Report on Building a Dynamic Frontend for a Webpage

This report outlines the process of building a dynamic frontend for my webpage, with a focus on components such as product listing, product details, a shopping cart, user info and checkout.

---

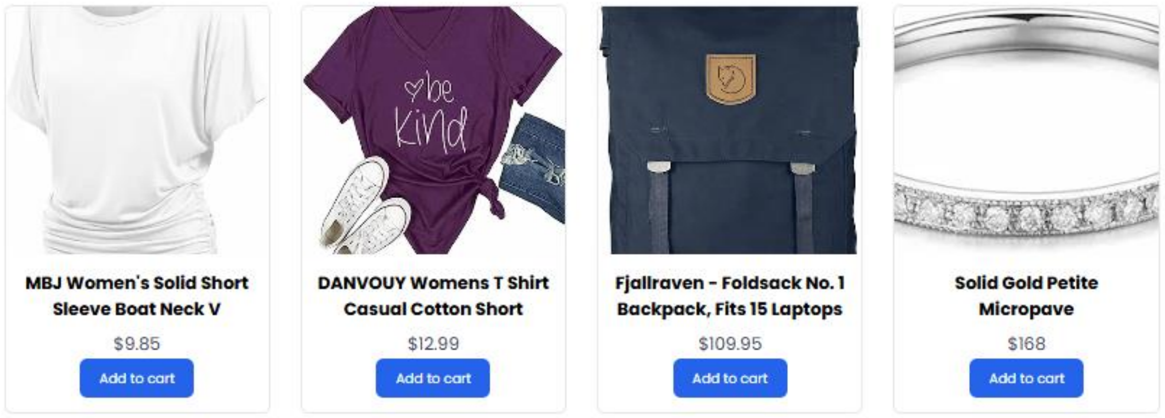
1. Product Listing Component

Purpose:

The product listing component displays all available products in a grid or list format, allowing users to browse items easily.

Steps:

1. **Create a Data Model**:
    - Define the structure for product data (e.g., id, name, price, image, description).
2. **Fetch Products Data**:
    - Use Sanity to query product data efficiently.
    - Example:
    - const fetchProducts = async () => {
    -   const query = `*[_type == "product"]`;
    -   const products = await client.fetch(query);
    -   setProducts(products);
    - };
3. **Build the UI**:
    - Create a responsive grid using CSS or a framework like Tailwind CSS.
    - Display product details such as name, price, and image.
    - Add "View Details" and "Add to Cart" buttons.
4. **Implement Pagination (Optional)**:
    - Allow users to navigate through multiple pages of products.
5. **Optimize Performance**:
    - Use lazy loading for product images.
    - Cache data where possible to reduce API calls.


MBJ Women's Solid Short Sleeve Boat Neck V
$9.85
Add to cart
DANVOUY Womens T Shirt Casual Cotton Short
$12.99
Add to cart
Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops
$109.95
Add to cart
Solid Gold Petite Micropave
$168
Add to cart

