Coding Activity: E-Commerce Cart System (React & NodeJS)

Duration 120 min

Problem Statement & Objective

In today's digital world, online shopping is a major industry. A well-structured e-commerce system ensures a smooth shopping experience, from browsing products to checking out securely. Your task is to develop a **Mini E-Commerce Cart System** using **React & NodeJS**.

Scenario:

A startup, **ShopEase**, wants to build a simple shopping cart system that allows users to:

- View available products fetched from a backend API
- Add products to their cart and manage quantities
- Remove items and view a dynamic total price
- Proceed to checkout with a validated form
- Receive an order confirmation

As a developer, your goal is to implement this system by focusing on **React component structuring, state management, event handling, form validation, routing, and API integration**. You will also need to ensure that **no plagiarism** occurs in submissions by following the submission guidelines strictly.

User Stories Table

User Story ID	User Story	Acceptance Criteria
US-1	*As a user, I should see a list of products fetched from an API.*	Products must be displayed dynamically using `fetch` or `Axios` from a JSON server.

US-2	*As a user, I should be able to add products to my shopping cart.*	Clicking "Add to Cart" should update the state and reflect changes in the UI instantly.
US-3	*As a user, I should see a summary of my cart with item names, quantities, and prices.*	A cart summary should display all selected products with accurate total price calculations.
US-4	*As a user, I should be able to update the quantity of an item in my cart.*	Quantity updates should dynamically modify the cart total.
US-5	*As a user, I should be able to remove an item from my cart.*	Removing an item should reflect immediately in the UI and update the total price.
US-6	*As a user, I should be able to fill a checkout form with my name, email, and address.*	The checkout form should validate inputs using `Formik` and `Yup` before submission.
US-7	*As a user, after submitting my order, I should see an order confirmation page.*	Successful form submission should navigate the user to an order confirmation page.
US-8	*As a user, I should be able to navigate between Home, Cart, and Checkout pages.*	Navigation should be handled using `React Router`, and checkout should be protected from direct URL access.

Evaluation Rubric (Correctness & Scoring)

User Story	Points (0-3)	Points (4-7)	Points (8-10)
Product Listing Page	Hardcoded products, no API call	API call works, but UI is incomplete	API fetch works, products displayed with styling
Add to Cart	Button missing or non-functional	Item added but UI not updating properly	Item added with state management, UI updates instantly
View Cart Summary	Cart page missing	Cart page exists but incorrect calculations	Fully functional summary with accurate price calculations

Update Item Quantity	No option to update	Quantity updates but not reactive	Correctly updates quantity and recalculates total dynamically
Remove Item from Cart	No remove button	Remove works but UI doesn't update properly	Remove works instantly with UI updating
Checkout Form	Form missing or unvalidated	Form exists but lacks proper validation	Form correctly validates input using Formik/Yup
Order Confirmation	No confirmation message	Basic message, lacks styling/navigation	Full confirmation page with styled UI and navigation
Navigation & Routing	No navigation or hardcoded links	Navigation works, but state is lost	Navigation works seamlessly with protected checkout route

---

Submission Guidelines & Anti-Plagiarism Measures

To ensure originality and avoid **code copying**, the following **submission rules** must be followed:

1. **Project Structure & File Naming:**
   - Components must be properly structured in separate files.
   - Each participant should include their **name in filenames** (e.g., `Cart_Amit.js`, `Checkout_Sneha.js`).
2. **Code Documentation & Screenshots:**
   - Add **comments** in code explaining key logic.
   - Submit **screenshots** of output pages with the participant's name in the UI.
3. **Plagiarism Check:**
   - **Code similarity detection** tools will be used to verify originality.
   - Participants submitting identical or heavily copied code will be **disqualified**.
4. **GitHub Submission Format:**
   - Upload the project to **GitHub** and submit the repository link.
   - README file should include **setup instructions & explanations**.
5. **Instructor Verification:**
   - Each submission will be reviewed manually.
   - Participants might be asked to **explain their code logic** in a follow-up discussion.

🛒 E-Commerce Cart System - Evaluation Rubric (60 Marks)

Project: Mini E-Commerce Cart System using React & NodeJS
Duration: 120 minutes
Total Marks: 60

User Story & Functionality	Criteria	Marks Breakdown	Total Marks
Product Listing Page (US-1)	Fetches product data from API correctly	2	**6**
	Products dynamically displayed on UI	2	
	Proper styling applied	2	
Add to Cart (US-2)	Clicking "Add to Cart" updates state	2	**6**
	UI updates instantly after adding an item	2	
	State persists correctly when navigating	2	
View Cart Summary (US-3)	Cart displays selected items properly	2	**6**
	Correct quantity and price calculations	2	
	UI updates dynamically as cart changes	2	
Update Item Quantity (US-4)	Users can change item quantity	2	**6**
	Quantity updates reflect in UI instantly	2	
	Total price updates correctly	2	
Remove Item from Cart (US-5)	"Remove" button is available	2	**6**
	Item is removed from the cart on click	2	

	UI updates dynamically after removal	2	
Checkout Form (US-6)	Form contains name, email, address fields	2	**6**
	Inputs validate using Formik/Yup	2	
	Prevents empty or incorrect submissions	2	
Order Confirmation (US-7)	Order submission is successful	2	**6**
	Confirmation page displays order details	2	
	UI is styled and user-friendly	2	
Navigation & Routing (US-8)	Home, Cart, and Checkout pages navigate properly	2	**6**
	Checkout route is protected from direct access	2	
	Navigation state persists properly	2	
Code Quality & Best Practices	Follows React best practices (hooks, clean structure)	2	**6**
	Well-structured components and modular code	2	
	Proper formatting, variable naming, and commenting	2	
Submission Guidelines & Anti-Plagiarism	File names follow required format	2	**6**
	GitHub submission includes README & documentation	2	
	Code originality verified, no plagiarism detected	2	
Total maks			60

Grading Scale for Each Sub-Level

Marks	Performance Level	Description
0	**Not Attempted**	Feature is missing or completely incorrect
1	**Partially Implemented**	Some functionality present but has errors
2	**Fully Functional**	Correct and fully working implementation