**Project Deliverable 2 – Documentation and Coding**

**Faculty Name:** Information Technology

**Module Code:** ITECA3-12

**Module Name:** Web Development and e-Commerce

**Content Writer:** Marc Kishinkwa

**Copy Editor:** Kyle Keens

|  |  |
| --- | --- |
| **Student Name** | JARRYD ROBSON |
| **Student Number** | WFQ2FNX46 |
| **Project Title** | ITECA DELIVERABLE 2 |
| **Submission Date** | 2025-06-06 |

**Table of Contents**

[2.1 Introduction 3](#_Toc200114882)

[2.2 Prototyping 4](#_Toc200114883)

[Home Page 4](#_Toc200114884)

[Customer Site 5](#_Toc200114885)

[Seller Tools 8](#_Toc200114889)

[Admin Dashboard 14](#_Toc200114893)

[2.3 Designing 19](#_Toc200114897)

[2.4 Coding 23](#_Toc200114899)

[PHP ……………………………………………………………………………………………………………………………..23](#_Toc200114900)

[HTML. 24](#_Toc200114902)

[JavaScript 26](#_Toc200114903)

[CSS……………………………………………………………………………………………………………………………….27](#_Toc200114904)

[MySQL 29](#_Toc200114905)

[2.5 Conclusion 29](#_Toc200114906)

# Introduction

Click for -> [Github](https://github.com/Jayrbsn/ecommerce-platform)

Click for -> [ITECA\_PLATFORM](http://51.21.251.55/ecommerce-platform/index.php)

This project entails the creation of a C-2-C (Customer-to-Customer) online commerce platform where users can purchase and exchange different goods online. The site includes two core components: the customer-facing website and the administrative backend used to manage users, product, and orders.

It has been implemented following the responsive design paradigm to be compatible with desktops, tablets, and mobile phones. The system includes functionality for customer registration, product browsing, shopping cart management, and dummy checkout. Sellers are able to add, update, and remove their own products, while administrators have full control over all users, products, and orders.

Proper role-based access control ensures that each user type can only access authorized functionalities. The platform was built using PHP and MySQL for the backend, with HTML, CSS, Bootstrap, and JavaScript (including jQuery) used for the frontend.

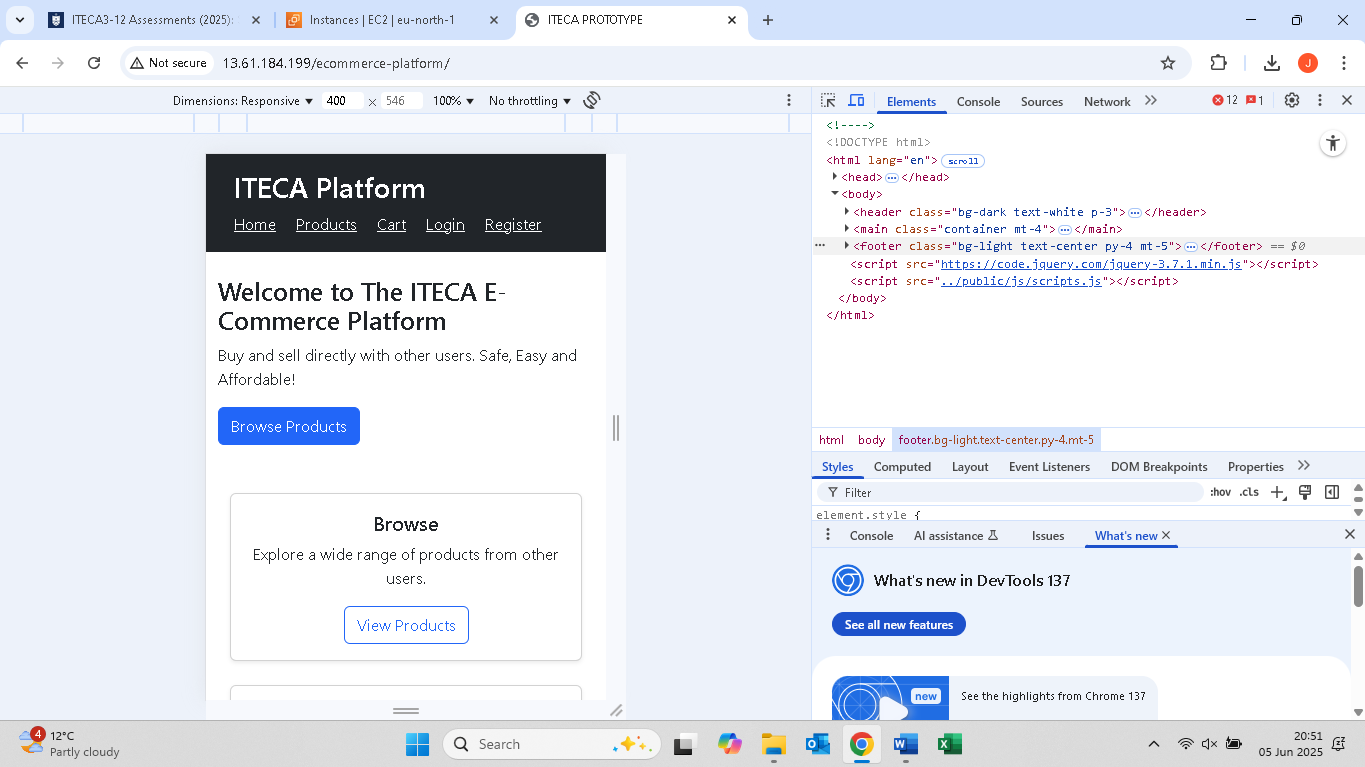
The final solution was deployed to an AWS EC2 instance, meeting the hosting requirement. In addition, visual artifacts such as CRC cards, EER diagrams, and use case diagrams were created to support the design and development process. The system was also populated with dummy data to demonstrate full functionality and reporting capabilities.