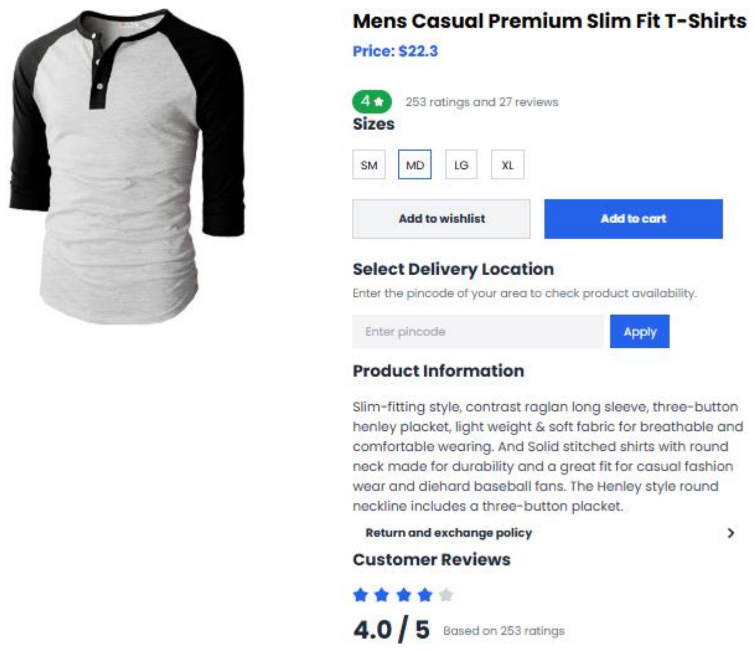
2. Product Detail Component

Purpose:

The product detail component displays detailed information about a selected product, including additional images, descriptions, and options.

Steps:

1. **Dynamic Routing**:
   - Set up a route for each product using a framework like Next.js or React Router.
   - Example:
   - <Route path="/product/:id" element={<ProductDetail />} />
2. **Fetch Product Details**:
   - Retrieve detailed product information using a Sanity query based on the id from the URL.
3. **Build the UI**:
   - Display product name, price, description, and additional images.
   - Include interactive elements like color/size selection if applicable.
4. **Add Actions**:
   - Add a quantity selector and an "Add to Cart" button.
5. **Enhance User Experience**:
   - Include a carousel for images.
   - Add animations or transitions for interactive elements.


Mens Casual Premium Slim Fit T-Shirts

Price: $22.3

4 ★ 253 ratings and 27 reviews

Sizes

Select Delivery Location

Enter the pincode of your area to check product availability.

Product Information

Slim-fitting style, contrast raglan long sleeve, three-button henley placket, light weight & soft fabric for breathable and comfortable wearing. And Solid stitched shirts with round neck made for durability and a great fit for casual fashion wear and diehard baseball fans. The Henley style round neckline includes a three-button placket.

Customer Reviews

4.0 / 5 Based on 253 ratings

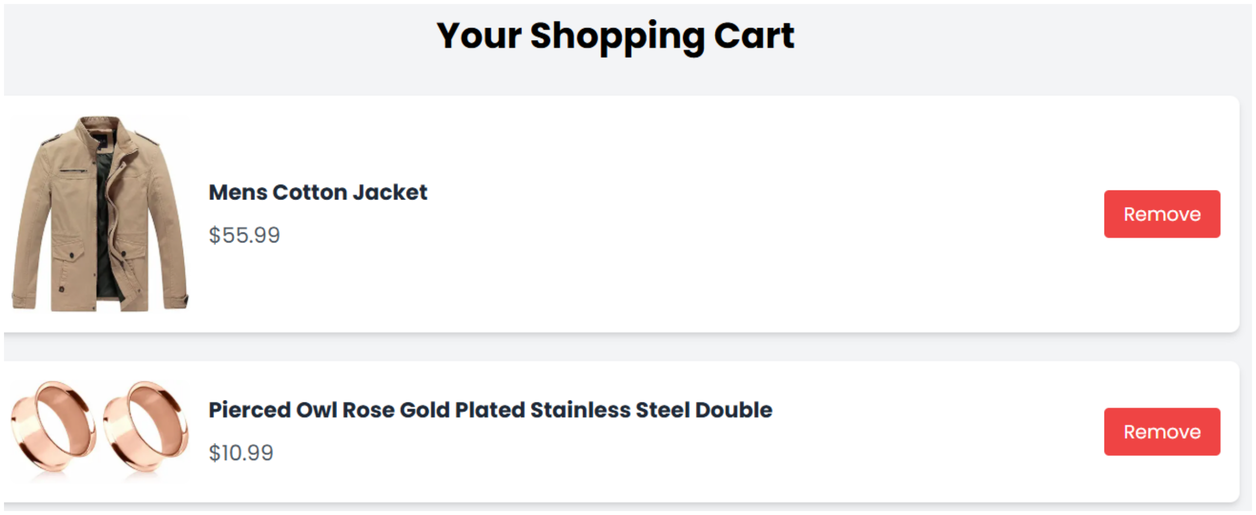
3. Shopping Cart Component

Purpose:

The shopping cart component allows users to view selected products, modify quantities, and proceed to checkout.

