Day 14 – ReactJS Advanced Concepts

Challenge 1: Lazy Loading & Code Splitting

User Story:
As a user of an online learning platform, I want modules to load only when I click on them so that the app loads faster.

Problem Statement:
Implement **lazy loading and code splitting** for “Course Details” and “Instructor Profile” components using React.lazy() and Suspense.

Expected Outcome:

- Only visible modules load initially.
- Clicking “View Details” dynamically imports the component.

Bonus:
Show a **loading spinner** during lazy loading using Bootstrap or a custom loader.

Challenge 2: Pure Components

User Story:
As a dashboard viewer, I want widgets to render only when their data changes so that performance is optimized.

Problem Statement:
Create a **Pure Component** StatsCard that receives props (title, value, lastUpdated) and re-renders only when props actually change.

Expected Outcome:

- Use React.PureComponent or React.memo.
- Log re-renders in the console to show optimization.

Bonus:
Add a “Simulate Update” button that changes only one card’s value and observe render behavior.

Challenge 3: Error Boundary

User Story:
As a product manager, I want the app to display a friendly error message instead of crashing when a component fails.

Problem Statement:
Create an **Error Boundary** component that catches rendering errors in child components (e.g., a broken ProductCard) and displays a fallback UI.

Expected Outcome:

- Implement componentDidCatch() and getDerivedStateFromError().
- Error message should appear without breaking the rest of the UI.

Bonus:
Log the error details to the console or send them to a mock monitoring API.

Challenge 4: Portals

User Story:
As a user, I want to view notifications and modal popups that appear above all UI elements, even if nested inside components.

Problem Statement:
Create a **Modal** or **Notification** using **React Portals** that renders outside the main DOM hierarchy.

Expected Outcome:

- Use ReactDOM.createPortal().
- Modal opens/closes using state toggle in the parent component.

Bonus:
Add fade-in/out animation using Framer Motion or CSS transitions.