# Prototyping

# Home Page

A screenshot of a computer

AI-generated content may be incorrect.

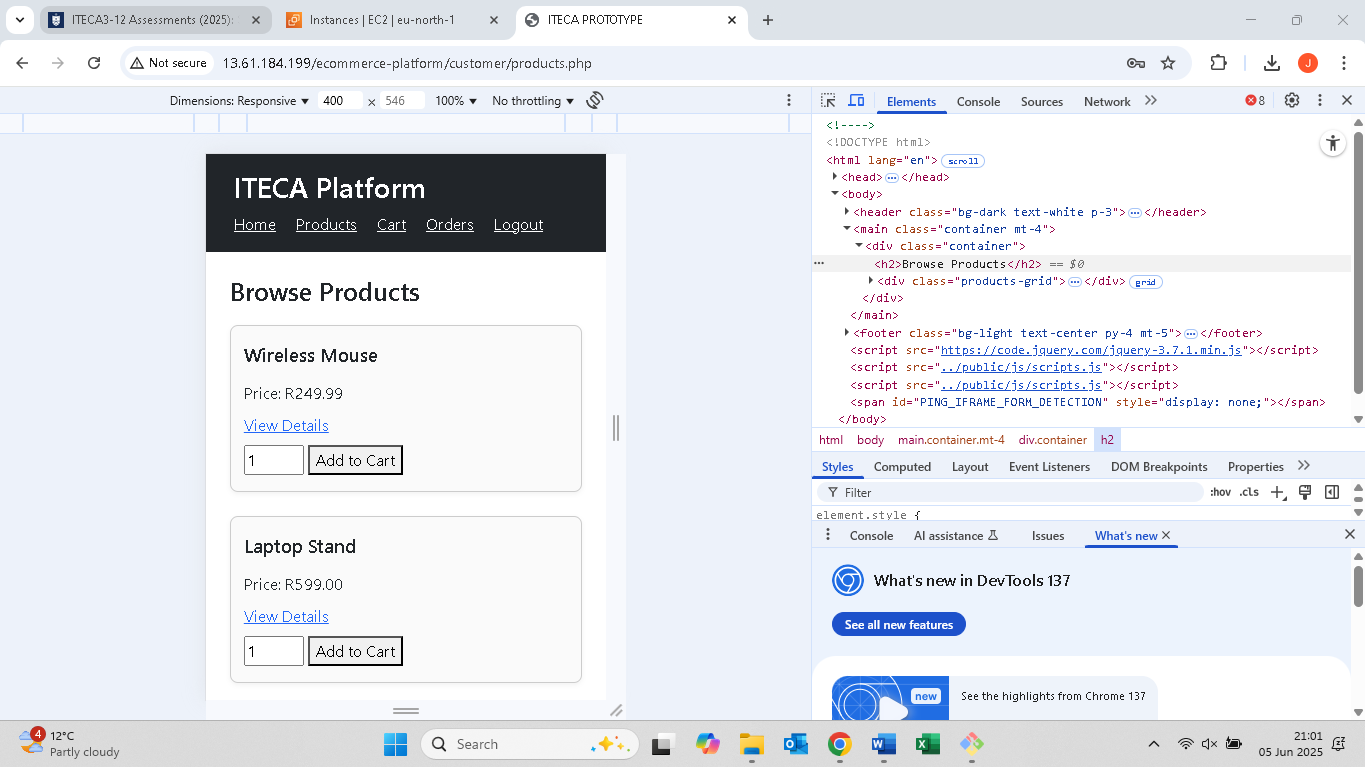