Steps:

1. **Set Up State Management**:
   - Use Redux, Context API, or a similar library to manage cart state.
2. **Build the UI**:
   - List all selected products with their name, image, price, and quantity.
   - Show the total price at the bottom.
3. **Implement Actions**:
   - Add buttons to increment or decrement product quantities.
   - Include a "Remove" button to delete items from the cart.
4. **Sync with Backend**:
   - Save the cart state to the server or local storage for persistence.
5. **Optimize Responsiveness**:
   - Ensure the cart is accessible on both desktop and mobile devices.


Your Shopping Cart

Mens Cotton Jacket

$55.99

Pierced Owl Rose Gold Plated Stainless Steel Double

$10.99

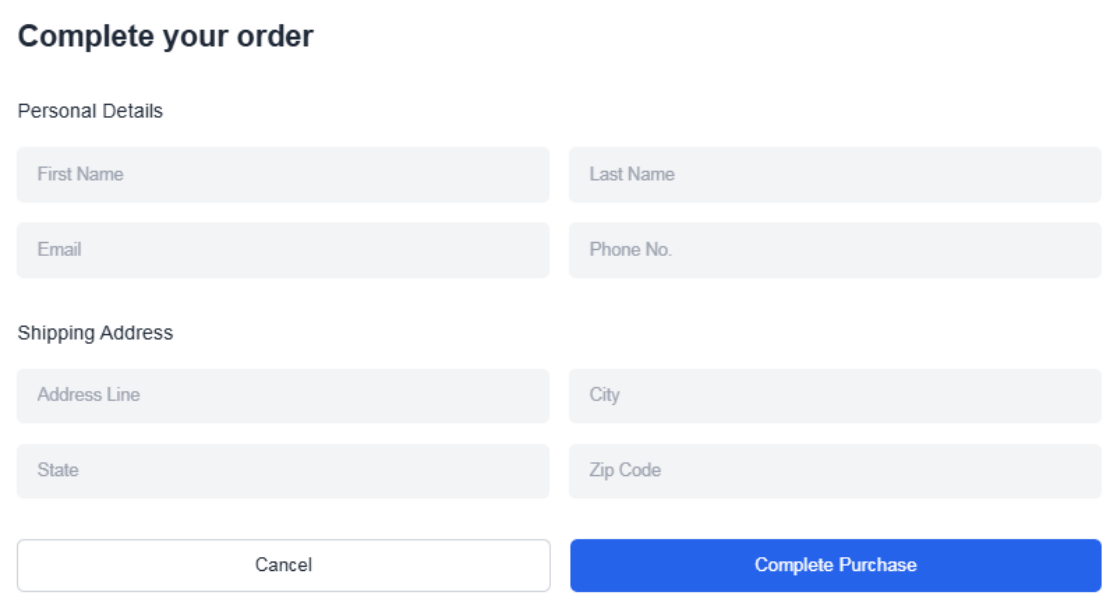
4. Checkout Component

Purpose:

The checkout component allows users to finalize their purchase by entering shipping and payment details.

Steps:

1. **Collect User Information**:
   - Include fields for name, address, email, and phone number.
2. **Add Payment Integration**:
   - Integrate payment gateways like Stripe or PayPal.
3. **Validate Input**:
   - Ensure all required fields are filled and valid before submission.
4. **Handle Errors**:
   - Display error messages for invalid inputs or payment issues.
5. **Provide Feedback**:
   - Show a confirmation message or redirect users to a success page after completing the checkout process.


Complete your order

Personal Details

First Name	Last Name
Email	Phone No.

Shipping Address

Address Line	City
State	Zip Code

5. Search Bar Component

Purpose:

The search bar component allows users to quickly find products by entering keywords or phrases.

