ECE 9065 – Web Application Development

Assignment #4 – Project Report

Online Computer Store and Stock Tracking System:

Group Number:24

Team Members:

Bhavya Champaneri(bchampan@uwo.ca)
Nadia Khosravany (nkhosra5@uwo.ca)
Vigneshwaren Sunder(vsunder2@uwo.ca)

ECE 9065 – Web Application Development

Assignment #4 – Project Report

ECE 9065 – Web Application Development

Assignment #4 – Project Report

---

1. Project Overview

1.1 Project Description

The **Online Computer Store and Stock Tracking System** is a comprehensive web application designed to streamline the process of browsing, purchasing, and managing stock for a variety of computer-related products.
The project has two main components:

1. **Online Computer Store**: A user-facing interface for viewing and purchasing products with features like a shopping cart, simulated checkout, and user authentication.
2. **Stock Tracking System**: A back-end interface for administrators to manage inventory levels, update stock, and monitor stock availability.

1.2 Functional Objectives

- **User-Friendly Platform**: Enable customers to browse, search, and view detailed product information effortlessly.
- **Shopping Features**: Provide an interactive shopping cart with dynamic subtotal updates and simulated checkout functionality.
- **Secure Authentication**: Offer role-based login functionality for both customers and administrators, ensuring secure access.
- **Stock Management**: Allow administrators to manage inventory by tracking stock levels, updating product information, and viewing low-stock alerts.
- **Simulated APIs**: Integrate simulated services for product reviews, payments, and user authentication to enhance functionality.

1.3 Technologies Used

- **Front-end Framework**: React, Tailwind CSS, Vite
- **Back-end Framework:** Node.js with Express.js**.**
- **Database**: MongoDB for managing products, Carts and Users Data.
- **Version Control**: Git (hosted on GitHub).

---

2. Project Scope and Planning

ECE 9065 – Web Application Development

Assignment #4 – Project Report

2.1 Scope and Requirements

Main Scope

The project is divided into two main parts:

- **Part 1: Online Computer Store**
  This section focuses on providing an intuitive user interface for customers to browse, search, and purchase products online.
- **Part 2: Stock Tracking System (ADMIN PANEL)**
  This part is designed for administrators to manage inventory, monitor stock levels, and ensure products are available to customers.

Specific Requirements

- **Part 1: Online Computer Store**
  - Product Catalog: Display a list of products with images, details, and prices.
  - Product Search: Enable users to filter and search for products.
  - Shopping Cart: Add/remove items, display total prices, and update cart dynamically.
  - Simulated Checkout: Provide a payment simulation with order confirmation.
  - User Authentication: Allow secure registration, login, and role-based access.
- **Part 2: Stock Tracking System (For ADMINS)**
  - Inventory Management: Add, update, and delete stock information via an admin interface.
  - Stock Monitoring: Display "In Stock" or "Out of Stock" status on product pages.
  - Alerts: Notify admins of low-stock levels for timely replenishment – did not implement

2.2 Wireframes and Design

ECE 9065 – Web Application Development

Assignment #4 – Project Report

Key Pages:

- **Product Catalog**: A grid layout showcasing product images, prices, and details. Includes a search bar and filters for categories.
- **Product Details**: A detailed view with product specifications, pricing, and an "Add to Cart" button.
- **Shopping Cart**: Displays selected items, their quantities, individual prices, and total price with an option to proceed to checkout.
- **Checkout Page**: Simulated payment form with a success confirmation page.
- **Stock Management (Admin)**: Dashboard for adding, updating, and deleting products, along with stock status and low-stock alerts.
- **Design Choices**:
- **Minimalist Design**: Focus on usability with clear typography, buttons, and intuitive navigation.
- **Responsive Layout**: Ensure compatibility with various devices (mobile, tablet, desktop).

User Flow:

- Customers: Sign Up/Login > Browse > Add to Cart > Checkout.
- Admins: Sign Up/Log in > View Completed Orders/Manage Stocks > View Reports/Alerts.