# Customer Site

Browse

A screenshot of a computer

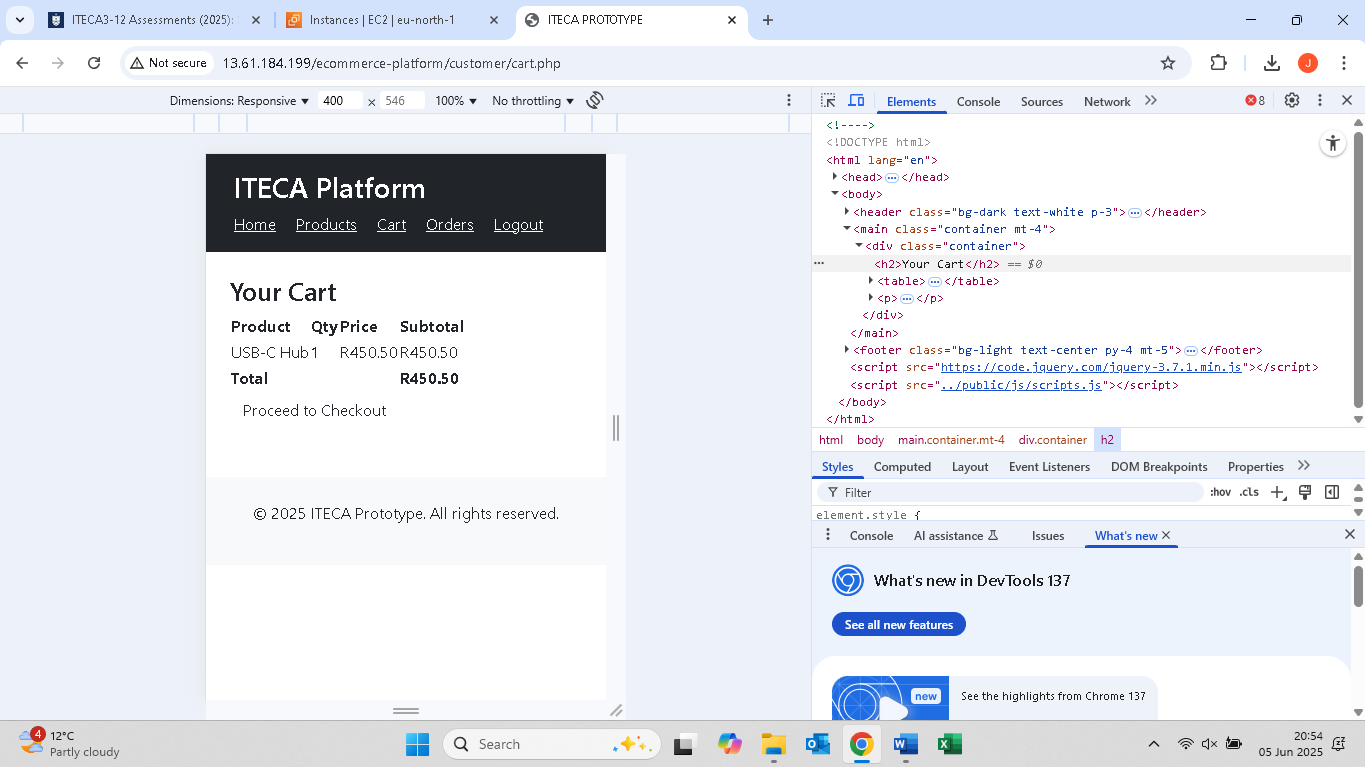
AI-generated content may be incorrect.



