Web Application Development Project Submission 2025

Name: Andrew Walker

Student ID: G00472939

Date: 22/04/25

Home page/index page/start page (eg., page user should open first):Index page, shop page, login page. The home page is http://localhost.3000/ this is rendered by using index.ejs file. All files are located in the Flex-Gear folder. This page will feature a bootstrap carousel and a navigation bar.

Using the site - information: This site is an ecommerce website that is for a gym clothing line. Open app by using the http://localhost:300/. The page starts off on the home page(index page) then by clicking on the shop and login page this will lead you to the other aspects of the web design.By clicking the login page you can access the login form the credentials for this are below. After login you get redirected to the home page. In the shop select a product and fill in the form to receive your order form. This form is validated through JavaScript before the submission happens.

Project Requirements Implementation

ITEM 1	Reference
Allow the customer to enter their login details: *CSS Styling*	
Login details validated (via a login screen) before receiving a summary of the order:	
Username set to "user@123.com"	user@123.com
Password set to "pass"	pass
Brief description of implementation details:	Login details had to be validated before a customer could enter , so they could see a summary of their order. As seen above the username and password to be used for this. With the login page I used a standard HTML type style (<form>) which I implemented in my way from the references above. This was to collect a users email and password. The validation came from using the HTML5's required attribute. With the styling page it was all done with style css. This was based on style matching each page to follow design ideas and layout goals. To check the credentials, node.js was used. To verify the input (req.body.email == "user@123.com")n was used from node route/login.

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	public/script.js
	function validateForm() {
	  const name =
	document.getElementById('customerName').value.trim();
	  const email =
	document.getElementById('emailAddress').value.trim();

	const quantity =
	parseInt(document.getElementById('qty').value);
	
	if (!name || !email || isNaN(quantity) || quantity < 1) {
	alert('Please fill in all fields correctly.');
	return false;
	}
	return true;
	}

Brief description of implementation details:	This order form is on product.ejs what it uses is the validation form function. This will ensure that no field is empty and email.quantity inputs will be valid.

ITEM 3	**Reference**
Include a slideshow or carousel which displays a different image each time the page is loaded;	views/index.ejs(Bootstrap 5 carousel) <div id="carouselExampleSlidesOnly" class="carousel slide" data-bs-ride="carousel"> <div class="carousel-inner"> <div class="carousel-item active"><img src="/images/tshirt.jpg" ...></div> <div class="carousel-item"><img src="/images/tracksuit.jpg" ...></div> ... </div> </div>
Brief description of implementation details:	On the shop page I included this bootstrap 5 carousel. It has rotating images of the products for the customer to view before their purchase. Each time the page is loaded the images will change.

ITEM 4	**Reference**
Allow the user to 'purchase' items from the site;	views/products.ejs, routes/shop.js <form method="POST" action="/checkout" onsubmit="return validateForm();">

Brief description of implementation details:	With this a customer can submit the product order form. When the form is submitted the form is validated and then passed to the /checkout, this will then render a confirmation with purchase.

ITEM 5	**Reference**
Use an object or an array in JavaScript;	
Brief description of implementation details:	

ITEM 6	**Reference**

Use at least one custom module in node;	db.js
	const mysql = require('mysql2');
	const pool = mysql.createPool({...});
	module.exports = pool.promise();

Brief description of implementation details:	This db.js is a custom module that will export a configured MySQL connection. This is needed for the shop.js, it needs the database to query when it comes to a purchase.

ITEM 7	Reference

Include capability for handling post and get requests;	routes/shop.js
	router.get('/shop', async (req, res) => { ... });
	router.post('/checkout', (req, res) => { ... });

Brief description of implementation details:	This is for the GET request and is used to serve the /shop, /loging pages. Where as Post requests handles the forms end of things for login and the checkout data.

ITEM 8	Reference
Include both static and dynamic content;	Static: CSS/images in the public folder DynamicL EJS templates that is rendered from the variables in MySQL Database.
Brief description of implementation details:	With the static content you will get things like images and stylesheets for the webpage. Dynamic content is for the product listings that is gathered from the MySQl and then injected into the EJS templates. (product.ejs)

ITEM 9	Reference
Include the use of templates in Node;	views/index.ejs, views/login.ejs views/products.ejs res.render('product', { product });

Brief description of implementation details:	The templates will use EJS for the rendering of HTML with the dynamic content. Such things as product data and the user input for the website.

ITEM 10	Reference

Include error messages to provide feedback to user sin case of issues or errors;	
	```
	if (!name || !email || !product || !quantity) {
	  return res.send("fill in all fields");
	}
	...
	if (email !== 'user@123.com') {
	  return res.send("Invalid login.");
	}
	```

Brief description of implementation details:	This happens when the server-side routes will provide an error message, for when a user's login or form validation fails to work with the wrong details.

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);	db.js + shop.js const [product] = await db.query('SELECT * FROM product');
Brief description of implementation details:	This connects to MySQl using my student number (G00472939). Having this will retrieve the product information needed, which is used in the rendering in the shop.