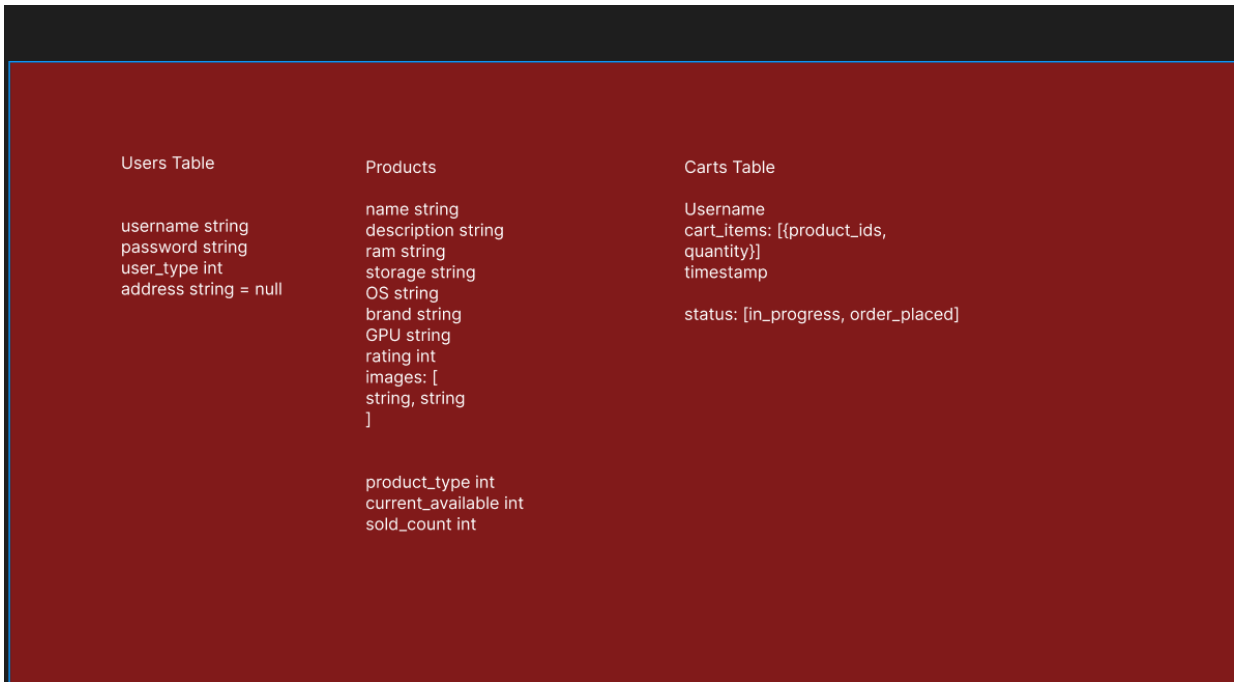
ECE 9065 – Web Application Development

Assignment #4 – Project Report

Draft Figma Wireframes

ECE 9065 – Web Application Development

Assignment #4 – Project Report



3. Implementation

3.1 Part 1: Online Computer Store

Product Catalog Management

- Administrators can manage the product catalog through a user-friendly interface:
  - **Add Products**: Admins enter product details (name, description, category, price, and stock quantity) via a form, which sends the data to the server through a POST request.
  - **Update Products**: Existing products can be updated, including changes to price, description, or stock levels, via a PUT request.
  - **Delete Products**: Products can be removed from the catalog by triggering a DELETE request to the server.
- Administrators can View the Detailed Completed Orders created by Users

Shopping Cart

ECE 9065 – Web Application Development

Assignment #4 – Project Report

- Users interact with the cart dynamically:
    - **Add Products to Cart**: Users click "Add to Cart" on a product, which updates the cart state and sends a POST request to the API.
    - **Subtotal Updates**: The subtotal updates automatically whenever items are added or removed.
    - **Checkout**: The checkout dynamically checks if the product is on stock and once checked out, the available stock is reduced.

