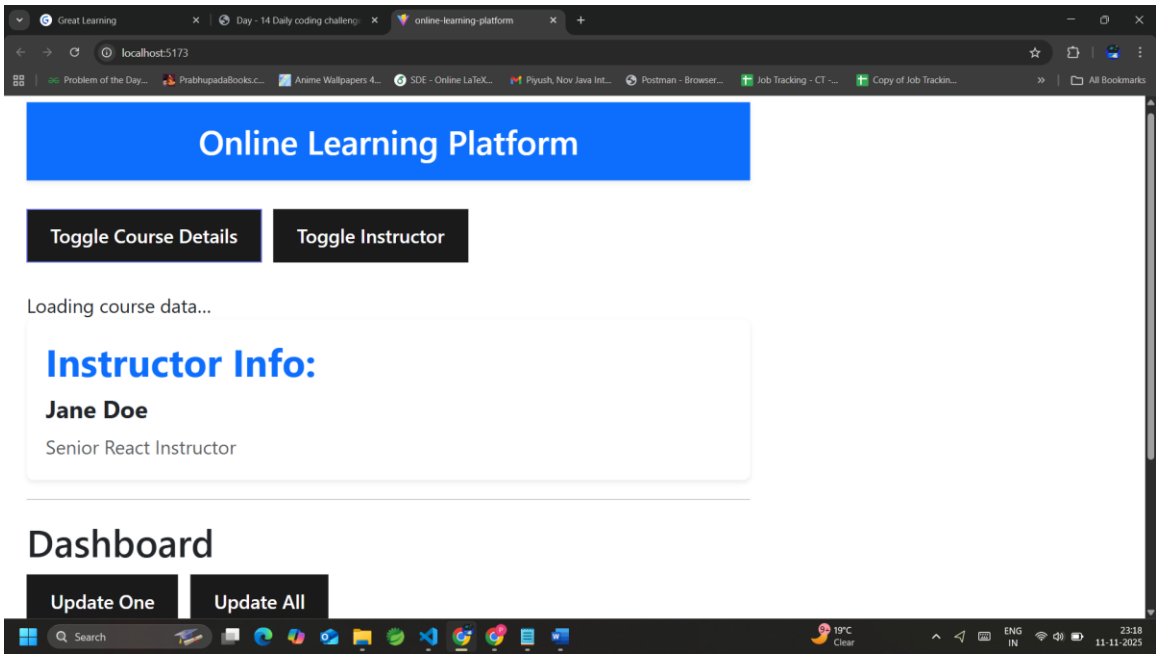
Day 14 Online Learning Platform- Documentation Report

This report documents the Day 14 Online Learning Platform. It explains how each user story was implemented using React, with code snippets, explanations, and screenshot of outputs.

1. Lazy Loading & Code Splitting (CourseDetails & InstructorProfile)

React.lazy() and Suspense are used to defer loading of CourseDetails and InstructorProfile components. This approach reduces initial load time and improves performance. While components load, a Bootstrap spinner (Loader) is displayed.

[Screenshot Placeholder Here]



- Code Snippet:

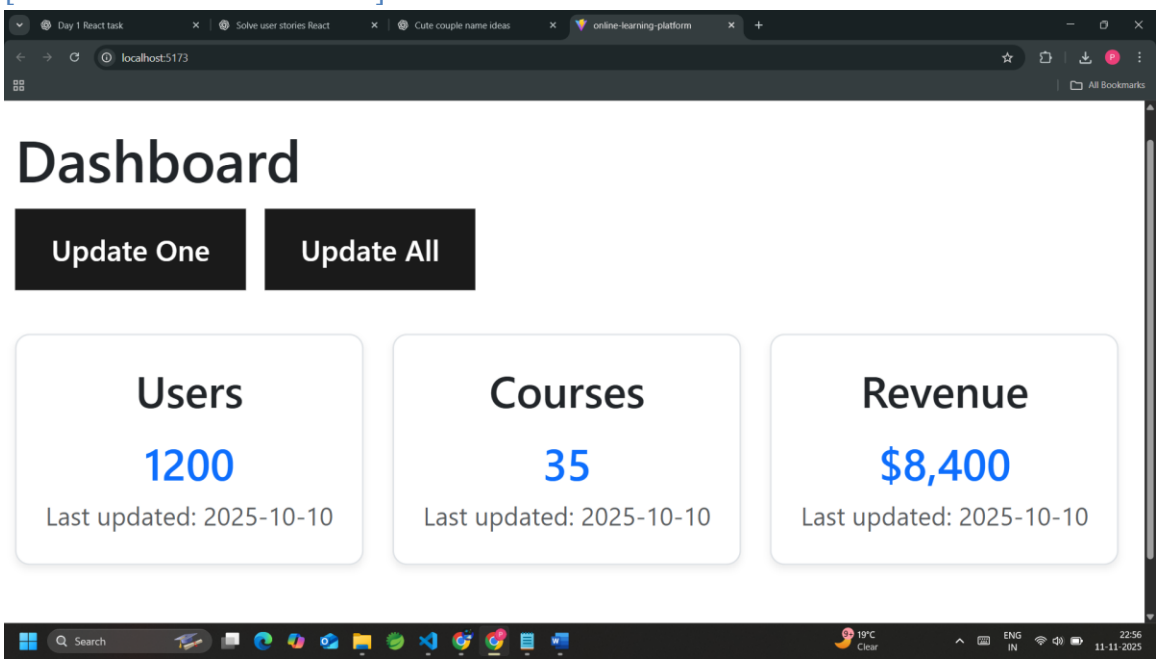
```
import React, { Suspense, useState } from "react";
import Loader from "./components/Loader";
import Dashboard from "./pages/Dashboard";
import Shop from "./pages/Shop";
import ExampleWithModal from "./pages/ExampleWithModal";

// Lazy load demo components
const CourseDetails = React.lazy(() =>
import("./components/CourseDetails"));
const InstructorProfile = React.lazy(() =>
import("./components/InstructorProfile"));

// Root App component - hosts all demos
```

```
export default function App() {
  const [showCourse, setShowCourse] = useState(false);
  const [showInstructor, setShowInstructor] = useState(false);

  return (
    <div style={{ padding: 20 }}>
      <header className="bg-primary text-white py-3 mb-4 text-center
shadow-sm">
        <h1 className="h3 m-0 fw-semibold">Online Learning Platform</h1>
      </header>

      <button onClick={() => setShowCourse(prev => !prev)}>Toggle Course
Details</button>
      <button onClick={() => setShowInstructor(prev => !prev)} style={{
marginLeft: 10 }}>Toggle Instructor</button>

      <div className="mt-4">
        <Suspense fallback={<Loader />}>
          {showCourse && <CourseDetails courseId={1} />}
          {showInstructor && <InstructorProfile instructorId={2} />}
        </Suspense>
      </div>

      <hr />
      <Dashboard />
      <hr />
      <Shop />
      <hr />
      <ExampleWithModal />
    </div>
  );
}
```

2. Pure Components (StatsCard)

StatsCard is wrapped in React.memo() to prevent unnecessary re-renders. It displays card-based statistics styled with Bootstrap.

[Screenshot Placeholder Here]



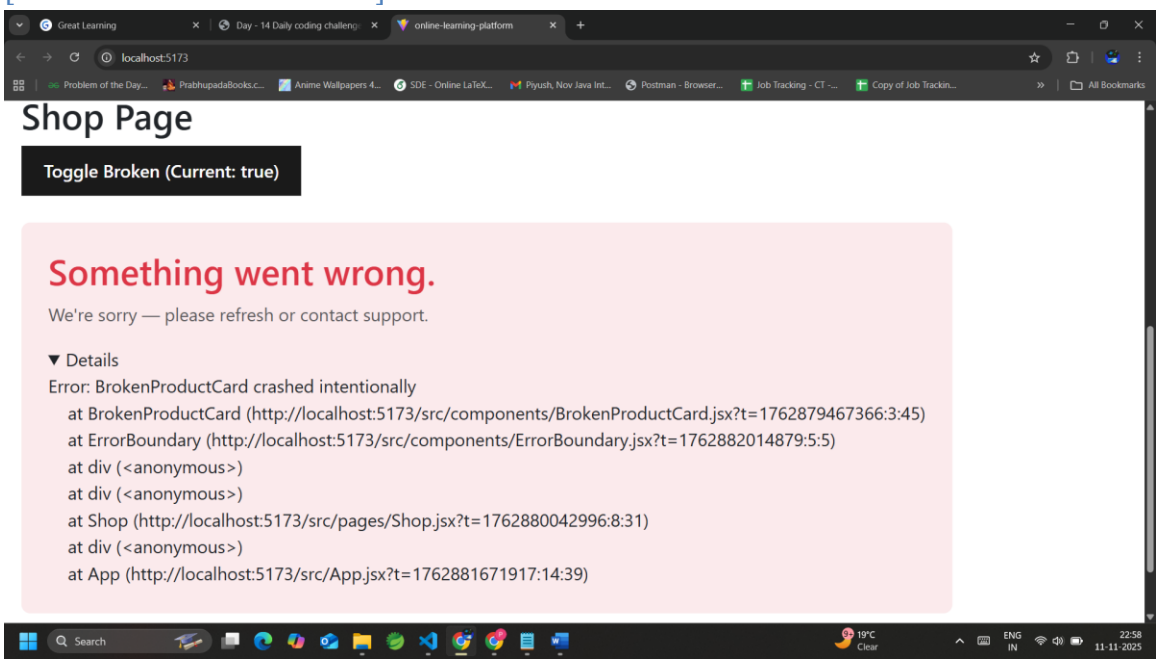
- Code Snippet:

```
import React from "react";

// Pure component: re-renders only when props actually change
function StatsCardInner({ title, value, lastUpdated }) {
  console.log(`Render: ${title}`);
  return (
    <div className="card border rounded-3 p-3 shadow-sm text-center mb-3">
      <h4 className="mb-2">{title}</h4>
      <div className="fs-4 fw-semibold text-primary">{value}</div>
      <small className="text-muted">Last updated: {lastUpdated}</small>
    </div>

  );
}

const StatsCard = React.memo(StatsCardInner);
export default StatsCard;
```