Steps:

1. **Build the UI**:
   - Create a search input field with a "Search" button.
   - Use Tailwind CSS for styling and responsiveness.
2. **Handle Input**:
   - Add a state variable to store the search query.
   - Example:
   - const [query, setQuery] = useState('');
3. **Filter Products**:
   - Filter the product list based on the search query.
   - Example:
   - const filteredProducts = products.filter(product =>
   - product.name.toLowerCase().includes(query.toLowerCase()));
4. **Optimize Search**:
   - Debounce the search input to reduce unnecessary filtering.
5. **Enhance User Experience**:

- Show suggestions or autocomplete options as the user types.

---

6. User Profile Component

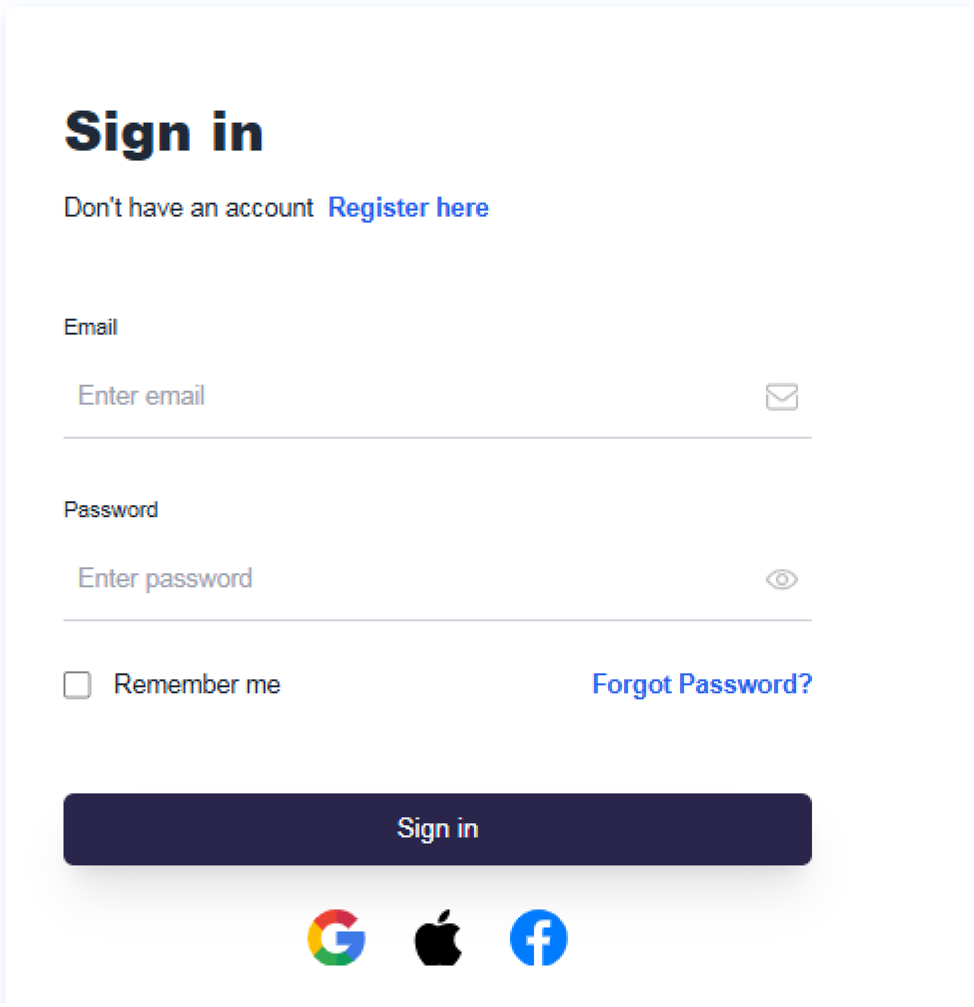
Purpose:

The user profile component allows users to view and edit their personal information and order history.

