

ECE 9065 – Web Application Development

Assignment #4 – Project Report

Online Computer Store

Group Number: 2

Team Members:

Jiaxin Yang, 251439047, jyan682@uwo.ca

Wenxiao Bu, 251451921, wbu2@uwo.ca

Yu Feng, 251405583, yfeng568@uwo.ca

ECE 9065 – Web Application Development

Assignment #4 – Project Report

1. Project Overview

1.1 Project Description

This project designed an online E-commercial platform for computers and related accessories. Generally, the system includes the following three categories:

Online Computer Store

The store has two types of roles: users and administrators. Users can modify their personal information, browse product homepages, purchase items, and simulate checkout process. Administrators can manage products and view or delete user orders.

Both roles must log in to access the system. Administrator accounts are pre-stored in the database, while all accounts created through registration are assigned user roles.

Stock Tracking System

During browsing, users can see the available stock of each item and cannot purchase items beyond the stock quantity.

Administrators can modify stocks, and if an item's inventory is set to zero, the system will automatically remove that item from all users' shopping carts.

1.2 Functional Objectives

Overall Design Objectives

The project should implement clear and comprehensive product information display, timely and friendly error notifications, and responsive layouts that adapt to different screen sizes, creating an intuitive and convenient shopping environment for users.

Online Computer Store Functionality (All Users)

Secure Access: User can only view product list before login. User can not visit admin dashboard.

Profile Management: View and modify profile information.

Product Display: Browse product homepage with search bar or filters and view product details.

ECE 9065 – Web Application Development

Assignment #4 – Project Report

Shopping: Shopping cart management and simulated checkout with confirmation email.

Order History: View historical order.

Online Computer Store Functionality (Admin Only)

Order Management: View and delete all users' historical orders.

Stock Tracking System

Stock Track for User: View product stock. Excess purchase is limited.

Stock Track for Admin: View and edit product stock.

1.3 Technologies Used

- **Front-end Framework:** React.js
 - **Back-end Framework:** Express.js
 - **Database:** MySQL
 - **Version Control:** Git (GitHub)
 - **Third-Party Services/Simulations:** ZeroTier, Postman, JWT Library, Material UI, Mock API (<https://mockapi.io/>), Tencent Mail SMTP Server, Railway, Vercel
-

2. Project Scope and Planning

2.1 Scope and Requirements

- Figure 1 illustrates the project's functional structure, with all modules implemented according to the requirements specified in Section 1.2.

ECE 9065 – Web Application Development

Assignment #4 – Project Report

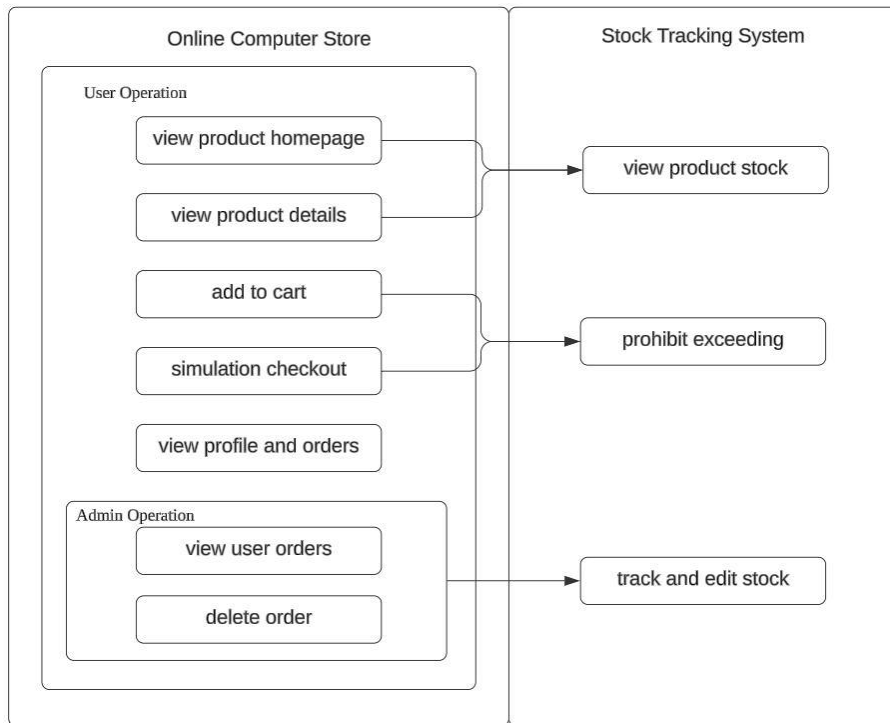


Figure 1 Project Main Scope

- **User Operation Requirements:**

Basic operation: Log in, register and log out

View product homepage: Users can browse the product list

View product details: Users can access detailed information for individual products

Add to cart: Users can add items to their shopping cart with persistent

Simulation checkout: Users can perform simulated checkout operations and receive confirmation email

View profile and orders: Users can view their profile and order history as well as edit profile

- **Admin Operation Requirements:**

ECE 9065 – Web Application Development

Assignment #4 – Project Report

Basic operation: Log in and log out

View user orders: Admin can view all users' order information

Delete order: Admin can delete user orders

- Stock Tracking System:

View product stock: Display real-time stock for both user and admin

Prohibit exceeding: Prevent purchases that exceed available stock quantities

Track and edit stock: Admins can monitor and modify product stock

2.2 Wireframes and Design

The following screenshots show the key pages of our system.