Cart

A screenshot of a computer

AI-generated content may be incorrect.



Checkout

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

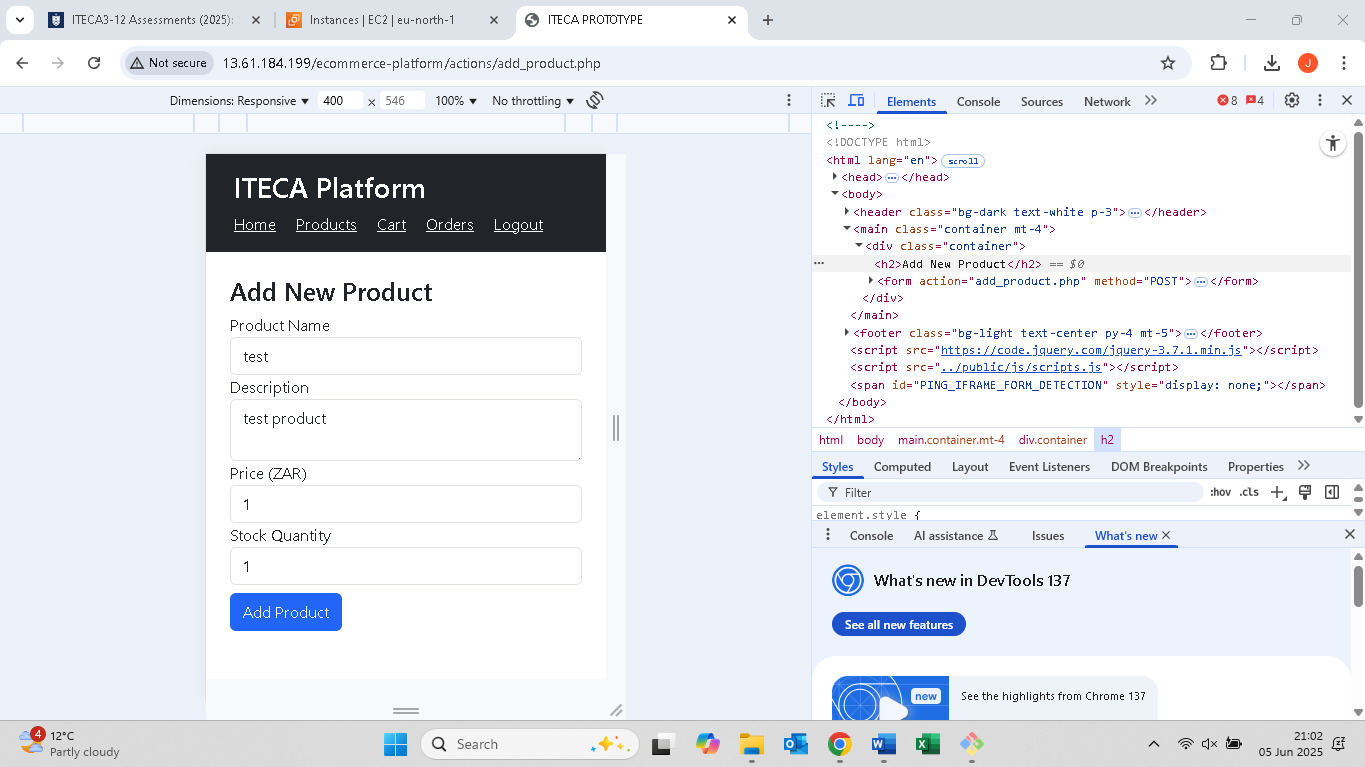
AI-generated content may be incorrect.