Simulated Payment

- The checkout process provides a simulation:
    - On successful checkout, a confirmation message (e.g., "Order Successful!") is displayed, and the order data is stored in the database.

User Authentication

- **Registration**: New users sign up by providing their details (name, email, password). The server encrypts the password using bcrypt.
- **Login**: Users log in using their credentials, and the server generates a JWT for session management.
- **Role-Based Access**: Admins and customers have different roles, restricting admin-specific actions to authorized users only.

RESTful API

- **API Structure**:

    Admin Products: -

    - **GET** /v1/admin/products: Fetch all products.
    - **GET** /v1/admin/products/{id}: Fetch a product by ID.
    - **POST** /v1/admin/products: Create a product (parameters for type, details).
    - **PUT** /v1/admin/products/{id}: Update product details.

ECE 9065 – Web Application Development

Assignment #4 – Project Report

- **DELETE** /v1/admin/products/{id}: Delete a product.

Cart Management: -

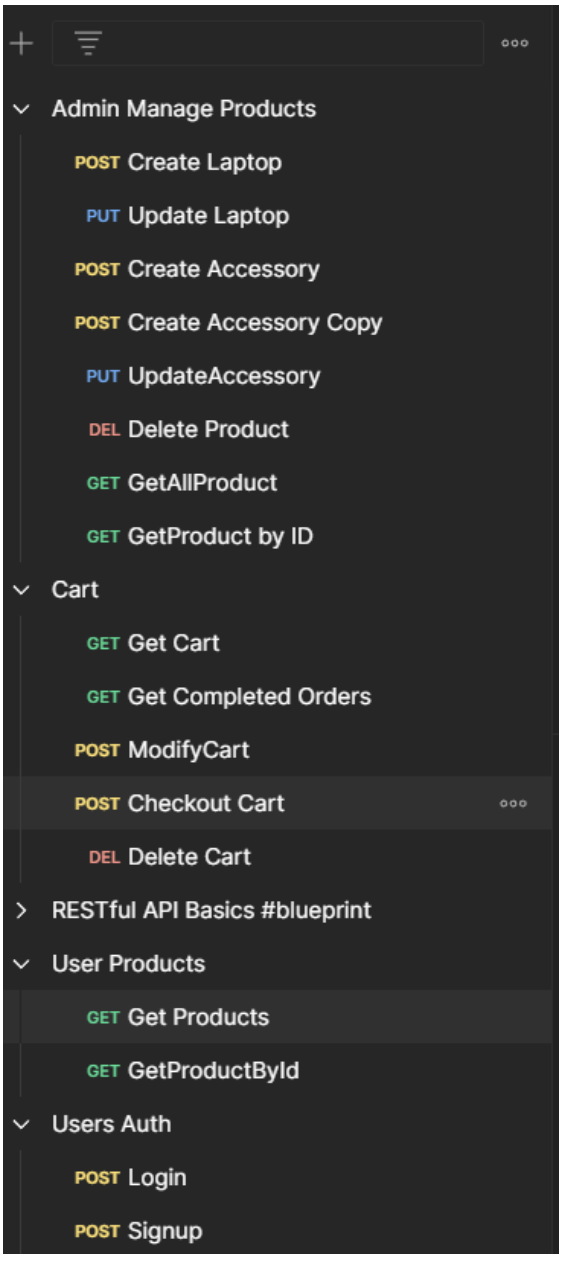
- **GET** /v1/user/cart: Retrieve cart.
- **POST** /v1/user/cart: Add or update cart items.
- **DELETE** /v1/user/cart: Delete cart.
-
- **User Authentication:**
- **POST** /v1/auth/login: User login.
- **POST** /v1/auth/signup: User signup.

Products for Users:

- **GET** /v1/products? sortby={filter parameter}&order={asc/desc}: Fetch all products (with filters).
- **GET** /v1/products/{id}: Fetch product details by ID.