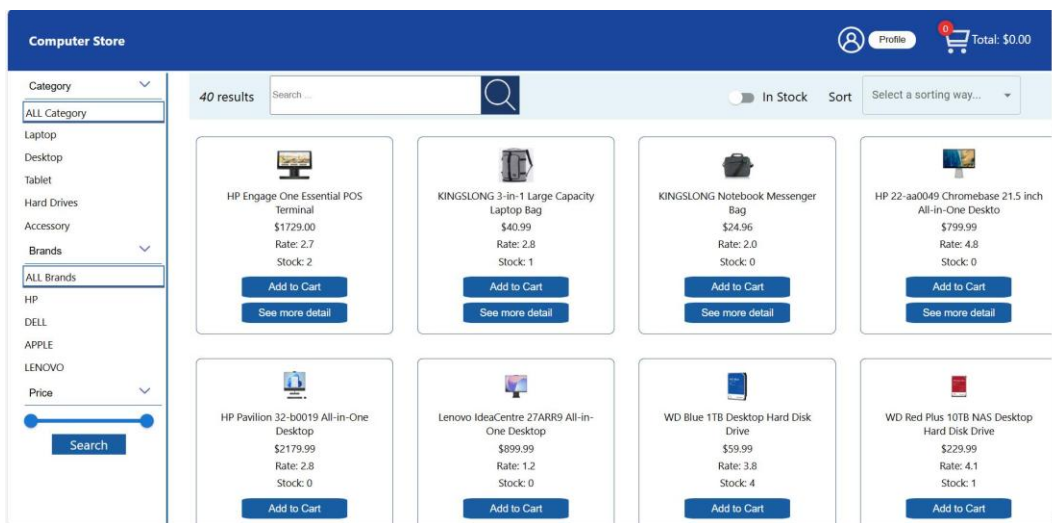


Figure 2 Product Catalog

ECE 9065 – Web Application Development

Assignment #4 – Project Report

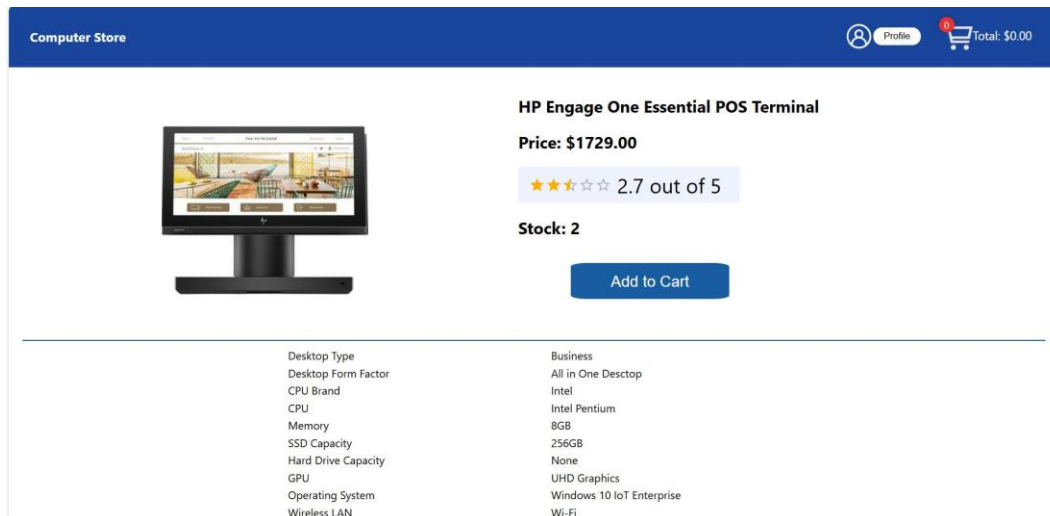


Figure 3 Product details 1



Figure 4 Product details 2

ECE 9065 – Web Application Development

Assignment #4 – Project Report

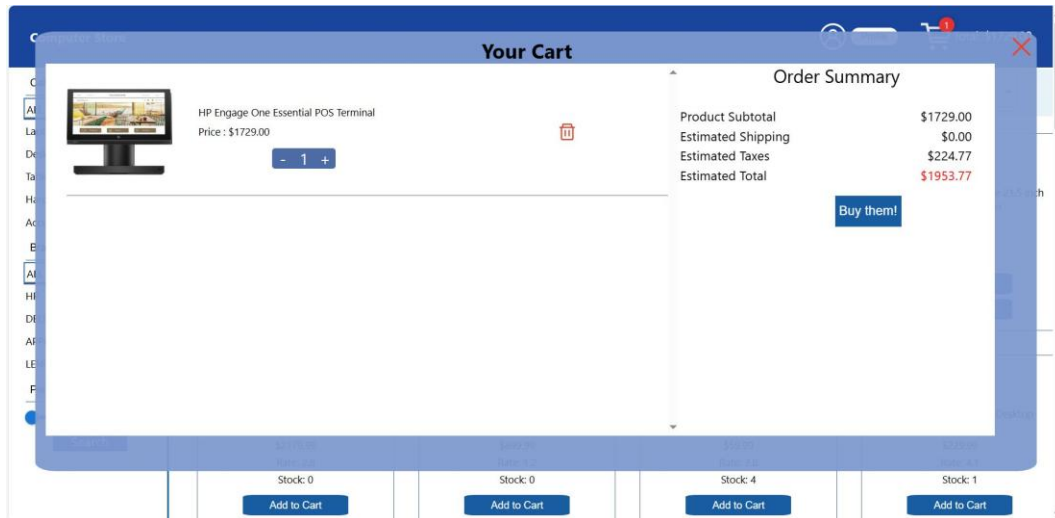


Figure 5 Shopping Cart

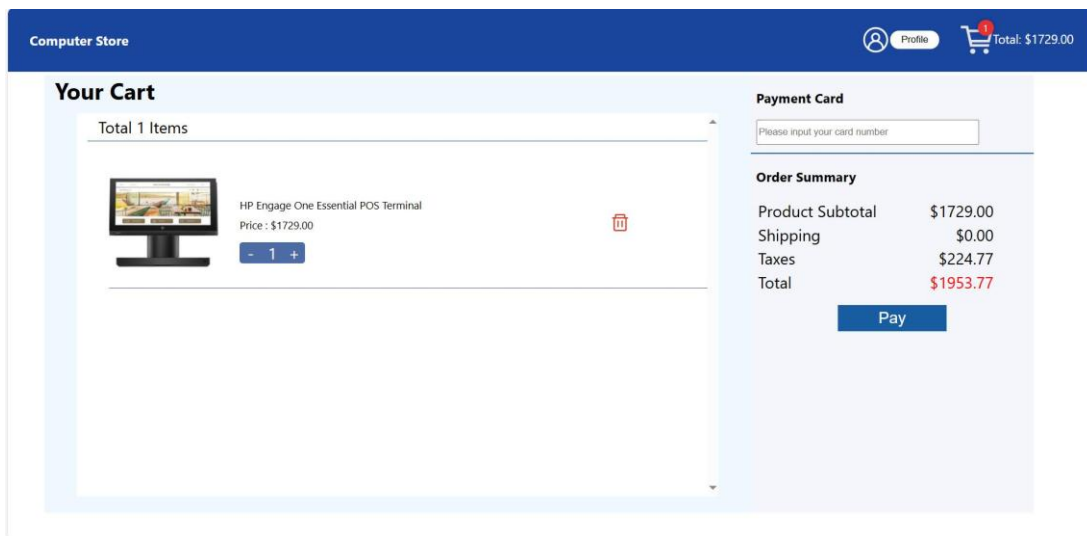


Figure 6 Checkout Page

ECE 9065 – Web Application Development

