are used in all templates, to enhance the visual and make it look good in various screen sizes, such as tv, mobile, tablet, etc.
Brief description of implementation details:	Bootstrap enhances HTML code by providing pre-defined CSS classes and JavaScript components. These classes and components create visually appealing designs. - Bootstrap's grid system allows a responsive layout. - Bootstrap provides styles for buttons, forms, navigation bars, and other UI elements.

Additional information: I did not have enough time to include an authentication for users before making a purchase, as when I was trying to apply the authentication code and function, it interrupted or make the other functions in the code not working, making the website dysfunctional. If I have a bit more time and plan it better before creating the website, I would have able to achieve all the requirements needed for this website. .....

.....

.....