3. Error Boundary (ErrorBoundary & BrokenProductCard)

ErrorBoundary is implemented as a class component to catch rendering errors in child components. It prevents app crashes and shows a friendly Bootstrap alert fallback.

[Screenshot Placeholder Here]



- Code Snippet:

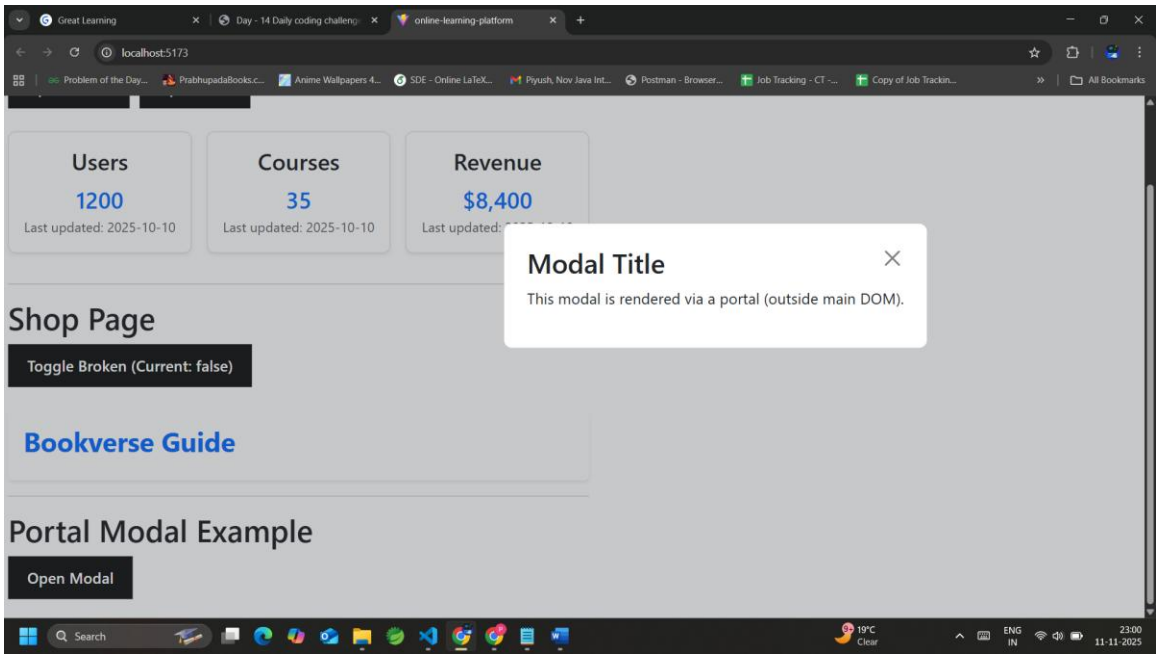
```
import React from "react";

// ErrorBoundary class component to catch runtime render errors in
children
export default class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false, error: null, info: null };
  }

  static getDerivedStateFromError(error) {
    return { hasError: true, error };
  }

  componentDidCatch(error, info) {
    console.error("ErrorBoundary caught:", error, info);
    this.setState({ info });
  }

  render() {
    if (this.state.hasError) {
      return (
        <div className="bg-danger bg-opacity-10 p-4 rounded-3">
```

```
          <h2 className="text-danger fw-semibold mb-2">Something went
wrong.</h2>
          <p className="text-muted mb-3">
            We're sorry — please refresh or contact support.
          </p>
          <details className="text-dark" style={{ whiteSpace: "pre-wrap"
}}>
            {this.state.error?.toString()}
            {this.state.info?.componentStack}
          </details>
        </div>

      );
    }
    return this.props.children;
  }
}
```

4. Portals (Modal Component)

The Modal component uses ReactDOM.createPortal() to render modals outside the root DOM hierarchy. Bootstrap's modal classes give it a sleek and responsive design.