Assignment #4 – Project Report

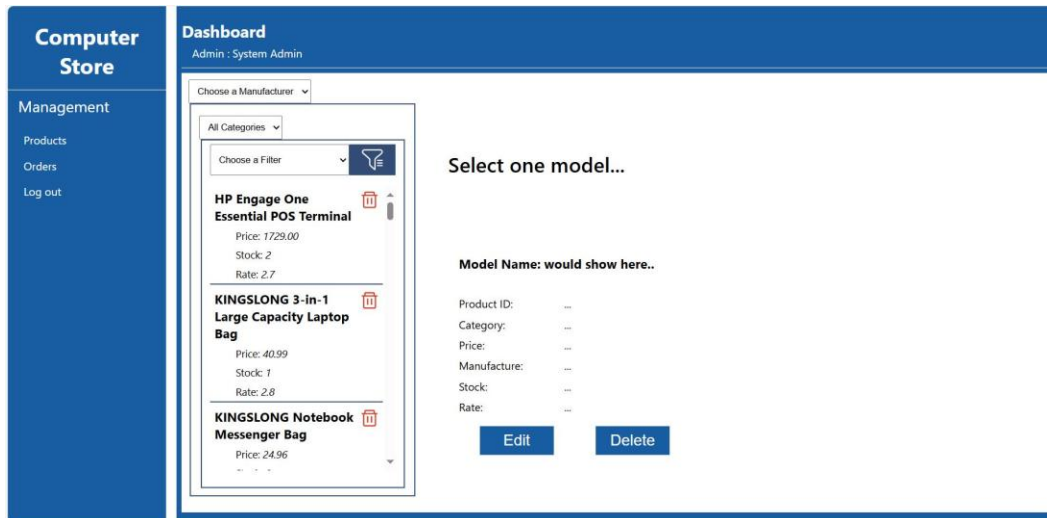


Figure 7 Admin Dashboard

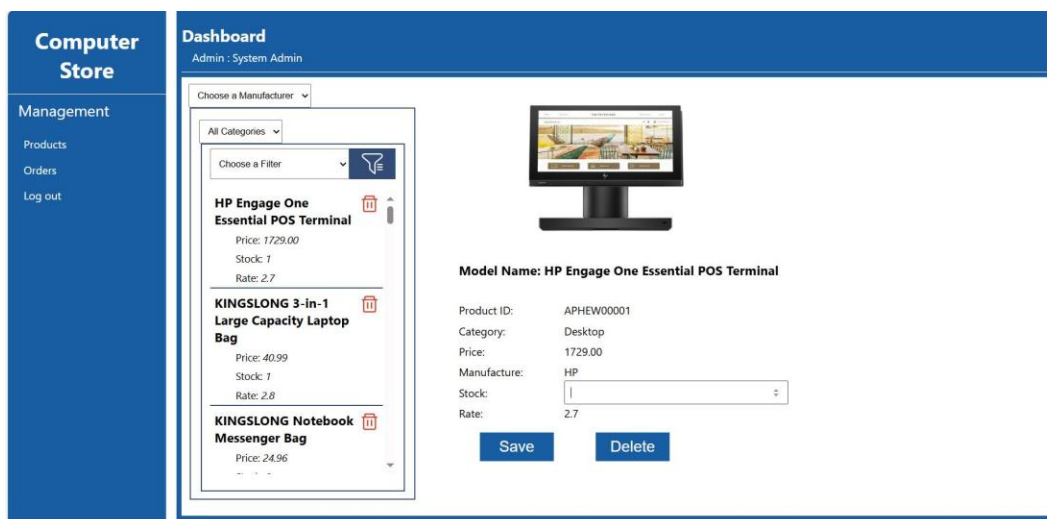


Figure 8 Admin Modify Stock

Figure 9 illustrates the complete workflow for both users and admins after logging in.

ECE 9065 – Web Application Development

Assignment #4 – Project Report

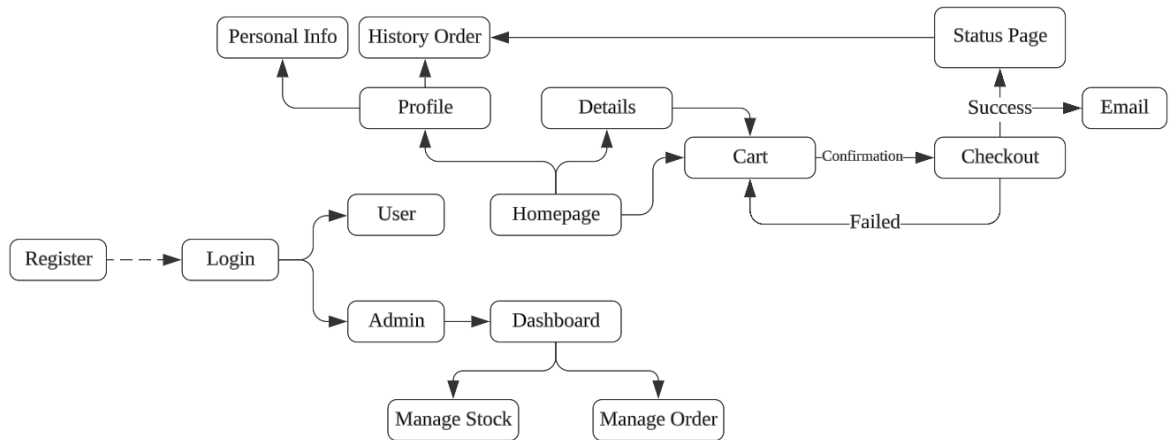


Figure 9 User Flow Diagram

Figure 10 illustrates the navigation header of our website.

