Goal

We want to build a basic e-commerce website. A user can browse the main page, which displays a gallery of products. They can select a particular brand, which will display the list of products sold by that brand. If they click on a product, it navigates them to a page showing more details about that product.

If they want to purchase products, they can add them to their cart. They're able to access their cart page and checkout, which would display an order summary page. There's an admin account that will be in charge of moderating these order requests.

Requirements

1. Coding
    a. Backend: Node.js Express server
    b. Frontend: HTML/CSS/JS, (optional) template engine
    c. Database: MongoDB (Mongoose)
    d. The server must be organized based on the MVC architecture.
2. Set up your Github repository and add the Training Managers as collaborators.
    a. eric.lee@beaconfireinc.com
    b. ehuang@beaconfireinc.com
3. Record a 1-2 minute video demo (no audio needed) that showcases the UI features you've implemented (no code implementations).

Features

1. Registration page
    a. Users must provide a unique username, password, and unique email (doesn't need to be real, but should follow the email format like prefix@domain).
2. Login page
    a. Users should provide their username and password to log in.
    b. If they are not logged in, users can browse products (view the home page, product page).
    c. If logged in, users are also able to view their orders, add products to cart, and checkout.
    d. If the user has an active session or valid token, and they refresh the page, they should still be logged in. (Hint: sessionStorage)
    e. Admins are not able to view the regular user home page, add products to cart, checkout, or view their own orders.
3. Navigation bar
    a. **Home**, **Cart**, **My Orders**, **Sign Up** (if not logged in), **Login** (if not logged in), **Logout** (if logged in)
4. Home page
    a. Users can browse a picture gallery of products. There should be previous & next button functionality to look through products in a row.
    b. Each brand should have its own section and gallery of products.
    c. Clicking a thumbnail navigates the user to a dynamically generated page for that product.
5. Product page

    a. Users will see the same thumbnail (optional: additional pictures) and additional information about the product.
    b. There is an add to cart section, where users can specify how many products they want, and click a button to add it to their cart.
    c. There is a product reviews section, where any user can submit a review (rating out of 5, comments).
        i. For simplicity, there is no limitation to how many reviews a user can submit, and they can submit reviews even if they haven't purchased it before.
6. Cart page
    a. Users will see all products that are currently in their cart, along with each product's information.
    b. There is a form that users can fill out to begin the checkout process. This form takes fields like name, phone number, email, shipping and billing address, payment method (credit/debit/paypal/etc.). Payment methods can be simple <option> tags.
        i. You don't need to specify payment information. There's no need to implement actual payment processing & security.
    c. When they submit their order request, the page will tell the user that their order has been submitted. By default, the order status is going to be "Pending".
    d. If users log out without placing an order and their cart has existing products, this data should be persisted. Upon the next login, their carts should be repopulated with the same products.
7. My Orders Page
    a. Users can see their order history and each product's order status (Pending, Approved, Rejected).
8. Admin page
    a. Admin logs in and has access to all checkout requests. They are able to review the order summary before approving or rejecting.
    b. They can view all existing user accounts and their order history.

NOTE:
For any features that involve user input, invalid inputs should be appropriately handled, and users should be notified via alerts or DOM manipulation.

To obtain the data, you'll have to find your own images and products. It doesn't have to contain real products or brands. Aim for 2 brands with 10 products each.

The UI's design is ***entirely*** up to you. Please focus on implementing the correct functionality to avoid spending too much time on styling. We recommend starting the project in this order:

- Data modeling (entities, relationships, schemas)
- Set up the MVC-based backend structure
- To implement each page or feature, start with backend routing and then connect it to the frontend for that page or feature.

Good Practices

1. Input validation should be handled server-side.
2. Users who access unsupported routes (/user/aijfojaiewoi) should be properly redirected.
3. Pay attention to user authorization.
    a. Users should not be able to access other users' carts, order pages, etc.
    b. Handle page navigation via directly typing in the url vs. clicking redirect links on the page.

Submission

1. On the portal, under "Week 4 Day 3" with the tag "Project", please submit a zipped file titled "**EcommerceProject- FirstnameLastname**". This must contain all files (including the video demo) except the *node_modules* directory. You may need to compress the video demo to meet the file upload size requirements.
    a. You have until **Wednesday (10/5) at 9:00 AM EST** to submit the project.
2. Be prepared to demo your project on Tuesday at 10:00AM in the Zoom meeting. Slides are not required.