# Seller Tools

Add Product

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Edit Product

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Delete Product

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# Admin Dashboard

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Manage Users

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Manage Products

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Manage Orders

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

View Report

A screenshot of a computer

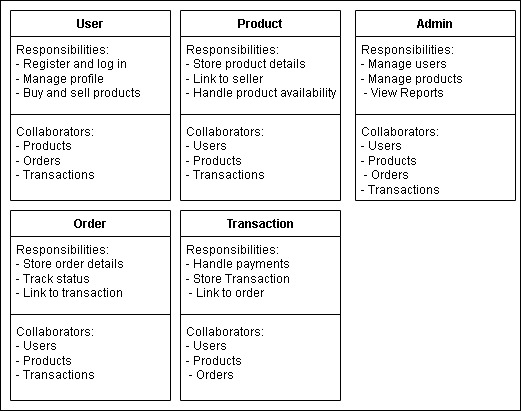
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# Designing

### Class Responsibility Collaborator (CRC) cards

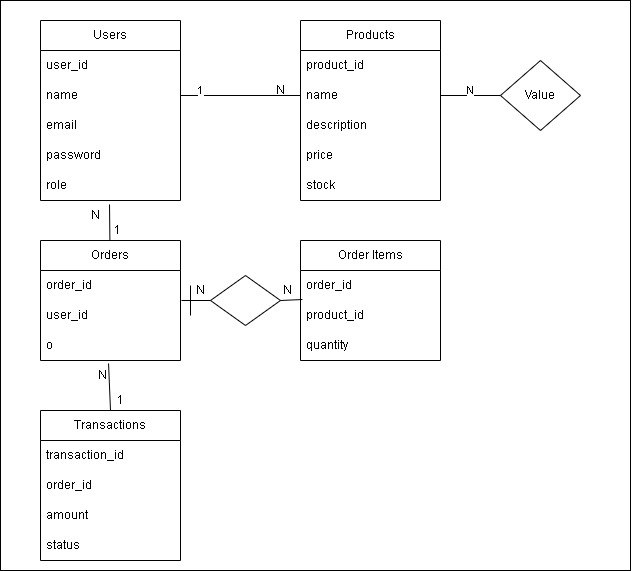


Context Diagram

A diagram of a software system

AI-generated content may be incorrect.

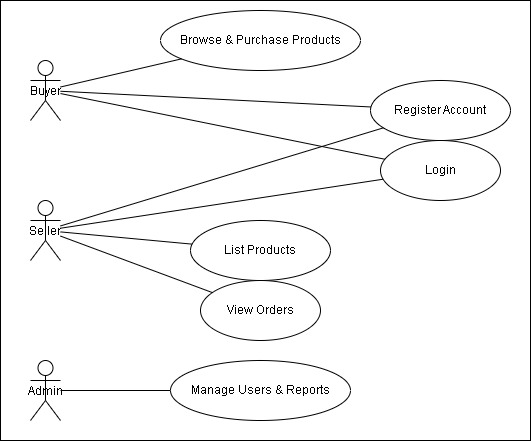
EERD



Data Flow Diagram

A diagram of a process

AI-generated content may be incorrect.

Use Case Diagram

Database Design  
  
A diagram of a product

AI-generated content may be incorrect.

# Coding

# PHP

# 

Sample of the PHP code used to handle user authentication: It first initiates a session to allow tracking of user data between various pages. It then includes the database configuration file to connect to the database. The email and password are initialized as empty strings, while the `$errors` array is initialized as empty to hold validation errors. When the form is submitted in `POST` mode, the script grabs and sanitizes the input provided by users by removing unnecessary whitespaces. It then checks whether email or password is missing; in this case, an error is added in the `$errors` array. If both are present, the script prepares the query to look for a user with email as the input provided, with the email value safely bound to the query before execution and finally stores the result to be validated further.