Comparing Figure 10 with Figure 11, the icon in the upper right corner changes from 'profile' to 'log in', which reflects our security design:

Access Control: Unauthenticated users can only access the product homepage, all other pages require authentication.

Authentication Mechanism: After successful login, backend generates a JWT token, which enables users to access protected pages and features.

Role-Based Authorization: Admin tokens contain specific role information, granting admins additional access to the admin dashboard.

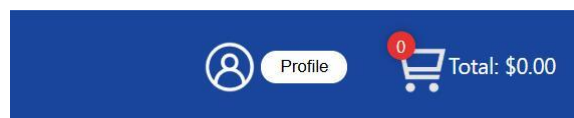


Figure 10 Nav bar after logging in

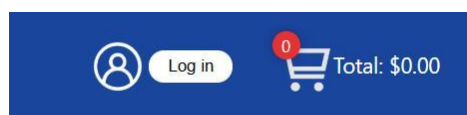


Figure 11 Nav bar before logging in

ECE 9065 – Web Application Development

Assignment #4 – Project Report

2.3 Development Environment

- Frontend: Visual Studio Code, React.js 18.3.1
- Backend: Visual Studio Code, Express.js 4.21.1

Besides, ZeroTier is used to implement the communication between front-end and back-end during development. Virtual backend address: <http://10.147.19.129:3036/>.

3. Implementation

3.1 Part 1: Online Computer Store

- **Admin Management:** Figure below shows the dashboard of admins. Admins can view products with smart filter, modify the stock (includes set zero to stock), view orders, delete orders.

OrderID	Operation
#225322L6K	Detail Delete
#184120QD8	Detail Delete
#170814D1B	Detail Delete
#165343JGZ	Detail Delete
#16521653I	Detail Delete
#1650292JR	Detail Delete
#164934LQE	Detail Delete
#164838I22	Detail Delete
#1635352AL	Detail Delete

Order ID: #225322L6K	
HP Engage One Essential POS Terminal	
Price : \$1729.00	
Quantity : 1	

User Name	
2	

Payment method	
12345678901234	

Summary	
Subtotal	\$ 1729.00
Taxes	\$ 224.77
Total	\$ 1953.77

Figure 12 Admin Dashboard

- **Shopping Cart:** As showed in homepage, the shopping cart located in the navigation bar, which enables cross-page synchronization and displays cart contents with the same component.

ECE 9065 – Web Application Development

Assignment #4 – Project Report

- **Simulated Payment:** For simulated payment, the backend verify the stock and user's purchase quantity, after twice check, the backend clear the cart of user and send a confirmation email to the user's email address in the profile.
- **User Authentication:** The system implements a comprehensive authentication and authorization mechanism:

1. Public Access (No Token Required):

Login page

Registration page

Product homepage browsing

Protected Routes (JWT Token Required):

2. Access requires a valid JWT token

Unauthorized attempts result in a 403 error

Users are automatically redirected to the login page on authentication failure

Admin Routes (Admin Token Required):

Access requires a JWT token with payload containing 'role=1'

Unauthorized attempts result in a 403 error

Non-admin users are redirected to the login page

3. Admin token validation ensures secure access to administrative functions

This tiered access control ensures appropriate system security while maintaining user experience for public features.

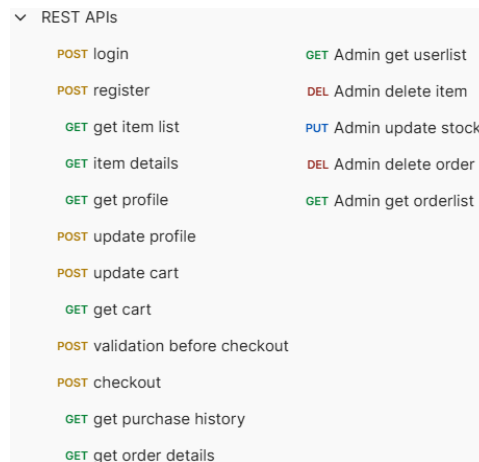
To ensure password security, the backend implements bcrypt hashing with salt before storing passwords in the database.

ECE 9065 – Web Application Development

Assignment #4 – Project Report

- **RESTful API:** We understand and make full use of RESTful request methods and write interfaces that meet the requirements.

Figure 7 shows the interface table from Postman. The meaning of each interface is detailed in Section 5.2.



REST APIs	
POST login	GET Admin get userlist
POST register	DEL Admin delete item
GET get item list	PUT Admin update stock
GET item details	DEL Admin delete order
GET get profile	GET Admin get orderlist
POST update profile	
POST update cart	
GET get cart	
POST validation before checkout	
POST checkout	
GET get purchase history	
GET get order details	

Figure 7 Interfaces Summary

3.2 Part 2: Stock Tracking System

- **Inventory Management:** The system provides comprehensive inventory management features where users can toggle the 'in stock' button while viewing product listings to filter out out-of-stock items.

To prevent overselling, users cannot add products to their cart in quantities exceeding the available stock. Furthermore, the backend performs two rounds of inventory verification during checkout to ensure the legitimacy of purchases, maintaining data consistency throughout the transaction process.