ECE 9065 – Web Application Development

Assignment #4 – Project Report



ECE 9065 – Web Application Development

Assignment #4 – Project Report

3.2 Part 2: Stock Tracking System

Inventory Management

- Admins manage inventory using the admin dashboard:
  - **Stock Updates**: Admins adjust stock details for products manually via a form or dropdown.
  - **Create New Product:** Admin can add any new product (laptop or accessory) and add details for them (e.g. Type, Name, Price, Available stock, Sold Stock, Seller, Rating, ...)

Stock Display

- Stock status is displayed on product pages:
  - **In Stock**: Products available in sufficient quantities are labeled as "In Stock."
  - **Out of Stock**: Products with zero availability display "Out of Stock" and disable the "Add to Cart" button.

Admin Stock Interface (Manage Products)

- Features include:
  - A list view of all products with stock levels.
  - For each product the site displays Product Image, Product Name, Available Stock and Sold Count (to show the number of items sold).

Database Schema for Inventory

Products Table:

ECE 9065 – Web Application Development

Assignment #4 – Project Report

```
// Product Schema
const productSchema = new mongoose.Schema(
  {
    name: { type: String, required: true },
    type: {
      type: Number,
      required: true,
      enum: [0, 1], // 0 for laptop, 1 for accessory
    },
    price: { type: Number, required: true },
    priceBeforeDiscount: { type: Number, required: true },
    spec: {
      type: mongoose.Schema.Types.Mixed,
      required: true,
    },
    availableStocks: { type: Number, required: true },
    soldStocks: { type: Number, default: 0 },
    seller: { type: String, required: true },
    releaseDate: { type: Date, required: true },
    rating: { type: Number, default: 0, min: 0, max: 5 },
  },
  { timestamps: true }
);

const Product = mongoose.model("Product", productSchema);

module.exports = Product;
```

- **name, description, ram, storage, OS, brand, GPU** (string): Captures product details.
- **rating** (int): Enables user ratings.
- **images** (array): Stores URLs for product images.
- **product_type** (int): Categorizes products (e.g., laptops, accessories).
- **current_available** (int): Tracks available stock.
- **sold_count** (int): Helps with analytics.

Users Table:

ECE 9065 – Web Application Development

Assignment #4 – Project Report

```
const userSchema = new mongoose.Schema(
  {
    email: {
      type: String,
      required: true,
      unique: true,
      match: [/.+@.+\..+/, "Please enter a valid email address"],
    },
    address: {
      type: String,
      default: "",
    },
    password: {
      type: String,
      required: true,
    },
    userType: {
      type: Number,
      required: true,
      enum: [0, 1],
      default: 0, // 0-> User, 1-> Admin

    },
  },
  {
    timestamps: true, // Adds createdAt and updatedAt
  }
);
```

- **email** (string): Uniquely identifies the user.
- **password** (string): Encrypted for security.
- **user_type** (int): Differentiates admin (0) from customers (1).
- **address** (string, nullable): Stores user shipping address.

ECE 9065 – Web Application Development

Assignment #4 – Project Report

Carts Table:

server > model > JS cartModel.js > ...

```
const mongoose = require("mongoose");

const cartItemSchema = new mongoose.Schema({
    product_id: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "Product",
        required: true
    },
    price: {
        type: Number,
        required: true
    },
    quantity: {
        type: Number,
        required: true,
        min: [1, "Quantity must be at least 1"]
    }
});

const cartSchema = new mongoose.Schema(
  {
    email: {
      type: String,
      required: true,
      match: [/.+@.+\..+/, "Please enter a valid email address"]
    },
    cart_items: [cartItemSchema],
    status: {
      type: String,
      enum: ["active", "completed"],
      default: "active"
    }
  },
  {
    timestamps: true // Automatically adds createdAt and updatedAt fields
  }
);

const Cart = mongoose.model("Cart", cartSchema);

module.exports = Cart;
```

- **username**: Links the cart to a user.
- **cart_items** (array of objects): Tracks product IDs and quantities.
- **timestamp**: Records the cart's last modification time.
- **status** (enum): Tracks cart progress (active, completed).

Logout
There is a logout button visible in the admin page under all the features for easier access, which clears the jwt information that is stored in local storage and forces the user to navigate to login/signup page.

