**Title: Building an E-commerce Website using React and Next.js**

Introduction: The purpose of this assignment is to build an e-commerce website using the React library and Next.js framework. React is a JavaScript library for building user interfaces, while Next.js is a framework that extends React and provides additional features for server-side rendering and routing. This assignment will cover the fundamental concepts and best practices for developing a functional e-commerce website using React and Next.js.

Assignment Tasks:

1. Setting up the Development Environment:
   * Install Node.js and npm (Node Package Manager) if not already installed.
   * Use the create-next-app command to create a new Next.js project.
   * Set up the basic project structure, including the necessary folders (pages, components, styles, etc.).
   * Install any additional dependencies required for the project.
2. Designing the User Interface:
   * Create a homepage component (e.g., Home.js) that displays a list of products.
   * Design a product listing page component (e.g., Products.js) that shows a grid of products with their details (title, image, price, etc.).
   * Implement a product detail page component (e.g., ProductDetail.js) that displays the selected product's complete information.
   * Create a navigation component (e.g., Navbar.js) to navigate between different pages.
   * Style the components using CSS or a styling library (e.g., CSS modules, styled-components, or Tailwind CSS).
3. Managing State with React Context or Redux:
   * Set up a global state management system using React Context API or Redux.
   * Define the necessary actions (e.g., addToCart, removeFromCart, updateQuantity) to handle cart functionality.
   * Implement reducers to handle these actions and update the global state accordingly.
   * Create a context provider component to provide the state and actions to the rest of the application.
   * Connect the necessary components (e.g., ProductDetail, Navbar) to the global state and dispatch actions when needed.
4. Fetching Product Data:
   * Simulate a backend or use an API to fetch product data (e.g., from a JSON file or a database).
   * Create a utility function or API service to fetch the product data asynchronously.
   * Use useEffect or a lifecycle method to fetch the product data when the homepage or product listing page mounts.
   * Pass the fetched data to the respective components (e.g., Products, ProductDetail) as props.
   * Implement filtering and sorting options to allow users to refine the product listing based on categories, prices, etc.
5. Implementing the Shopping Cart:
   * Create a cart component (e.g., Cart.js) that displays the selected products, quantities, and total price.
   * Provide buttons or actions to add/remove items from the cart and update quantities.
   * Use the global state management system to handle cart functionality (e.g., adding/removing items, updating quantities).
   * Display the cart component in the Navbar or on a dedicated cart page.
6. Implementing Checkout and Payment:
   * Design a checkout page component (e.g., Checkout.js) where users can enter their shipping and payment information.
   * Integrate a payment gateway (e.g., Stripe, PayPal) or simulate the payment process for testing purposes.
   * Handle successful and failed payment transactions and provide appropriate feedback to the user.
   * Update the cart state and clear the cart after a successful purchase.
7. Implementing User Authentication (Bonus Task):
   * Add user authentication functionality using a library like Firebase or implement your own authentication system.
   * Create user registration and login page components.
   * Secure the checkout process by requiring authentication before completing the purchase.
   * Store the user's authentication status and relevant information in the global state or using browser cookies/local storage.

Api service sample code:

import axios from 'axios';

const API\_URL = 'https://example.com/api'; // Replace with your actual API URL

// Service method to fetch all products

export const getAllProducts = async () => {

try {

const response = await axios.get(`${API\_URL}/products`);

return response.data;

} catch (error) {

throw new Error('Failed to fetch products.');

}

};

// Service method to fetch a specific product by ID

export const getProductById = async (productId) => {

try {

const response = await axios.get(`${API\_URL}/products/${productId}`);

return response.data;

} catch (error) {

throw new Error('Failed to fetch the product.');

}

};

// Service method to create a new product

export const createProduct = async (productData) => {

try {

const response = await axios.post(`${API\_URL}/products`, productData);

return response.data;

} catch (error) {

throw new Error('Failed to create the product.');

}

};

// Service method to update an existing product

export const updateProduct = async (productId, productData) => {

try {

const response = await axios.put(`${API\_URL}/products/${productId}`, productData);

return response.data;

} catch (error) {

throw new Error('Failed to update the product.');

}

};

// Service method to delete a product

export const deleteProduct = async (productId) => {

try {

const response = await axios.delete(`${API\_URL}/products/${productId}`);

return response.data;

} catch (error) {

throw new Error('Failed to delete the product.');

}

};

Bottom of Form