# CSE 340 – Outcome Mastery Report

Name: Edwell Kakunguwo

GitHub: URL: https://github.com/Edwell22/cse340-/tree/main

Deployed website URL: https://cse340kings.onrender.com

## Instructions:

*For each of the course outcomes listed below, include 2-3 examples of the way you have demonstrated this outcome. Each example should include:*

* *A reference to the specific file/function you have created.*
* *A 2-3 sentence description of what you did.*

## Outcome 1: Develop to current web frontend standards of validity and practice.

## Examples:

1. Example: Ensured box model consistency throughout the CSS file and created a responsive navigation menu.
   1. File/Function: styles.css
   2. Description: Implemented the box-sizing property with the value "border-box" for all elements, including pseudo-elements, ensuring consistent box model calculation and preventing layout issues caused by different box-sizing defaults. Styled the #navigation ul element to display a responsive navigation menu with a black background, centered items, and evenly spaced links. Used flexbox and media queries to adjust the layout for different screen sizes, providing a seamless user experience across devices.
2. Example: Implemented a function to handle the selection of classifications for inventory items and exported the function for use in other modules.
   1. File/Function: script.js
   2. Description: Created the function handleClassificationSelect() to handle the selection of car classifications in the inventory. The function retrieves the classification select element, checks if a selected classification is available (passed as a variable from the server-side), and sets the value of the classification select element accordingly. Exported the handleClassificationSelect() function using module.exports to make it accessible to other modules. This allows the function to be imported and used in other parts of the application, promoting code reusability and modular development practices.

## Outcome 2: Use variables, arrays, functions, and control structures in server code.

Examples:

1. Example: Used middleware functions for session management and authentication.

a. File/Function: server.js (lines 21-39)

b. Description: Implemented session management middleware using `express-session` to store session data and handle user authentication. Used control structures to configure session options, such as creating a session store with `connect-pg-simple` and setting session secret and name. Also utilized middleware functions like `connect-flash` and `cookie-parser` for additional functionality.

2. Example: Created a route for handling inventory-related requests.

a. File/Function: server.js (line 39)

b. Description: Implemented a route for handling inventory-related requests using `app.use` and the `inventoryRoute` module. This allows for the execution of specific server code when a request is made to the "/inv" endpoint. Control structures are used to define the route and invoke the corresponding functions from the `inventoryRoute` module.

3. Example: Implemented error handling routes and middleware.

a. File/Function: server.js (lines 47-63)

b. Description: Created routes for handling errors and displaying error pages using `app.use`. Implemented middleware functions to handle errors, including a custom error handler function that renders error pages based on the error status and message. Utilized control structures to define these error routes and error handling logic, ensuring proper error management within the server code.

## Outcome 3: Develop web applications that implement common design patterns.

1.Example: Implemented the Model-View-Controller (MVC) design pattern.

a. File/Function: index.ejs

b. Description: Organized the code in the index.ejs file using the MVC design pattern. The code separates the presentation logic (View) from the data and business logic (Model and Controller). The messages displayed using <%- messages() %> indicate the usage of the Controller to retrieve and pass dynamic data to the View for rendering.

2.Example: Utilized the Single Responsibility Principle (SRP) in the Delorean Reviews section.

a. File/Function: index.ejs

b. Description: Applied SRP by separating the presentation of Delorean reviews and their ratings. Each review is displayed with its corresponding rating, encapsulating the responsibility of displaying reviews and ratings separately. This promotes code modularity and maintainability by having separate sections for different aspects of the web page.

## Outcome 4: Design and use relational databases for CRUD interactions.

1. Example: Inserted a new record into the "account" table.

a. File/Function: assignment02.sql (lines 3-9)

b. Description: Executed an SQL query to insert a new record into the "account" table. The query specifies the values for the "account\_firstname," "account\_lastname," "account\_email," and "account\_password" columns. This demonstrates the creation (C) operation in CRUD (Create, Read, Update, Delete) interactions with the database.

2. Example: Updated the "account\_type" for a specific record in the "account" table.

a. File/Function: assignment02.sql (lines 11-14)

b. Description: Executed an SQL query to update the "account\_type" column for a record in the "account" table. The query uses the WHERE clause to specify the condition for the update, in this case, the "account\_id" is checked against a specific value. This demonstrates the update (U) operation in CRUD interactions with the database.

## Outcome 5: Validate data (client-side and server-side) appropriate to the task.

Examples:

1. Example: Implemented server-side data validation for user registration.

a. File/Function: account-validation.js (validate.registrationRules)

b. Description: Defined data validation rules using the `express-validator` library for user registration. The validation rules include checking that the "account\_firstname" and "account\_lastname" fields are not empty strings, the "account\_email" field is a valid email address and does not already exist in the database, and the "account\_password" field meets the specified complexity requirements. This ensures that the data provided by the user during registration is valid and meets the required criteria.

2. Example: Checked login data and returned errors if validation fails.

a. File/Function: account-validation.js (validate.checkLoginData)

b. Description: Implemented server-side validation for login data. The function checks if the "account\_email" field is a valid email address and if the "account\_password" field meets the specified complexity requirements. If any validation errors occur, the function collects the errors and renders the login page with the corresponding error messages. This helps to ensure that only valid login data is accepted and processed.

## Outcome 6: Demonstrate the skills of a productive team member (such as solving problems, collaborating with others, communicating clearly, fulfilling assignments, and meeting deadlines.)

Examples:

1. Example: Resolved an issue with the error.ejs template.

a. File/Function: error.ejs

b. Description: Identified and fixed a problem in the error.ejs template. The code checks if the "title" variable exists and if it does, it is displayed. If the "title" variable is not available, the code redirects the user to the homepage ("/"). This ensures that the error page behaves correctly by either displaying a custom title or redirecting to the homepage if the "title" is not provided.

2. Example: Collaborated with the team to implement error handling.

a. File/Function: N/A (team collaboration)

b. Description: Worked closely with team members to implement error handling in the application. This involved discussing requirements, identifying possible error scenarios, and designing a solution. By effectively communicating and collaborating with team members, we were able to fulfill the assignment of implementing error handling within the specified deadline. This demonstrated the ability to work as a productive team member and contribute to the overall success of the project.