Reach private Routes
This React application uses react-router-dom for navigation and handles two user types: **users** (requiredUserType="1") and **admins** (requiredUserType="0"). It protects routes using ProtectedRoute, checking if a user is authenticated and authorized based on their role.

ECE 9065 – Web Application Development

Assignment #4 – Project Report

---

4. Testing and Quality Assurance

4.1 Test Plan

The testing strategy for the e-commerce website was **manual testing**, focusing on key areas to ensure functionality and performance. We also used **Postman** for testing APIs.
We tested for various edge cases with respect to add to cart functionailities when the product is out of stock.

- **User Interface**:
  - Homepage layout: Tested that the homepage displays the navigation bar, product catalog, and footer correctly across various devices and verified logo alignment, button placement, and image scaling.
  - Cart: add to cart had the wrong action tag, which we fixed.
    - Tested the edge cases when the user adds more products to cart than the available stocks.
  - Admin name wasn't showing properly, which was later fixed.
  - Under create new product page wasn't able to scroll down.
  - Navigation: Tested all menu items, links, and dropdowns to ensure correct redirection.
- **Stock Tracking**:
  - Stock status: Verified that stock decreases accurately after a successful purchase.
  - Inventory: Manually verified CRUD operations including adding and deleting.

4.2 Testing Results

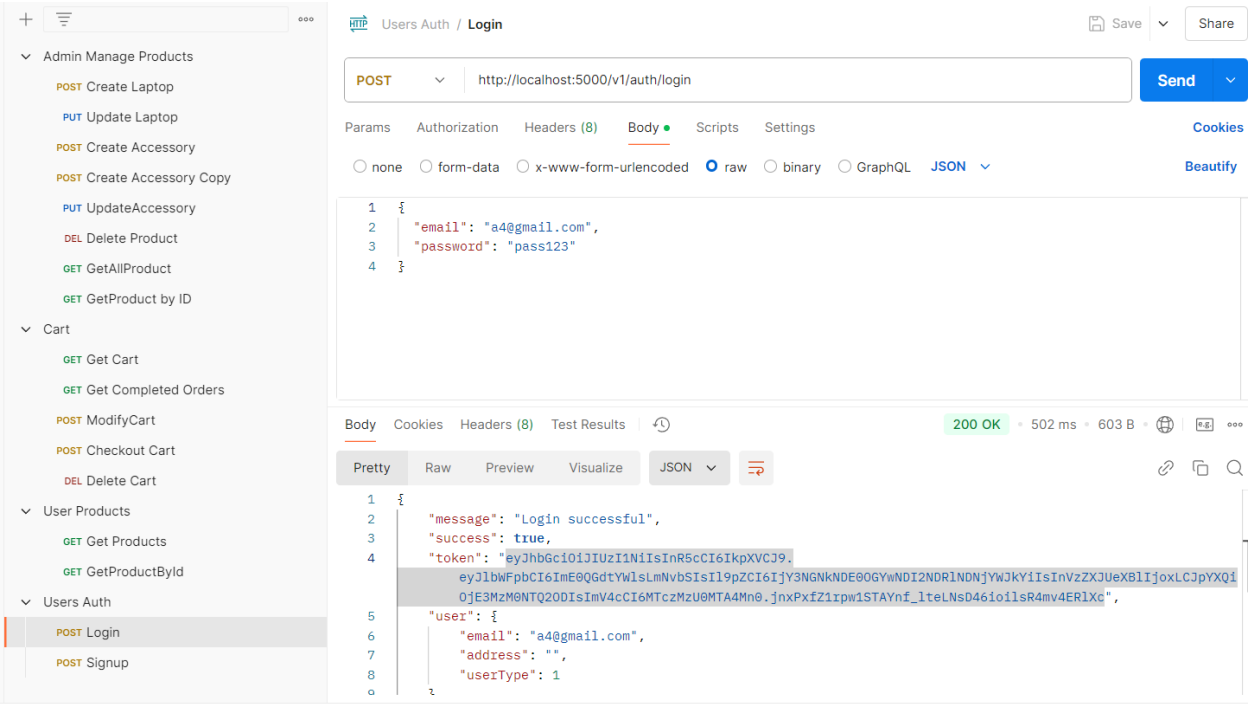
In the testing phase, errors were reported to the person responsible, and fixes were made on each of them.

