Web Application Development Project Submission 2024

Name: James McDonald

Student ID: G00425765

Date: 11/05/2024

Home page/index page/start page (eg., page user should open first): index.ejs - url - http://localhost:3000/

Using the site—information: Users begin on the index.ejs page, where they are asked for their login credentials. The only valid login credentials are “user” and “pass”, checked against stored details in a user's array found in “auth.js”. After logging in successfully, users are directed to the home page. On the home page, like all pages, there is a navigation bar; however, the home’s navigation bar includes quick navigation links to different sections, and it also features a responsive design, using Bootstrap 5 and CSS styling, and a transition function from “navbar.js”. Users will see the full-screen carousel displaying items randomly generated from the database. Product cards, also dynamically generated, are featured under the carousel, showing each product with an image, description, and a “View Details” button. Clicking on this button will direct the user to the “products.ejs” page, where the selected product is dynamically generated with the image, price, description, category, and the option to add between 0-5 items to the cart. The user will also see a button to return to the home page. Underneath the product cards on the homepage is a “Checkout” button. This button will direct the user to “checkout.ejs”, where they can “purchase” their items. The option to be directed to this page is also accessible via the navigation bar by selecting the “checkout” text with the cart icon. After the checkout button, an “About” and “Contact” section are included to provide a sense of realism to the site, although it is not a project requirement. At the bottom of the main page is a section where users can subscribe to a newsletter to demonstrate email validation, the functionality of which is stored in “validateNewsletterForm.js”. The site is fully responsive, using Bootstrap 5 to ensure proper display on all devices.

Project Requirements Implementation

ITEM 1	Reference
Allow the customer to enter their login details:	See page “index.ejs” and the associated JavaScript functions from “auth.js”
Login details validated (via a login screen) before receiving a summary of the order:	Validation performed using HTML5 on “index.ejs” page
Username set to “user”	user
Password set to “pass”	pass
Brief description of implementation details:	HTML5 validation was used for username and password, and the authentication was via the JavaScript function called “authenticate()” on the login.js page. This function uses a DOM query to extract the values and checks against those stored in the database.

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	See the "home.ejs" newsletter section and the associated JavaScript function "validateNewsletterForm()" in "validateNewsletterForm.js".

Brief description of implementation details:	Validation is performed using JavaScript through the "home.ejs" page of the newsletter subscription form. The script 'validateNewsletterForm.js' checks if the email field is not empty and validates the email format against a regular expression pattern found online and referenced in the code. If the input is invalid, it updates the content of a designated message div with appropriate error messages using Bootstrap classes to style the output.

ITEM 3	**Reference**
Include a slideshow or carousel which displays a different image each time the page is loaded;	See the “home.ejs” carousel section and “app.js” file app.get(“/ home…” line 138.
Brief description of implementation details:	The carousel is integrated on the 'home.ejs' page using the Bootstrap 5 template found in the carousel section of the Bootstrap documentation. This carousel fetches a set of images from the database dynamically. For each page load, a SQL query (SELECT * FROM products ORDER BY RAND()) retrieves a random selection of product images from the products database so that a different image is loaded every time. The carousel is set to auto-start and loop through these images.

ITEM 4	**Reference**
Allow the user to ‘purchase’ items from the site;	See “checkout.ejs”, “checkout.js”, “product.ejs”, and “addToCart.js”
Brief description of implementation details:	Users can add a chosen product amount to the cart from the “product.ejs” page using the “addToCart.js” functionality. The “addToCart.js” file adds the selected items and their quantity to localStorage. When the user decides to purchase the items, they go to the “checkout.ejs” page, where a summary of their selected items, price, and quantities are displayed, along with the total, a functionality provided by “checkout.js”. The page also allows users to enter their name, address and payment method.

ITEM 5	Reference
Use an object or an array in JavaScript;	See "auth.js" and "addToCart.js
Brief description of implementation details:	User credentials are stored in an array in “auth.js.” This array manages user authentication, where each user is stored as an object with a username and password. The “createUser” function adds new users to the array, and the “authenticateUser” function iterates through the array to verify if a user exists by finding the username and matching the password. Objects are used to store and manage product details in the shopping cart with the “addToCart.js” file. Products selected by the user are stored as objects containing price and quantity in localStorage.