<?php

session\_start();

require\_once '../config/db.php';

$email = $password = '';

$errors = [];

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

    $email = trim($\_POST["email"]);

    $password = trim($\_POST["password"]);

    if (empty($email) || empty($password)) {

        $errors[] = "Both fields are required.";

    } else {

        $stmt = $conn->prepare("SELECT user\_id, name, email, password, role FROM users WHERE email = ?");

        $stmt->bind\_param("s", $email);

        $stmt->execute();

        $stmt->store\_result();

# HTML



Sample of some HTML (with PHP) code: This creates a product card on the e-commerce site, showing product details dynamically. It first displays the product name, accounting for encoding to avoid security flaws. The price is kept to two decimals for uniformity. Depending on the user’s role and whether he owns the product, various choices are presented. If the user is selling and owns the product, he can edit or delete it using respective action links. If the user is a buyer, he only sees the "View Details" link and a form where he can add the product to his cart, mentioning the quantity.

<div class="product-card">

                    <h3><?php echo htmlspecialchars($row['name']); ?></h3>

                    <p>Price: R<?php echo number\_format($row['price'], 2); ?></p>

                    <?php if ($role === 'seller' && $row['user\_id'] == $user\_id): ?>

                        <a href="../actions/edit\_product\_seller.php?id=<?php echo $row['product\_id']; ?>" class="btn btn-sm btn-warning">Edit</a>

                        <a href="../actions/delete\_product.php?id=<?php echo $row['product\_id']; ?>" class="btn btn-sm btn-danger" onclick="return confirm('Are you sure?')">Delete</a>

                    <?php else: ?>

                        <p><a href="/ecommerce-platform/customer/product\_detail.php?id=<?php echo $row['product\_id']; ?>">View Details</a></p>

                        <form action="../actions/add\_to\_cart.php" method="POST">

                            <input type="hidden" name="product\_id" value="<?php echo $row['product\_id']; ?>">

                            <input type="number" name="quantity" value="1" min="1" style="width: 60px;" required>

                            <button type="submit">Add to Cart</button>

                        </form>

                    <?php endif; ?>

                </div>

# JavaScript

A computer screen shot of a program

AI-generated content may be incorrect.

Sample of JavaScript code that listens for the DOMContentLoaded event so that it only executes after the page is fully loaded. Once an an item is successfully added to a shopping cart the script adds a dynamic notification message ("Item added to cart!") in the form of a success alert. It is placed in the top-right corner of the screen with a static position so that it is visible. After 3 seconds, the message is gone, and the 'added' parameter is removed from the URL with the help of history.replaceState(), so that the notification doesn't return upon reload. This provides a good user experience with a temporary confirmation without polluting the URL or having to be dismissed manually.

window.addEventListener('DOMContentLoaded', function () {

  const urlParams = new URLSearchParams(window.location.search);

  if (urlParams.get('added') === '1') {

    const msg = document.createElement('div');

    msg.textContent = "Item added to cart!";

    msg.className = "alert alert-success";

    msg.style.position = "fixed";

    msg.style.top = "20px";

    msg.style.right = "20px";

    msg.style.zIndex = "1000";

    msg.style.padding = "10px 20px";

    msg.style.borderRadius = "5px";

    msg.style.boxShadow = "0 2px 8px rgba(0,0,0,0.2)";

    document.body.appendChild(msg);

    setTimeout(() => {

      msg.remove();

      urlParams.delete('added');

