CSC337 Project Report

Motivation:

Our group chose to create a full E-Commerce web application because technology and online shopping are interests, we all share. Modern online storefronts elegantly combine user interaction, data management, and responsive design, making them a perfect fit for demonstrating the concepts learned in CSC337.

We wanted a project that was both meaningful and manageable. An online tech store allowed us to explore secure authentication, session handling, database persistence, dynamic content rendering, and CRUD operations, all while building a polished experience that feels similar to a real commercial platform.

The TechEase Store aligns with our technical interests, allows for creativity in UI/UX design, and provides a practical example of how real-world web applications are structured.

Modules:

We have 3 modules: users, orders, and carts. Users hold a users login information, Our architecture uses **three main modules**:
Users, **Carts**, and **Orders**.
Each module is stored in MongoDB collections and interacts with the Node.js server through dedicated API routes.

Users Module

- Stores each user's username, email, and password.
- Handles authentication via login and registration.
- On successful login, generates a secure session token using crypto.
- Session tokens are stored in the sessions collection and used for authorization.
- Other modules reference the User ID to tie carts and orders to the correct user.

Carts Module

- Each user has a dedicated cart document created on first access.
- Stores all items the user intends to purchase but has not yet checked out.

- Contains product IDs and quantities.
- Supports adding items, removing items, updating quantities, and clearing the cart.
- Interacts with the Products module to fetch prices and images for display.

Orders Module

- Stores finalized purchases after checkout.
- Contains the purchased items, total cost, and timestamp.
- Allows users to view complete order history.
- Each item stored includes name, price, and image for easy rendering in the UI.
- Acts as a permanent record of the user's purchases.

Module Interactions

- The **Users module** authenticates the request.
- The **Carts module** retrieves, modifies, and prepares cart items for checkout.
- The **Orders module** receives data from the cart and stores the finalized purchase.

All three modules communicate through Express routes, with MongoDB acting as the shared persistent storage.

Functionalities:

The TechEase Store supports the full workflow of a modern online shopping experience. Core features include:

User Authentication

- Register new accounts using username, email, and password.
- Secure login with session tokens.
- Tokens stored in localStorage and validated on each API request.
- Protected routes require authentication.

Product Browsing

- Products stored in MongoDB with names, categories, descriptions, prices, and image paths.

- Products are dynamically loaded through /api/products.
- Fully responsive product grid with animations and images.

Shopping Cart

- Add items from product page.
- Increase/decrease quantity within the cart.
- Remove individual items.
- Cart persists per user.
- Displays automatic itemized cost and running total.

Checkout System

- Calculates total cost server-side based on product IDs and user cart.
- Creates a new order in orders collection.
- Clears the cart after successful checkout.
- Displays a confirmation page with total cost.

Order History

- Loads a user's past orders via /api/orders.
- Displays date, items, quantities, total cost, and item images.
- Items grouped by order and shown in stylized order cards.

Profile Page

- Shows user information (username, email).
- Links to order history.
- Includes logout button clearing session tokens.

Front-End Interface

- Clean, modern UI with:
    - Left-side animated navigation bar
    - HeroBox components with glassmorphic styling
    - Responsive product grid

    - Styled forms for login and registration
    - Image-backed product displays
    - Smooth fade-in animations

Technical Details:

Front-End

- HTML & CSS for structure and styling.
- JavaScript for dynamic content, event handling, and fetch requests.
- Glassmorphism-inspired theme with:
    - gradient backgrounds
    - blur effects
    - animated transitions
- Responsive design with CSS Grid and media queries.

Back-End

Node.js + Express

- Routes for user actions, product retrieval, cart management, checkout, and orders.
- Middleware (requireToken) for authentication on protected API routes.
- Static asset delivery for product images.

Database

MongoDB with the following collections:

- **users**
- **sessions**
- **products**
- **carts**
- **orders**

Each document structure was intentionally kept simple to avoid over-normalization and match the needs of the project.

Session Handling

- Upon login, server generates a 32-byte hex token using crypto.
- Token saved in sessions collection mapped to userId.
- Token stored client-side in localStorage.
- All protected API calls include token in Authorization header.
- Server validates token before granting access.

File Structure

- server.js — main Express application
- db.js — MongoDB connection helper
- resetDB.js — resets DB and seeds products
- /assets — product images
- / — HTML + CSS files for UI
- All pages directly served by Express