- **Admin Stock Interface:** Admins can browse product lists, order lists, and modify stocks on the dashboard. Our system does not implement low-stock alerts.
- **Database Schema for Inventory:** When stocks are modified, the backend initially only updates the stock value. Shopping cart adjustments are handled through two distinct

ECE 9065 – Web Application Development

Assignment #4 – Project Report

mechanisms: reactively when users attempt to exceed available stock, and proactively when stock reaches zero.

- In the latter case, the system automatically removes the corresponding items from all users' shopping carts, ensuring cart contents always reflect available inventory.

Field	Type	Null	Key	Default	Extra
id	varchar(15)	NO	PRI	NULL	
name	varchar(100)	NO		NULL	
description	varchar(300)	NO		NULL	
specification	text	YES		NULL	
price	decimal(10,2)	NO		NULL	
rate	decimal(10,1)	NO		NULL	
stock	int unsigned	NO		NULL	
brand	varchar(20)	YES		NULL	
ctg	varchar(20)	YES		NULL	
img_path	varchar(20)	YES		NULL	

Figure 8 Item Information Table

4. Testing and Quality Assurance

4.1 Test Plan

- **Responsive User Interface:**

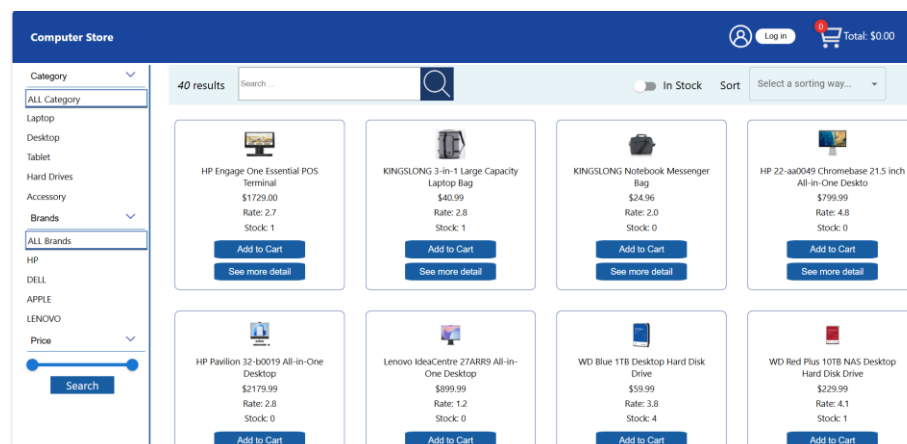


Figure 9 Homepage on laptop

ECE 9065 – Web Application Development

Assignment #4 – Project Report

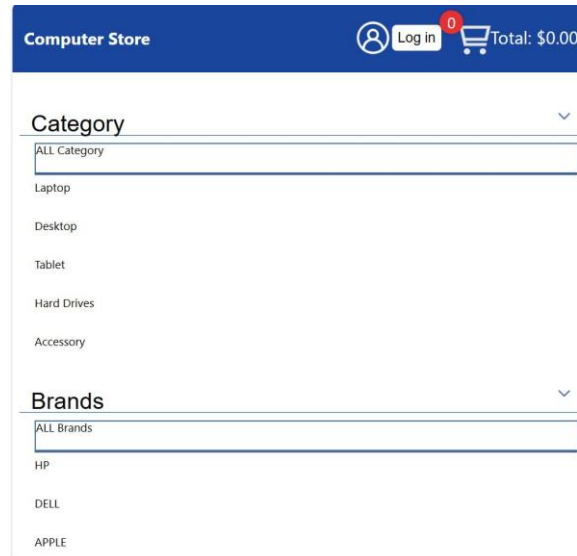


Figure 10 Homepage on Ipad-1

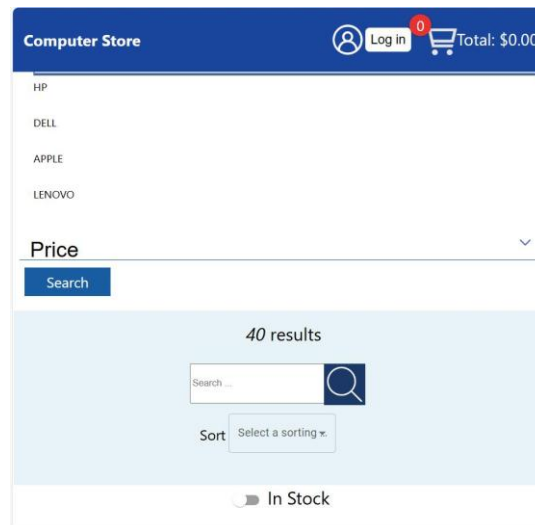


Figure 11 Homepage on Ipad-2

ECE 9065 – Web Application Development

Assignment #4 – Project Report

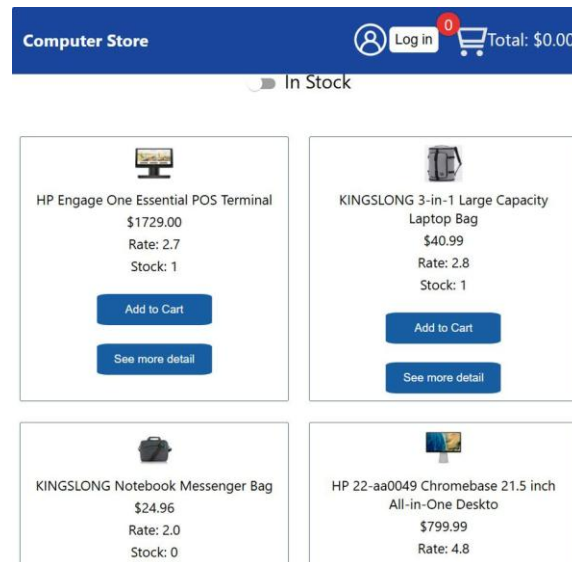


Figure 12 Homepage on Ipad-3

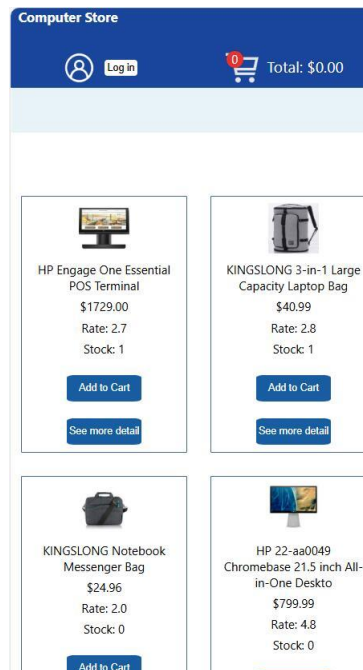


Figure 13 Homepage on iPhone

- **Functionality:**

ECE 9065 – Web Application Development

Assignment #4 – Project Report

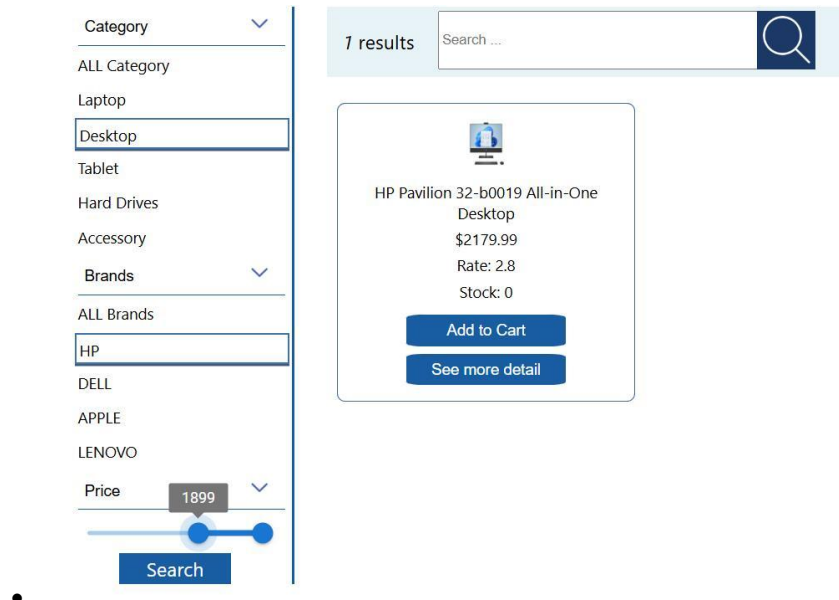


Figure 14 Product Filter

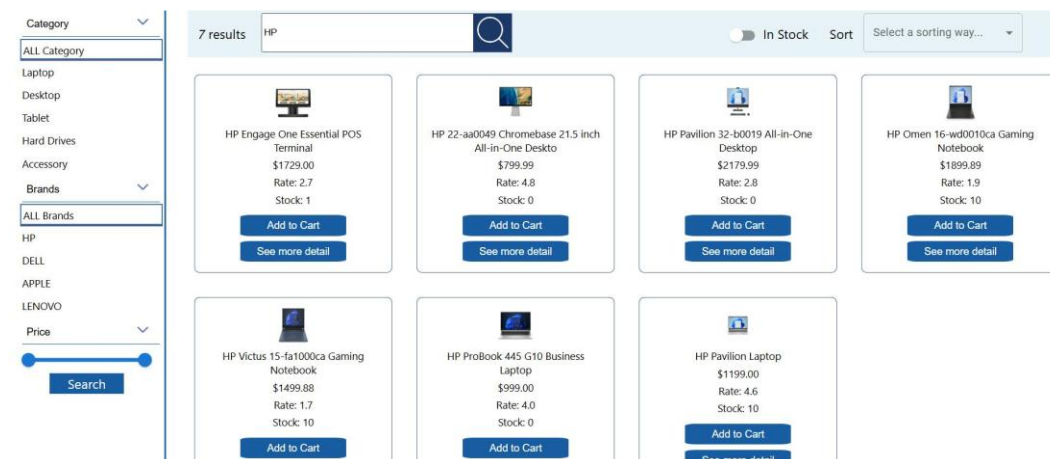


Figure 15 Product Search

ECE 9065 – Web Application Development

Assignment #4 – Project Report

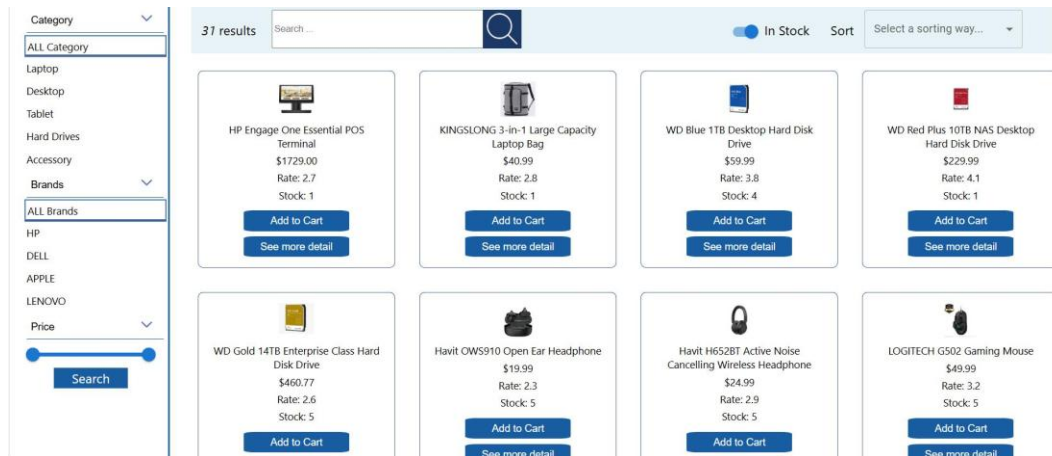


Figure 16 In stock search

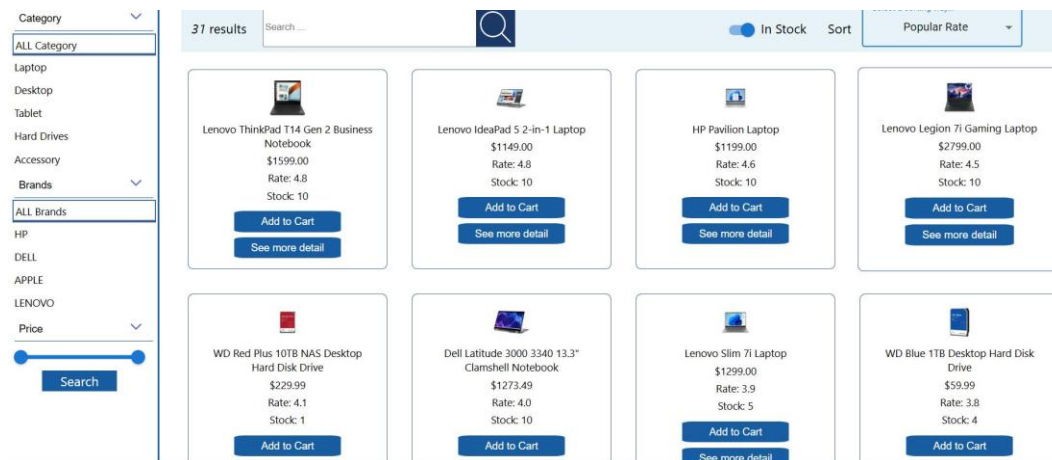


Figure 17 Sort by rating

ECE 9065 – Web Application Development

Assignment #4 – Project Report

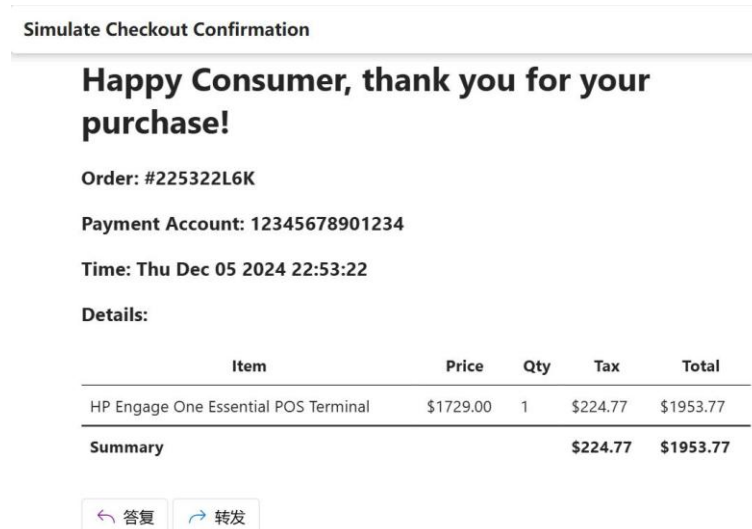


Figure 18 Simulation Confirmation Email