Here are screen shots of Postman testing:

ECE 9065 – Web Application Development

Assignment #4 – Project Report

A list of all the APIs are visible on the left here.



5. Results and Challenges

5.1 Achievements

The project successfully accomplished several milestones, including:

- **Feature Completion**:
  - Implementation of a dynamic product catalog with search and filtering capabilities.
  - Development of a fully functional shopping cart with real-time subtotal updates and a simulated checkout process.
  - User authentication with secure role-based access control for customers and administrators.
- **User Interface Design**:
  - Responsive and accessible UI built using React and Tailwind CSS.
  - Clear navigation flow ensuring ease of use for customers and admins.
- **Stock Management System**:

    - Admin dashboard with real-time inventory tracking and low-stock alerts.
- **Automated Build of Docker:**
    - We used github workflows to build the dockerimage and push to docker repo. We then deployed these easily on kubernetes.

5.2 Challenges Faced

- **Database Integration**:
    - **Challenge**: Ensuring seamless communication between the front-end and back-end while retrieving and updating MongoDB data.
    - **Solution**: Leveraged Mongoose for schema modeling and validation, reducing errors in database queries.
- **Front-End and Back-End Compatibility**:
    - **Challenge**: Managing state consistency between the client and server, especially for the shopping cart and inventory updates.
    - **Solution**: Implemented RESTful APIs and state management on the client side using React Context API for smooth interactions.
- **Security Concerns**:
    - **Challenge**: Safeguarding user authentication and sensitive data.
    - **Solution**: encrypted passwords using bcrypt and utilized JWT for secure session management.
- **Performance Optimization**:
    - **Challenge**: Handling large product datasets and ensuring fast page loads.
    - **Solution**: Added pagination and search filtering on the back-end, reducing the data sent to the client.

5.3 Lessons Learned

Technical Insights

- **Back-End Best Practices**: Learned the importance of modular design for controllers, routers, and middleware to enhance maintainability and scalability of the application.
- **Front-End Optimization**: Developed a deeper understanding of optimizing React applications for performance and responsiveness, including efficient state management and minimizing unnecessary re-renders.
- **Database Management**: Gained experience in schema design, handling relationships between data entities, and implementing validation mechanisms to ensure data integrity.

Collaborative Insights

ECE 9065 – Web Application Development

Assignment #4 – Project Report

- **Clear Communication**: Emphasized the value of effective communication during project planning and implementation to avoid scope creep and ensure all team members were aligned on objectives.
- **Group Meetings and Design Planning**: Conducted a group meeting where we collaboratively created a **Figma design** for the entire website, outlining the user interface, navigation flow, and layout for all key pages. This helped streamline the development process and align our vision for the final product. From then on, we had regular online or in-person meeting discussing the progress and solved any problems.
- **Version Control**: Regular Git commits and pull requests facilitated efficient collaboration and code review processes, ensuring that all team member's contributions were effectively integrated.

6. Future Enhancements

1. Real-Time Inventory Synchronization

- Make periodic api calls to fetch the page related product information to update the ui and make the product go out of stock wherever required.

2. Add Pagination

- Currently we are making api calls to get all data at once. Can add pagination like ?start=0&end=10

3. Use transactions for committing multiple collections on one go

- Currently we are making commits to multiple collections sequentially after the user checksout. We can make use of mongo transaction apis to perform ACID commits.

7. References

7.1 Code References

ECE 9065 – Web Application Development

Assignment #4 – Project Report

Authentication and JWT Implementation

- **Source**: Auth0 Blog - Node.js JWT Authentication
- **Description**: Used code snippets to implement user authentication and session management using JWT.