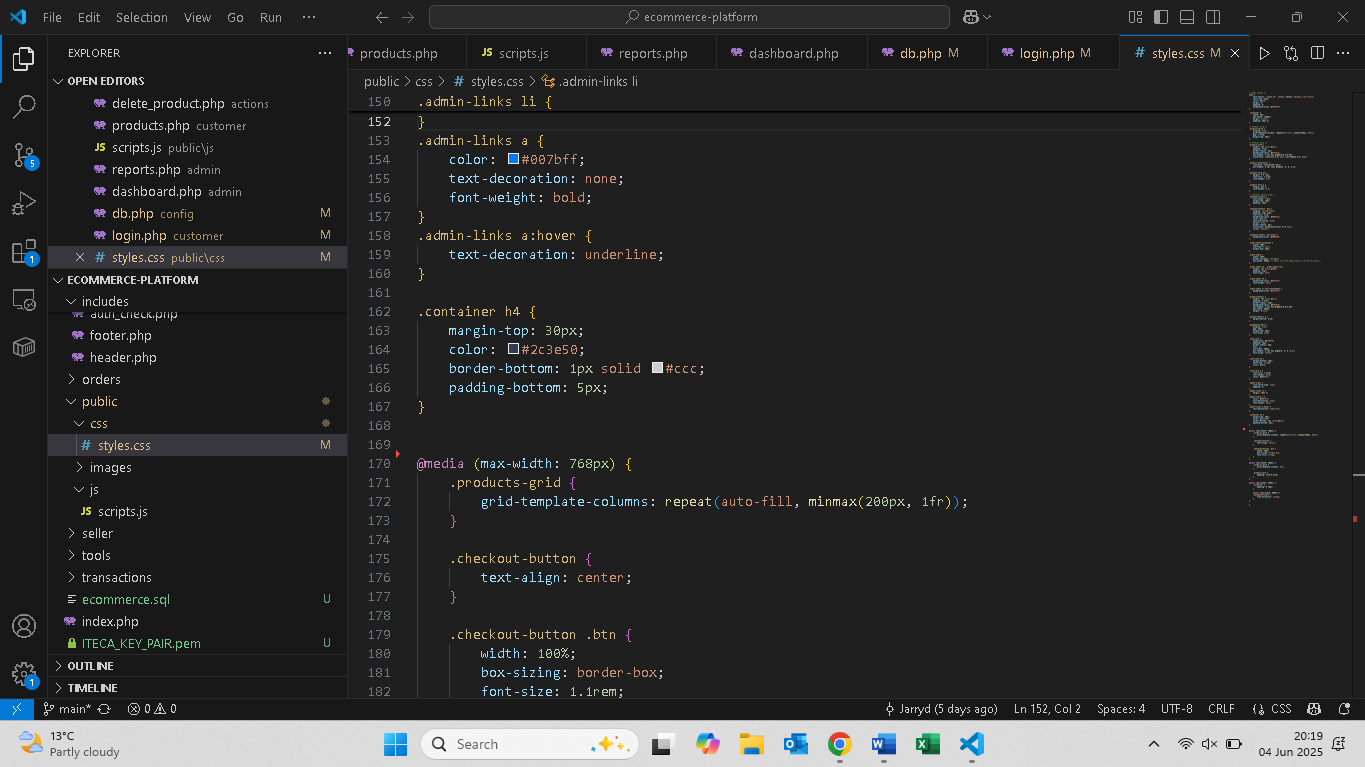
      history.replaceState(null, "", window.location.pathname);

    }, 3000);

  }

});

# CSS



Sample of CSS code starting with rule `.admin-links a:hover` that adds an underline on hovering over admin links to ensure improved visibility of navigation. The `.container h4` class is applied to the inner header elements of a container, adding a 30-pixel top margin and some text color. There is also a bottom border and bottom padding. See also the responsive code under the `@media (max-width: 768px)` block where the styles are modified for smaller screens. Inside this rule, `.products-grid` scales the layout through a grid automatically adapting column numbers while having a minimum of 200 pixels. Also inside this rule, `.checkout-button` is horizontally aligned in cases of a 768 pixels or smaller width of the screen to provide improved alignment on mobile devices. In this manner, usability and appearance are both improved with different sizes of screens.