- **Stock Tracking**

Both users and admins can view accurate stock displayed on product cards. For admins, the system provides additional inventory management capabilities, showed in Figure 8.

These changes are reflected in real-time across the system.

4.2 Testing Results

During testing, we identified and resolved several responsive design issues across components. These included layout offsets, content overflow, and component stacking problems. Through responsive design solutions such as implementing flexbox layouts and media queries, all issues have been successfully addressed. While the initial issues were documented during developing and testing, screenshots are not included here as these problems have been resolved at the time we write this report.

5. Results and Challenges

5.1 Achievements

ECE 9065 – Web Application Development

Assignment #4 – Project Report

We have implemented all the functionality required in the rubric and instructions with security authorization and role check method.

Figure 1 in section 3.1 shows the list of API endpoints (captured from Postman), and Table below provides detailed specifications for each endpoint.

Interface	Method	Description
Login	POST	User or admin log in to system
Register	POST	User register a new account
Item/list	GET	Show a list of all products
Item/details	GET	Show detailed specifications of a product
User/info	GET	Show user's profile
User/info	PUT	Update user's profile
Cart/list	GET	Show user's former cart list
Cart/list	PUT	Update user's cart list
Cart/validation	POST	Check if the purchase quantity exceeds the stock
Checkout	POST	Proceed simulation checkout
Order/list	GET	Show all of a user's order history
Order/details	GET	Show details of a selected order
User/list	GET	Admin get all users
Item/info	DEL	Admin delete an item
Order/fullist	GET	Admin get all orders
Item/stock	PUT	Admin update stock
Order/info	DEL	Admin delete an order

5.2 Challenges Faced

1. Image Transmission

Issue: As the product homepage needs to display numerous product images, directly transmitting original image files would result in extended page loading times, negatively impacting the user browsing experience.