Shopping Cart Functionality

- **Source**: MDN Web Docs - Managing State in React
- **Description**: Examples from the source were used to build the dynamic shopping cart feature.

Admin Dashboard Design

- **Source**: Tailwind Components Library
- **Description**: Tailwind CSS component designs for dashboards was used as reference (general reading)

Protected routs on react

- **Source**: https://www.robinwieruch.de/react-router-private-routes/
- **Description**: Used as reference to building private routs that only admin can access.

7.2 External Resources

- **MongoDB**
- **Axios**
- **Figma**
- **Github**
- **React Icons GitHub repository**

8. Appendices

ECE 9065 – Web Application Development

Assignment #4 – Project Report

8.1 Git Log Summary

- commit 6cdf9b295c957d56a050bb6775804856aa017e9b

```
Author: vws-codes <vsunder2@uwo.ca>
Date:   Thu Dec 5 21:48:27 2024 -0500

    fixed the a href on productcartlist

commit f90e9fdebde933fca9c98bfc7cca3c22b561ea88
Author: vws-codes <vsunder2@uwo.ca>
Date:   Thu Dec 5 19:51:02 2024 -0500

    added productDetailsPage

commit ed5a02e34a4308a2b7b5ecfbb9beff3677737b3e
Author: vws-codes <vsunder2@uwo.ca>
Date:   Thu Dec 5 17:01:48 2024 -0500

    updated gh wf to build when releasing new tag

commit 8411cac5794a58d3bf412b0e63a259c9eae11fb1
Author: vws-codes <vsunder2@uwo.ca>
Date:   Thu Dec 5 17:00:09 2024 -0500

    updated dockerfile

commit 7f9c231598364ac47c00a2f9b0a1dbb3cad0adb8
Author: vws-codes <vsunder2@uwo.ca>
Date:   Thu Dec 5 16:58:15 2024 -0500

    updates

commit 80bfa9543b4e8679d4e3a6192d3c22c2173962ff
Author: vws-codes <vsunder2@uwo.ca>
Date:   Thu Dec 5 16:56:55 2024 -0500

```

ECE 9065 – Web Application Development

Assignment #4 – Project Report

```
    minor updates

commit 751cfd2ede765636d42ca67f957c4147fd0f9e74
Author: vws-codes <vsunder2@uwo.ca>
Date:   Thu Dec 5 16:52:04 2024 -0500

    updated dockerfile

commit 60acbb19ab08a4991e6f31a8c79684b6aeac0ada
Author: vws-codes <vsunder2@uwo.ca>
Date:   Thu Dec 5 16:48:52 2024 -0500

    minor fix

commit f18f68058e26161a024a198e92e55ff7f2a7b6e0
Author: vws-codes <vsunder2@uwo.ca>
Date:   Thu Dec 5 16:42:46 2024 -0500

    added dockerfile

commit ab5b8c12f8990ba6490e82ff8f2995844a077088
Author: Vignesh (Vinn) <51945617+vignesh-codes@users.noreply.github.com>
Date:   Thu Dec 5 16:36:47 2024 -0500

    Create build-docker.yaml

commit 87cb30208ee39bc2539df9a544e419bd0791fbb3
Author: vws-codes <vsunder2@uwo.ca>
Date:   Thu Dec 5 15:53:01 2024 -0500

    added client side edge cases on cart page

commit 3bd6ddd836d04d9a3206ab0bf410c48119038aca
Author: vws-codes <vsunder2@uwo.ca>
Date:   Wed Dec 4 17:39:33 2024 -0500