ITEM 6	Reference
Use at least one custom module in node;	See "auth.js" in the “scripts” folder
Brief description of implementation details:	This custom module, “auth.js”, is used to manage user authentication. The module exports two functionalities: “createUser” is a function that takes username and password as parameters and adds them to the "users" array. “authenticateUser” - a function that takes username and password as parameters and checks if they match any user objects in the “users” array.

ITEM 7	Reference
Include capability for handling post and get requests;	See “app.js".
Brief description of implementation details:	**GET Requests:** Routes such as ‘/‘, ‘/home’, ‘/checkout’, and ‘/shop’ retrieve information from the server. For example, ‘/home’ serves the main page and ‘./shop’ fetches product details based on the product id provided in the query string **POST Requests: ‘**/login’ and ‘/shop’ routes handle form submissions. The ‘/login’ route uses POST to authenticate uses through the login form.

ITEM 8	Reference
Include both static and dynamic content;	See files "index.ejs", "home.ejs", "product.ejs", "checkout.ejs".
Brief description of implementation details:	**Static Content:** This includes CSS styles and images. These are consistent across all pages and are mainly served through the public directory. **Dynamic Content:** Dynamic content is served through EJS templates, where product details, user information, and order summaries are fetched from a database and rendered on the page. For example: “home.ejs” displays products dynamically loaded from the database. “product.ejs” fetches and displays information about a specific product when a user clicks on a product card. “checkout.ejs” dynamically generates a summary of the user’s cart items, which are stored in localStorage, calculating total prices and generating payment options.

ITEM 9	Reference
Include the use of templates in Node;	See “home.ejs”, “product.ejs”, and “checkout.ejs”
Brief description of implementation details:	This project uses EJS (Embedded JavaScript) as the templating engine to generate HTML markup with JavaScript. EJS templates in home.ejs, product.ejs, and checkout.ejs are focused on rendering HTML based on product data fetched from the database. This ensures that the content users see, such as product listings, details, and items in the cart, is always up-to-date with the latest database entries.

ITEM 10	Reference
Include error messages to provide feedback to user sin case of issues or errors;	“home.ejs”, “failed.ejs”, and “app.ejs”.
Brief description of implementation details:	**Newsletter Validation:** In the newsletter sign-up (home.ejs and validateNewsletterForm.js), error messages are displayed if the email is not in a valid format. **Login Error Handling:** If the user enters incorrect information, the application renders the failed.ejs page, which provides a user-friendly error message indicating incorrect login credentials **Database Error Feedback:** On the server side, during product fetches or database interactions, any errors encountered (such as a failed database connection or query errors) are handled and logged, with appropriate messages sent back to the user.

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);	See “app.js” and “mySqlDatabase”.
Brief description of implementation details:	Database Configuration: The database connection is configured with the required credientials. - host: 'localhost', user: 'root', password: 'root', database: 'G00425765' Query Execution: SQL queries are executed throughout the application to fetch data. For example, product information is retrieved for the carousel and product detail pages. Prices are dynamically retrieved during the checkout process.

ITEM 12	Reference
Use Bootstrap version 5 via CDN	All files in “Views” folder
Brief description of implementation details:	Bootstrap 5 was used extensively throughout the project to ensure a consistent and responsive layout across all pages. The majority of the styling was achieved using Bootstrap's classes and components, requiring only a minimal amount of custom CSS.

Additional information: Xampp was used to create the database for this project and has been exported and included as part of the project. For the front end, I primarily focused on exploiting Bootstrap's functionality to the full extent. I only used CSS when Bootstrap did not offer an adequate solution. This involved extensive time with Bootstrap's documentation to become acquainted with its features. Doing this has been a positive experience, demonstrating how quickly and straightforward a front-end application can be produced and providing me with ideas for creating frameworks. Before beginning the project, I felt I would be interested more in the backend and not as interested in the front end. I expected front-end development to pose more significant challenges, but Bootstrap alleviated these difficulties and boosted my confidence for front-end tasks. By the end, the back end was confirmed as my preference, but, as I mentioned, the front end side grew on me. Overall, the project has enhanced my technical skills and piqued my interest in full-stack development.