Solution: We designed an optimized image transmission solution. First, all images are encoded in base64 format during transmission. Additionally, we preprocess each image to

ECE 9065 – Web Application Development

Assignment #4 – Project Report

generate two versions: the original file and a compressed icon. For pages displaying multiple products, such as the homepage and cart, the backend only transmits icon version files to improve loading speed; for product detail pages where high-quality images are necessary, the original files are transmitted.

2. Shopping Cart Storage (Frontend)

Issue: Since users can view and modify cart contents from various locations including the homepage, product detail pages, and the shopping cart page, passing this data through component props became cumbersome and difficult to maintain. The cart required a unified state management solution to ensure real-time synchronization of cart data across all relevant components.

Solution: Through technical research, we found two viable solutions: `localStorage` and `useContext` in `React.js`. We opted for `useContext` as it enables parent components to provide data directly to any nested child component without prop drilling.

By placing all components that need access to cart data under a single `Context`, we achieved global state management for the shopping cart, making it possible to synchronize cart data across different pages.

3. Shopping Cart Update (Frontend and Backend)

Issue: While we need to store cart data in the database for persistence, frequent communication between frontend and backend results in high latency. If every cart operation (such as adding items, removing items, or modifying quantities) triggers immediate synchronization with the backend, it would not only increase server load but also impact users' experience.

Solution: We implemented a smart data synchronization strategy, where cart updates are not triggered by product change. Instead, the frontend monitors three key scenarios: 1. users close the cart. 2. page is about to navigate away. 3. users proceed to checkout.

Only these specific events trigger communication with the backend to persist cart data in the database, significantly reducing server load.