```

ECE 9065 – Web Application Development

Assignment #4 – Project Report

```
    added completed orders page

commit 0f1ccc6e7d4236e0f183942801e755f41b05a08f
Author: vws-codes <vsunder2@uwo.ca>
Date:   Wed Dec 4 17:23:18 2024 -0500

    added a buggy add new product feature on admin panel

commit 3e9143c00978584bd75102a7f1e29ff6429d3b0b
Author: vws-codes <vsunder2@uwo.ca>
Date:   Wed Dec 4 16:47:43 2024 -0500

    added admin panel client-side integrations

commit fc74a4aa0e4314bb356ffd596e07d1c405fcd74f
Author: vws-codes <vsunder2@uwo.ca>
Date:   Tue Dec 3 19:36:23 2024 -0500

    completed users client-backend apis

commit 09f2dda9bf910e884abe1242688babd098e1f0de
Author: vws-codes <vsunder2@uwo.ca>
Date:   Mon Dec 2 00:28:19 2024 -0500

    completed tons of backend apis

commit 5bdd1056e8049737dbc09e46240d1e4fe447ff40
Author: vws-codes <vsunder2@uwo.ca>
Date:   Sun Dec 1 15:49:50 2024 -0500

    added client side auth

commit 6b72295db513ce2379d11625e7b4a5dc08c23859
Merge: 9f29608 9409d1d
Author: vws-codes <vsunder2@uwo.ca>
Date:   Sun Dec 1 13:26:36 2024 -0500
```

ECE 9065 – Web Application Development

Assignment #4 – Project Report

```
    Merge branch 'master'

commit 9409d1d53df245d1bfd63872ff788a8c7c81a20e
Author: vws-codes <vsunder2@uwo.ca>
Date:   Sun Dec 1 13:25:19 2024 -0500

    added latest server codes

commit 9f29608c33e401ed819eb06e0b1fe126dd0b9911
Merge: 7f82c51 7f08eda
Author: Bhavyakcwestern <bchampan@uwo.ca>
Date:   Sat Nov 30 13:08:18 2024 -0500

    Merge pull request #1 from Bhavyakcwestern/feat/dev-client-base-setup-v1

    Feat/dev client base setup v1

commit 7f08edac2a080b2bffbb11932fabdc76f047ea29
Author: vws-codes <vsunder2@uwo.ca>
Date:   Sat Nov 30 00:29:08 2024 -0500

    added admin panel navbar

commit 58b56762f6fe7ffc697973884e94b2c5c9023a1b
Author: vws-codes <vsunder2@uwo.ca>
Date:   Fri Nov 29 14:25:07 2024 -0500

    added product page, cart page and some cleans

commit 3239341d9d12b23a978bd58b112a66a598845e8d
Author: vws-codes <vsunder2@uwo.ca>
Date:   Sun Nov 17 02:06:44 2024 -0500

    added products page and designs for search options

commit 7a1c9d357fcf8287c5653a425b36d0b224912361
Author: vws-codes <vsunder2@uwo.ca>
```

ECE 9065 – Web Application Development

Assignment #4 – Project Report

```
Date:   Sat Nov 16 02:05:45 2024 -0500

    readme

commit febc706cfe20103ca65d18fbbcc84647f65fb6ce
Author: vws-codes <vsunder2@uwo.ca>
Date:   Sat Nov 16 02:04:20 2024 -0500

    eslint disabled few warnings

commit e357a82b1532bb1d5b583158bd5d54f77bbdbe98
Author: vws-codes <vsunder2@uwo.ca>
Date:   Sat Nov 16 02:02:51 2024 -0500

    initial commit

commit 7f82c518effe7fe8ca18294cee47519d158934ff
Author: Bhavya Champaneri <bchampan@uwo.ca>
Date:   Mon Nov 11 23:33:06 2024 -0500

    Added boiler plate code to Server Side  app.js
```

8.2 Additional Diagrams or Screenshots

Please see the figma diagram link attached.

https://www.figma.com/design/PMYSCah3Tr9MwEWvBHQOcs/Untitled?node-id=0-1&node-type=canvas&t=1QDXE92bOOnOECVR-0

Product Cards

https://componentland.com/component/product-card-2

ECE 9065 – Web Application Development

Assignment #4 – Project Report

---

https://pagedone.io/docs/carousel

Carts
https://flowbite.com/blocks/e-commerce/shopping-cart/#default-shopping-cart
https://dev.to/themesberg/tailwind-css-shopping-cart-component-examples-3fdn