ITEM 12/13	**Reference**
Use Bootstrap version 5 via CDN cart.ejs	index.ejs, product.ejs <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
Brief description of implementation details:	This was used in all the ejs files for design and styling. The Bootstrap 5 included via the CDN. Without this the web pages would not have the matching styles, created in styles CSS. The cart is stored in req.session.cart which enables persistence. Dynamic products, this is fetched from products table in my SQL. Has add to cart feature which adds products to cart and then final a view cart feature to see whats purchased.

Additional information:

All files structured in /public, /routes, /views

Fully tested on localhost:3000

References for sites used for help:

Item 1:

MozDevNet (no date) *<form>: The form element - HTML: Hypertext markup language: MDN</form>*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/form (Accessed: 22 April 2025).

W3schools.com (no date) *How To Create a Login Form*. Available at: https://www.w3schools.com/howto/howto_css_login_form.asp (Accessed: 22 April 2025). #

(No date) *Express routing*. Available at: https://expressjs.com/en/guide/routing.html (Accessed: 22 April 2025).

Item 2:

W3schools.com (no date a) *JavaScript Form Validation*. Available at: https://www.w3schools.com/js/js_validation.asp (Accessed: 22 April 2025).

MozDevNet (no date b) *Client-side form validation - learn web development: MDN*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Forms/Form_validation (Accessed: 22 April 2025).

Item 3:

Mark Otto, J.T. (no date) *Carousel*, *· Bootstrap v5.0*. Available at: https://getbootstrap.com/docs/5.0/components/carousel/ (Accessed: 22 April 2025).

Official CDN of bootstrap and font awesome (no date) *Official CDN of Bootstrap and Font Awesome ·*. Available at: https://www.bootstrapcdn.com/ (Accessed: 22 April 2025).

Item 4:

MozDevNet (no date) *<form>: The form element - HTML: Hypertext markup language: MDN</form>*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/form (Accessed: 22 April 2025).

MozDevNet (no date c) *Express tutorial part 6: Working with forms - learn web development: MDN*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs/forms (Accessed: 22 April 2025).

Item 5:

MozDevNet (no date d) *Working with objects - javascript: MDN*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_objects (Accessed: 22 April 2025).

MozDevNet (no date b) *Array - javascript: MDN*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array (Accessed: 22 April 2025).

Item 6:

Creating node.js modules (no date) *npm Docs*. Available at: https://docs.npmjs.com/creating-node-js-modules (Accessed: 22 April 2025).

MYSQL2 (no date) *npm*. Available at: https://www.npmjs.com/package/mysql2 (Accessed: 22 April 2025).

Item 7:

(No date a) *Express routing*. Available at: https://expressjs.com/en/guide/routing.html (Accessed: 22 April 2025).

MozDevNet (no date d) *Express tutorial part 4: Routes and controllers - learn web development: MDN*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs/routes (Accessed: 22 April 2025).

Item 8:

Serving static files in Express (no date) *Express*. Available at: https://expressjs.com/en/starter/static-files.html (Accessed: 22 April 2025).

Using template engines with Express (no date) *Express*. Available at: https://expressjs.com/en/guide/using-template-engines.html (Accessed: 22 April 2025).

Item 9:

(No date a) *EJS*. Available at: https://ejs.co/ (Accessed: 22 April 2025).

Sev, C. (2021) *How to use EJS to template your node application*, *DigitalOcean*. Available at: https://www.digitalocean.com/community/tutorials/how-to-use-ejs-to-template-your-node-application (Accessed: 22 April 2025).

Item 10:

(No date a) *Express error handling*. Available at: https://expressjs.com/en/guide/error-handling.html (Accessed: 22 April 2025).

MozDevNet (no date f) *Window: Alert() method - web apis: MDN*, *MDN Web Docs*. Available at: https://developer.mozilla.org/en-US/docs/Web/API/Window/alert (Accessed: 22 April 2025).

Item 11:

MYSQL2 (no date a) *npm*. Available at: https://www.npmjs.com/package/mysql2 (Accessed: 22 April 2025).

W3schools.com (no date a) *W3Schools Online Web Tutorials*. Available at: https://www.w3schools.com/nodejs/nodejs_mysql.asp (Accessed: 22 April 2025).

Item 12:

Mark Otto, J.T. (no date b) *Introduction*, *· Bootstrap v5.0*. Available at: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (Accessed: 22 April 2025).

Iteam 13

MYSQL2 (no date) npm. Available at: https://www.npmjs.com/package/mysql2 (Accessed: 22 April 2025).

Express.js, 2024. Express - Node.js web application framework. [online] Available at: https://expressjs.com/ [Accessed 21 May 2025].

EJS, 2024. Embedded JavaScript templates. [online] Available at: https://ejs.co/ [Accessed 21 May 2025].

npm, 2024. express-session. [online] Available at: https://www.npmjs.com/package/express-session [Accessed 21 May 2025].