React Project: ShoppyGlobe E-commerce Application (250 marks)

Objective: Create a basic e-commerce application named ShoppyGlobe.

Requirements:

Set up a new React project using vite. (If your project is not created using Vite, it will not be considered your own work, and **you may receive a score of 0**.)

Component Structure: (20 marks)

- App: The main component.
- Header: Display the navigation menu and shopping cart icon
- ProductList: Displays a list of products.
- ProductItem: Represents a single product including an “Add to Cart” button.
- Product Detail: Show detailed information about a selected product.
- Cart: Displays the items added to the cart with options to modify quantities or remove items.
- CartItem: Represents a single item in the cart.
- NotFound: Display a 404 page for unknown routes.
  In the 404 page, display proper error details on the UI

Checkout: Create a dummy form to collect user details, along with a summary of the products added to the cart. Include a "Place Order" button. Upon clicking the button, display a message saying "Order placed" and empty the cart, then redirect the user back to the Home page automatically.,

Props: (10 marks)

- Utilize props to pass data from parent components to child components.
- Ensure components are functional and reusable with appropriate prop types.

Data Fetching with useEffect: (40 marks)

- ProductList Component: Use useEffect to fetch the list of products from an API endpoint ('https://dummyjson.com/products') when the component mounts. Store the fetched data in the component’s state. Create a custom hook for fetching the product list. (20 marks)

- ProductDetail Component: Use useEffect to fetch details of a selected product based on route parameters when the component mounts. Store the fetched data in the component’s state. (10 marks)

- Error Handling: Implement error handling to manage failed data fetch requests gracefully. (10 marks)

State Management: (70 marks)

- Redux: Implement Redux for more complex state management.
- Create actions, reducers, and selectors to manage the state of cart items. (50 marks)
- Implement a search feature to filter products (using redux state) in the ProductList. (20 marks)

Event Handling: (20 marks)

- Add a button in each ProductItem to add the product to the cart.
- Add a button in each CartItem to remove the product from the cart.
- Add functionality to allow users to adjust the quantity of products in the cart. However, the quantity of each product should not go below 1.
- Ensure that the add product and remove product functionality is implemented correctly using Redux.

React Routing: (20 marks)

(by CreateBrowserRoute (better features, data handling) from the traditional router)

- Implement routing using React Router.
- Create routes for Home, Product Detail, Cart, and Checkout pages.
- Use route parameters for product details. (The router should be dynamic)

React Lists: (10 marks)

- Render the list of products in the ProductList component.
- Render the list of cart items in the Cart component.
- Ensure that you provide a unique key to each list.

Performance Optimization: (20 marks)

- Implement code splitting and lazy loading for components using React.lazy and Suspense for all the components. Implement lazy loading for images.

Styling: (20 marks)

- Apply CSS for styling.
- Ensure the application is responsive and works well on different screen sizes.

Submission Guidelines: (20 marks)

- Ensure your application runs without errors. (7 marks)
- Use proper indentation and comments to explain your code. (5 marks)
- Make sure to remove the `node_modules`. (3 Marks)

- Submit a link to your GitHub repository with the completed project. There should be at least 25 commits in the GitHub repository (Commits should be relevant) (5 marks)

Note: Do not share the Git-Repo link only. Put the repo link inside the Readme file and send the entire code folder.