4. Authorization and Role Control

ECE 9065 – Web Application Development

Assignment #4 – Project Report

Issue: Only admins have the authority to view all orders and modify stocks. It is necessary to prevent malicious users from attempting to perform unauthorized operations by directly accessing URLs or other ways.

Solution: We implemented user authentication using JWT and stored user roles (0 for users, 1 for admins) in the token's payload.

Based on this, we developed a checkRole middleware for access control. For admin-only operations, this middleware performs permission verification, and immediately returns error messages when unauthorized access attempts are detected.

5. Simulation API

Issue: Due to tight development deadline, we were unable to design API documentation or follow a sequential development process, requiring parallel development instead. In this situation, it became crucial to prevent development inefficiencies caused by mismatches between front-end and back-end interfaces.

Solution: The backend uses Postman to simulate API requests, while the frontend uses Mock API to create simulated data, allowing us to independently develop and test the user interface without waiting for the real frontend or backend to be completed.

5.3 Lessons Learned

- Building frontend interfaces with React.js, becoming proficient in core concepts such as components, Hooks, and state management.
- Developing backend services using Express.js, mastering key features like route configuration and middleware implementation.
- Gaining a deep understanding of frontend-backend communication mechanisms, becoming skilled in RESTful API design and implementation.
- Establishing a comprehensive understanding of Web application architecture through hands-on practice.
- Mastering development tools like Mock API and Postman for interface simulation and debugging, improving development efficiency and collaboration quality.

6. Future Enhancements

ECE 9065 – Web Application Development

Assignment #4 – Project Report

- Login and Retrieve Password through Email

Since we have implemented the email sending functionality, we can generate verification codes in the backend and send them to specific users via email, enabling verification code-based user login and password recovery.

Challenge: Verification code expiration. For security, verification codes cannot remain valid indefinitely, it should be expired in 10 or 15 minutes. Therefore, we need to design mechanisms such as message queues or Redis cache database to track the status of these codes.

- Upload new products (Admin only)

Admins should be able to upload new products, including the product name, description, specifications, price, stock, images, and so on.

Challenge: As product specifications are of variable length, we need to design a flexible mechanism to support parameter addition and validation. Meanwhile, for product image uploads, it is also required to balance image quality with storage and loading performance.

7. References

7.1 Code References

All functional code in this project was written by ourselves. ChatGPT and Claude were utilized to generate the following auxiliary code:

Solutions for extracting raw data from tokens (backend/api/service/jwt_auth.js, checkRole function).

Solutions for bulk insertion of product data into the database (database_deploy.js).

The amount lines of these codes are 100 lines, less than 5% of our complete codes.

7.2 External Resources

ECE 9065 – Web Application Development

Assignment #4 – Project Report

- Canada Computer: <https://www.canadacomputers.com/>
 - Material UI: <https://mui.com/>
 - Mock API: <https://mockapi.io/>
 - JWT: <https://jwt.io/>
 - Railway: <https://railway.app/>
 - Tencent Mail SMTP Server: <https://mail.qq.com/>
 - ZeroTier: <https://www.zerotier.com/>
 - Postman: <https://www.postman.com/>
 - Vercel: <https://vercel.com/>
-

8. Appendices

8.1 Git Log Summary

commit 75a8a8e41e3038f30bb040b41aba1772becda25c

Author: Wenxiao <buwenxiao@outlook.com>

Date: Thu Dec 5 20:06:52 2024 -0500

implement auth check between user and admin

commit 50ba5e3f389ccea2a7bc7b95c5a67c8144a25138

Author: Wenxiao <buwenxiao@outlook.com>

Date: Thu Dec 5 18:58:11 2024 -0500

enable JWT and bcrypt for security

commit 603eab9daee636425d2a189a7f57d6f75cccea8b

Author: Wenxiao <buwenxiao@outlook.com>

Date: Thu Dec 5 18:40:02 2024 -0500

add responsive design to pages

commit fea9cba858f46d8553a2f0992f7a938f2b031150

ECE 9065 – Web Application Development

Assignment #4 – Project Report

Author: Wenxiao <buwenxiao@outlook.com>

Date: Thu Dec 5 12:06:29 2024 -0500

connect to cloud database

commit 4076d5f4d5f3974c1ee716bc4920797b6dd3c9c0

Author: Wenxiao <buwenxiao@outlook.com>

Date: Wed Dec 4 20:37:59 2024 -0500

Modified the implementation of deleting products, now the product information will not be removed

commit 8daa01f80bd00e2878e353cf0b4d5dda7faaf501

Author: Wenxiao <buwenxiao@outlook.com>

Date: Wed Dec 4 19:35:51 2024 -0500

fix bugs about getting order details and check payment validation

commit 311f249c1a73f179a3783ff9a7b9752a6b77fa39

Author: Wenxiao <buwenxiao@outlook.com>

Date: Wed Dec 4 18:47:21 2024 -0500

implement admin functions

commit d7dfe9dbda91458b39724e8be8de8b4cf0e0caae

Author: Wenxiao <buwenxiao@outlook.com>

Date: Wed Dec 4 15:29:17 2024 -0500

implemente all user functions

commit 35e18bb9743a5d768c288f4dc45d4dee28dcdf1a

Author: Wenxiao <buwenxiao@outlook.com>

Date: Thu Nov 28 23:02:18 2024 -0500

ECE 9065 – Web Application Development

Assignment #4 – Project Report

implement login and register function

commit f908cafe13e3d392947879ed20daceec8077ddf2

Author: Wenxiao <buwenxiao@outlook.com>

Date: Thu Nov 28 12:02:51 2024 -0500

add jwt authorization

commit 7b45e9d0116c555667fc25726717b409fb89f8fa

Author: Wenxiao <buwenxiao@outlook.com>

Date: Mon Nov 11 00:04:42 2024 -0500

build an example of frontend, backend and communication.

8.2 Additional Diagrams or Screenshots

Sequence diagram of Login

ECE 9065 – Web Application Development

Assignment #4 – Project Report

