Web Application Development Project Submission 2023

Name: Richard Daly

Student ID: G00246442

Date: 21/05/2023

Home page/index page/start page (e.g., page user should open first):

Home Page (homepage.ejs) by any of the following: http://localhost:3000/, http://localhost:3000/home

Using the site - information:

1. Download G00246442_DB.sql and import into MySQL. It is expected to be called g00246442 and the username and password both to be 'root'. If set up differently, details relating to the database must be updated in index.js within the ProjectWebServer directory.
2. Download the ProjectWebServer directory.
3. Using the Command line interface from within the above directory use the following command to install dependencies required (read from package.json): `npm install`
4. Once installation is complete use this command to start the server: `node index.js`
5. Using the browser of your choice enter the following URL to you to the site: http://localhost:3000/

Project Requirements Implementation

ITEM 1	Reference
Allow the customer to enter their login details:	Login is handled by: • 'login.html' & 'loginFailed.html' (in public folder, both static files) • POST request in 'index.js' (app.get("/login") • custom node module 'users.js' New user can be created by following the register button in navbar or below login form. **NB:** If node server is shutdown/restarted this new user will be lost.
Login details validated (via a login screen) before receiving a summary of the order:	On navigation to Checkout via the navbar on any page, the user will be asked to input login details before getting to checkout page. Once logged in the user will not need to log in again due to logged in variable in index.js. Reset if server is shutdown/restarted. Details of login will be validated by custom node modules users.js
Username set to "user"	user
Password set to "pass"	pass
Brief description of implementation details:	HTML 5 form used to gather details of username and password; both are required. On form submission, a POST request is used which is then handled in index.js by express. Details are authenticated in custom node module users.js function authenticateUser (username, password) Both username and password authenticated. On success user is directed to checkout page. On failure user is directed to a failed version of login page.

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.	login.html, register.ejs and checkout.ejs all perform form validation through HTML. Checkout.ejs has both text and email inputs in form with id = paymentForm.
Brief description of implementation details:	All input tags used within forms throughout the website have the HTML input required attribute to ensure all are filled out before the form can be submitted. HTML input types are also used to distinguish the fields and the validation required. An email requires an '@' symbol etc.

ITEM 3	Reference
Include a slideshow or carousel which displays a different image each time the page is loaded.	Bootstrap Carousel used in homepage.ejs, short JavaScript script at end of document selects image to be used.
Brief description of implementation details:	Each carousel item in has a number id attribute assigned to it. A short script at the end of the document generates a random number on page load. Using the following the statement, the class attribute with the corresponding number id is updated to be displayed: document.getElementById("number id").classList.add("active) **NB:** The carousel contains five images, the same number may be generated, reload page if this occurs.

ITEM 4	Reference
Allow the user to 'purchase' items from the site;	On navigation to a product page rendered by kayakProduct.ejs or clothingProduct.ejs the user can select a quantity and select add to cart. **NB**: Navigation to these pages should be from kayaks.ejs or clothing.ejs so that a product id included.
Brief description of implementation details:	Each product has an addToCart () JavaScript function included at the end of the page which retrieves the product name, price, and quantity. These are each then saved to local storage and retrieved from the same for use within checkout.ejs. Completion of the form on checkout.ejs will bring the 'transaction' to conclusion and clear the products selected from local storage.

ITEM 5	Reference
Use an object or an array in JavaScript;	Object: • index.js in each instance that information is retrieved from the database, stored as an object, and use as a parameter in rendered an ejs page. • kayakProduct.ejs and clothingProduct.ejs in the addToCart () function. Information stored in object, converted to a string, and stored in local storage. Array • users.js the user data is stored in an array. • Checkout.ejs in script used on load multiple statements stored in array and then passed to an element's inner html. • kayakProduct.ejs and clothingProduct.ejs both use an array the appendAlert arrow function.
Brief description of implementation details:	An object is used where a key value pair is necessary and where multiple values may be associated with one key. An array was use where a list was needed, and single value as associated with one key.

ITEM 6	Reference
Use at least one custom module in node;	Users.js
Brief description of implementation details:	Information of users and their passwords is stored here. There are multiple functions to create a new user and check if the user already exists. All functions are exported from the module for use. Index.js requires this module and creates a variable 'users' to access its methods.

ITEM 7	Reference
Include capability for handling post and get requests;	Get and post requests are handled by express in index.js
Brief description of implementation details:	Where the information being passed was less sensitive and required less security a get request was used. Get: • app.get (“/”) • app.get (“/home”) • app.get (“/kayaks”) • app.get (“/kayak”) • app.get (“/clothing”) • app.get (“/product”) • app.get (“/checkout”) • app.get (“/login”) • app.get (“/register”) Where additional security was required, and that information would not be shown in the URL a post request was used. Instead transferred in payload. Post: • app.post (“/loginsubmit”) • app.post (“/submitUser”)

ITEM 8	Reference
Include both static and dynamic content;	Fully static Pages: login.html and loginFailed.html All other pages are rendered dynamically using EJS templates. Dynamic elements: homepage.ejs – carousel element. clothingProduct.ejs & kayakProduct.ejs – live alert button.
Brief description of implementation details:	login.html and loginFailed.html are fully static pages that do not differ in anyway. All EJS pages while are still technically static once rendered are different depending on the information supplied when rendered and choices made by the user and as such are dynamic. The homepage carousel is dynamic and can be changed without reloading the page. This was implemented with bootstrap carousel. The live alert button that displays an alert on button click is also dynamic and implemented using bootstrap alert

ITEM 9	Reference
Include the use of templates in Node;	Handled by the EJS node module.
Brief description of implementation details:	Index.js has set the view engine to be EJS. A views folder (default folder EJS looks for) has been created to hold the EJS files and a sub folder partial to store EJS partials. Express then makes a render call to an EJS page through either get or post requests. Partials were put into use to reduce the amount of repeated code.

ITEM 10	Reference
Include error messages to provide feedback to user sin case of issues or errors;	Index.js: each app.get interacting with the database and final app.use
Brief description of implementation details:	Each time the database queries from the database. If an error in getting information from the database occurs. Detailed error information is sent to the console and the user presented with a simple error message and a return to home link. The final app.use is used if an unhandled route such http://localhost:3000/error is entered. The 404.ejs page is rendered.

ITEM 11	Reference
Connect to a database that contains relevant site information (e.g., product info, prices) using NODE (your database name should be your ATU ID);	This is handled in index.js with the MySQL module.
Brief description of implementation details:	Index.js requires the MySQL module. A connection is then made with the following details: host: 'localhost', user: 'root', password: 'root', database: 'g00246442' The connection variable using the details above then queries the database when called.

ITEM 12	Reference
Use Bootstrap version 5 via CDN.	Bootstrap version 5 stylesheets and scripts are included on each page.
Brief description of implementation details:	In the static pages login.html and loginFailed.html both are included in the head element and end of body. Each ejs page has the same but is included as a partial stored in views/partials with the names stylesheets.ejs and bootstrapScripts.ejs. Each ejs page then has include statements for each.