🎬 Final Project Demonstration Transcript

(Silent Walkthrough of Full E-Commerce Application)

Note to Reviewers:
Due to unexpected technical limitations with my PC's audio system, this recorded demonstration does not include narration. However, it visually guides viewers through each major feature of my fully developed e-commerce application, showcasing functionality, design choices, and backend integration in real time.

🛒 Project Summary: "ShopStream" E-Commerce Web Application

The goal of this capstone project was to design and implement a realistic, responsive, and secure online shopping experience that simulates an actual retail storefront. The final application—built entirely from scratch—includes core e-commerce functionality such as:

- **User registration & secure login**
- **Dynamic product browsing**
- **Persistent shopping cart**
- **Streamlined, validated checkout**
- **Order confirmation system**
- **Backend data management via MongoDB Atlas**

Tech Stack Highlights:

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Node.js, Express.js
- **Database:** MongoDB Atlas
- **Security:** JWT for session handling, bcrypt for password encryption
- **Testing Tools:** Thunder Client (for API testing)

💻 Graphical User Interface (GUI) & Features Walkthrough

◇ Home Page (Home.html)

- Clean landing page introducing the brand and core navigation.
- Quick access to Login, Register, and About sections.

◇ User Authentication

Login (login.html):

- Real-time form validation for usernames and passwords (5+ characters).
- Color-coded feedback with dynamic button states.
- Loading animations to improve UX during login attempts.
- Redirects user securely upon successful login.

Registration (register.html):

- Comprehensive input validation (length restrictions, character rules).
- Real-time feedback on username availability.
- Encrypted password storage via bcrypt.

◇ Product Display & Shopping

Shop Page (shop.html):

- Products are dynamically loaded from the MongoDB Products collection using a secured GET API.
- Each product displays name, category, description, and price.
- Add-to-cart icons trigger backend updates while maintaining session-specific carts.
- Real-time cart counter badge and hover interactions enhance usability.

Shopping Cart Sidebar:

- Slide-in cart panel with item previews, total price, and quantity modifiers.
- Auto-updates upon product addition or removal.
- Cart state is persisted and protected by JWT middleware.

◇ Checkout Process

Checkout Page (checkout.html):

- Refined interface with:
    - Dropdown of 20 major U.S. cities
    - Phone input auto-formatted to (XXX) XXX-XXXX
    - Real-time field validation and feedback
- Upon validation, user submits shipping details and processes order via a secure POST request.

Confirmation Page (confirmation.html):

- Final screen presents:
    - Unique order ID
    - Summary of purchase
    - Options to print receipt or return to shop

🔐 Security & API Logic

- JWT tokens securely authenticate and authorize user sessions.
- Protected pages (shop, checkout, confirmation) verify session tokens before allowing access.
- Full API suite includes:
    - /register, /login
    - /products, /cart/add, /cart/update, /cart/remove

    - /orders for final order submissions
- Error handling is responsive, with toast-style notifications for feedback.

📦 Data Layer & Database Integration

- **MongoDB Atlas** hosts all persistent data:
    - Users
    - Products
    - Cart Items
    - Orders
- Data integrity ensured through structured Mongoose models and middleware protections.

Interesting Feature That Was Hard to Implement

The **order processing system** following the checkout process was the most challenging feature to implement, presenting significant errors and roadblocks during development. This feature required seamless integration of the shopping cart, checkout form, and order confirmation, all while ensuring data persistence and user feedback.

- **First Issue: Inability to Load Cart Items**
  - **Problem**: Initially, cart items failed to load correctly during checkout due to mismatches between the frontend and backend data structures.
  - **Resolution**: The database schema was remade multiple times to align the cart data structure with the MongoDB backend. This involved restructuring the CartItem.js model to ensure consistent data formats and updating API endpoints to handle cart data correctly.
- **Second Issue: Failure to Load Order Information**
  - **Problem**: Order details were not populating correctly on the confirmation page, often resulting in incomplete or missing data.
  - **Resolution**: Extensive debugging using browser console network logs identified issues in the POST /orders and GET /orders API routes. The authentication middleware and route handlers were reworked to ensure proper JWT validation and data retrieval. Real-time validation scripts were also refined to handle dynamic updates and prevent data loss during order submission.
- **Additional Challenges**: Other issues, though not fully recalled, included intermittent API failures and validation inconsistencies, which required iterative testing and refinement. The complexity of synchronizing cart data, validating user input, and ensuring order persistence nearly led to abandoning this feature. However, persistent debugging and schema adjustments ultimately enabled a robust, user-friendly order processing system.

Possible Enhancements and Extra Features

- **Order History Page**: Implement an order history page (as outlined in FR-010) allowing users to view past orders with details like order ID, date, items, and totals.
- **Product Search and Filtering**: Add search functionality and category filters to the shop page to enhance product discovery.
- **User Profile Management**: Introduce a user profile page where customers can update their account details, such as email or password.
- **Wishlist Feature**: Allow users to save products to a wishlist for future purchases, stored in MongoDB and accessible via a new API endpoint.
- **Email Notifications**: Integrate email confirmations for order placement and account registration using a service like Nodemailer.
- **Advanced Analytics**: Add tracking for user behavior (e.g., popular products, cart abandonment) to provide insights for the business owner.

Features Replaced/Tweaked/Removed

- **Registration and Login Validation**:
  - **Original**: Allowed single-character usernames and passwords, which posed security risks and lacked robustness.
  - **Change**: Enhanced to require a minimum of 5 characters for both fields, with alphanumeric restrictions for usernames, character limits (30 for usernames, 100 for passwords), and real-time validation feedback.
- **Payment Processing Simulation**:
  - **Original Plan**: Included a simulated payment processing step in the checkout flow.
  - **Change**: Removed due to time constraints and complexity, as it was not feasible to implement realistically in this application. The checkout process was simplified to focus on shipping information (city dropdown and phone validation), with paymentInfo made optional in the database schema.
- **Current Status**: No known bugs remain after these refinements.

Limitations

- **No Payment Processing**: The application lacks a payment processing system, limiting it to a simulated shopping experience rather than a fully operational store.
- **Limited City Options**: The checkout form restricts city selection to 20 major US cities, which may exclude some users.
- **No Order History**: While planned (FR-010), the order history feature is not yet implemented, preventing users from viewing past purchases.
- **Local Development Dependency**: The application relies on local development tools (Live Server, Thunder Client) and a fixed port (http://localhost:5000), which may complicate deployment.
- **No Multi-Language Support**: The interface and validation messages are English-only, potentially limiting accessibility for non-English-speaking users.
- **Scalability Constraints**: The current MongoDB setup and localStorage-based JWT storage may face performance issues with a large user base or high traffic.

Screencast Link: https://go.screenpal.com/watch/cTXnqKnFbmh

Github Portfolio Link: https://github.com/Kdeshun/CST-452