.admin-links a:hover {

    text-decoration: underline;

}

.container h4 {

    margin-top: 30px;

    color: #2c3e50;

    border-bottom: 1px solid #ccc;

    padding-bottom: 5px;

}

@media (max-width: 768px) {

    .products-grid {

        grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));

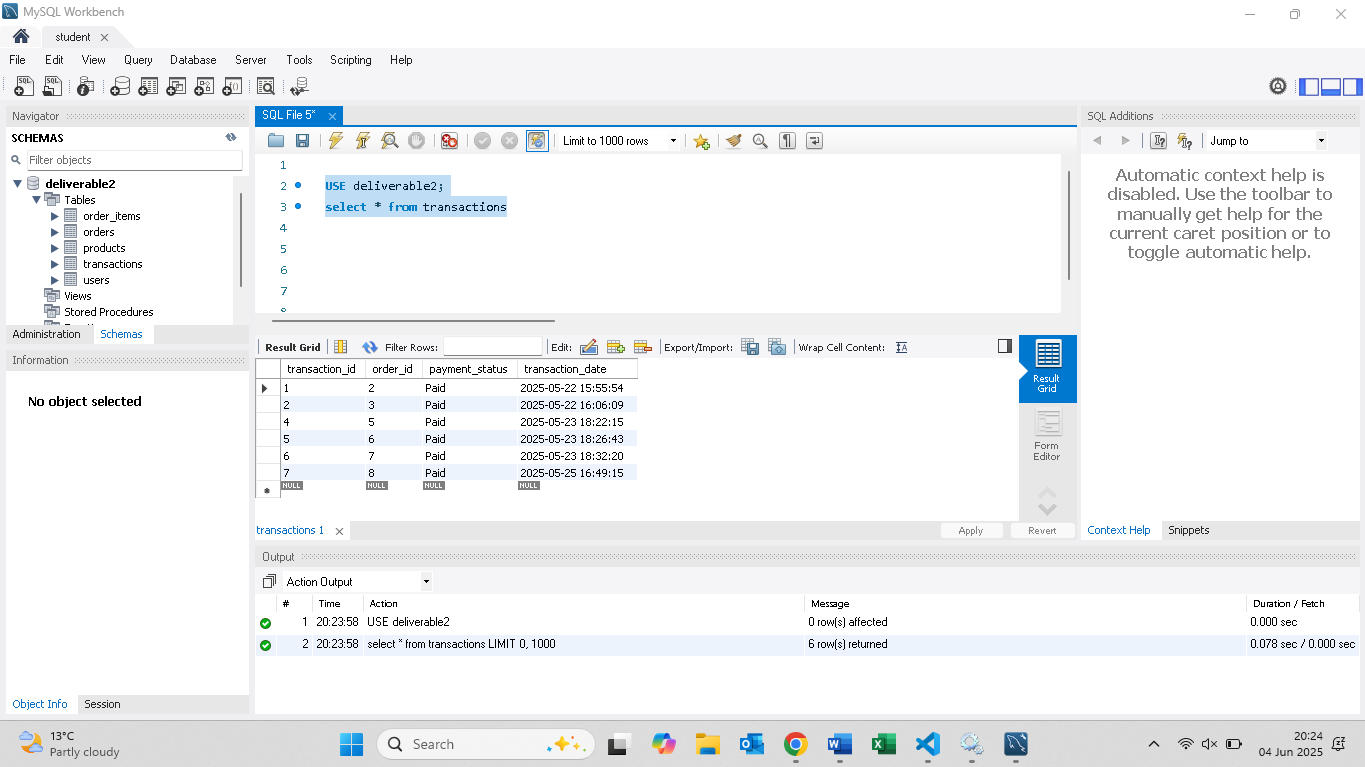
    }

    .checkout-button {

        text-align: center;

    }

# MySQL



Sample of MySQL query which is self-explanatory. The query is selecting all rows form the ‘transactions’ table for display under the deliverable2 database. *This is very short and simple code so I will just leave a screenshot and not paste the actual code text below.*

# Conclusion

Development and implementation of the e-commerce site successfully accomodated the requirements laid out in the project brief. The site provides buyers and sellers with a secure and easy-to-use interface, as well as providing administrators with complete control over site operations. Major features such as user authentication, product management, and order management, as well as reporting tools, were implemented, along with responsive design across multiple devices.

The platform was efficiently and effectively created using appropriate software design methods, such as diagrammatic modeling and structured database design. The deployment using AWS EC2 makes it possible to remotely access the system to facilitate demo and testing. The project demonstrates the integration of front-end and back-end technologies effectively to deliver a resolution of the actual problem faced in digital business.