Steps:

1. **Build the UI**:
   - Display user details such as name, email, and profile picture.
   - Include sections for editing information and viewing past orders.
2. **Fetch User Data**:
   - Retrieve user data from an API or local storage.
   - Example:

     ```
     const fetchUserData = async () => {
       const response = await fetch('/api/user');
       const data = await response.json();
       setUser(data);
     };
     ```
3. **Implement Edit Functionality**:
   - Allow users to update their information through a form.
   - Validate inputs and handle API calls for updates.
4. **Display Order History**:
   - List past orders with details like order date, items purchased, and total amount.
5. **Enhance Security**:
   - Sensitive information is securely handled and displayed.


Sign in

Don't have an account Register here

Email

Enter email

Password

Enter password

Remember me Forgot Password?

Sign in

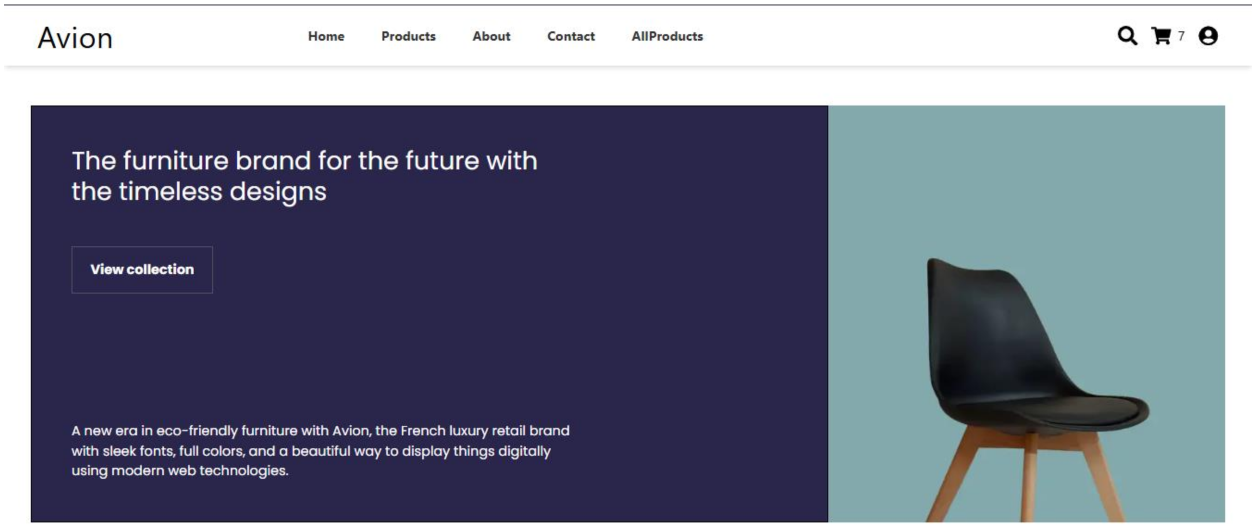
7. Responsive Design and Accessibility

Purpose:

The frontend is usable on all devices and accessible to all users.

Steps:

1. **Use Responsive Frameworks**:
   - Use Tailwind CSS or CSSfor responsive layouts.
2. **Test Across Devices**:
   - Use browser developer tools to test on various screen sizes.
3. **Add ARIA Attributes**:
   - Enhance accessibility by adding ARIA labels and roles to interactive elements.
4. **Keyboard Navigation**:
   - Ensure all functionalities are accessible via keyboard.



The furniture brand for the future with the timeless designs

A new era in eco-friendly furniture with Avion, the French luxury retail brand with sleek fonts, full colors, and a beautiful way to display things digitally using modern web technologies.

8. Performance Optimization

Purpose:

Enhance the speed and efficiency of the frontend.

Steps:

1. **Minimize API Calls**:
   - Use caching and batch requests where possible.
2. **Optimize Images**:
   - Use modern formats like WebP and enable lazy loading.
3. **Bundle and Minify Code**:
   - Use tools like Webpack or Vite to bundle and minify your code.
4. **Enable CDN**:
   - Serve assets via a Content Delivery Network to reduce load times.

Conclusion

By following these steps, you can build a dynamic, responsive, and user-friendly frontend for your web page. Focus on modular components, efficient state management, and a seamless user experience to ensure success.

Prepared by Kashaf Tariq