[Screenshot Placeholder Here]



- Code Snippet:

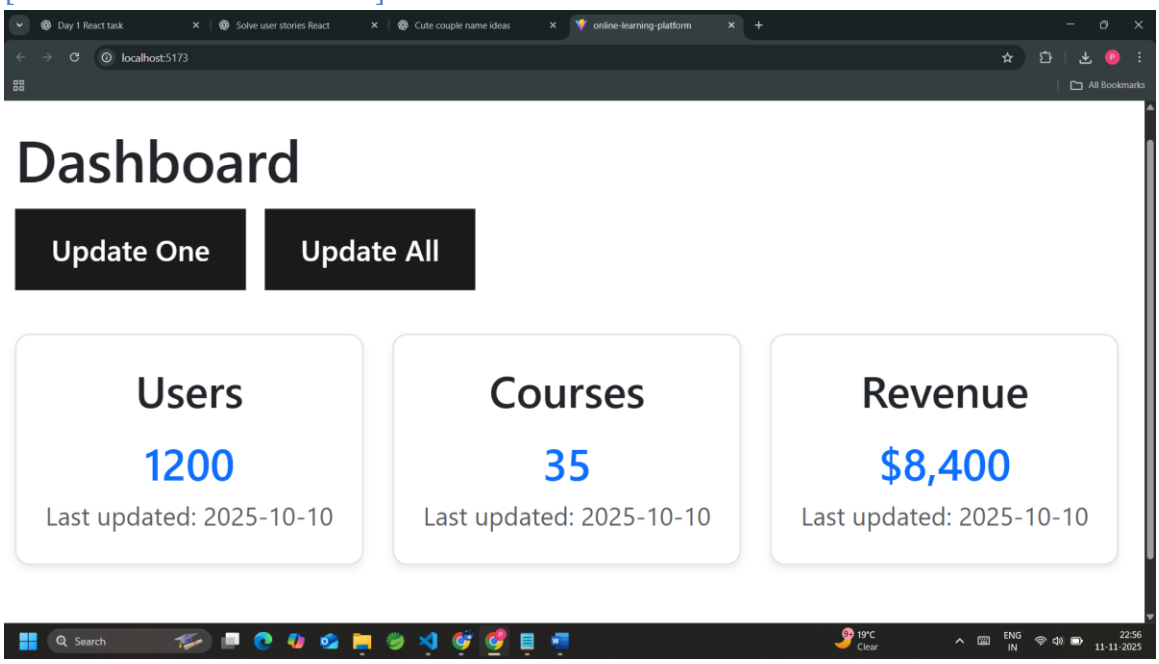
```
import React, { useEffect } from "react";
```

```jsx
import ReactDOM from "react-dom";

// Portal-based Modal component (renders outside main DOM hierarchy)
const modalRoot = document.getElementById("modal-root");

export default function Modal({ open, onClose, children }) {
  useEffect(() => {
    function onKey(e) {
      if (e.key === "Escape") onClose?.();
    }
    if (open) document.addEventListener("keydown", onKey);
    return () => document.removeEventListener("keydown", onKey);
  }, [open, onClose]);

  if (!open) return null;

  return ReactDOM.createPortal(

    <div
      className="position-fixed top-0 start-0 w-100 h-100 d-flex align-
items-center justify-content-center bg-dark bg-opacity-25"
      onMouseDown={onClose}
    >
      <div
        className="bg-white text-dark p-4 rounded-3 min-vw-25"
        onMouseDown={(e) => e.stopPropagation()}
      >
        <button
          onClick={onClose}
          className="btn-close float-end"
          aria-label="Close"
        ></button>
        {children}
      </div>
    </div>
    ,
    modalRoot
  );
}
```

5. Dashboard Page

Displays StatsCard components in a responsive Bootstrap grid. Includes buttons to simulate updates for one or all cards.

[Screenshot Placeholder Here]



- Code Snippet:

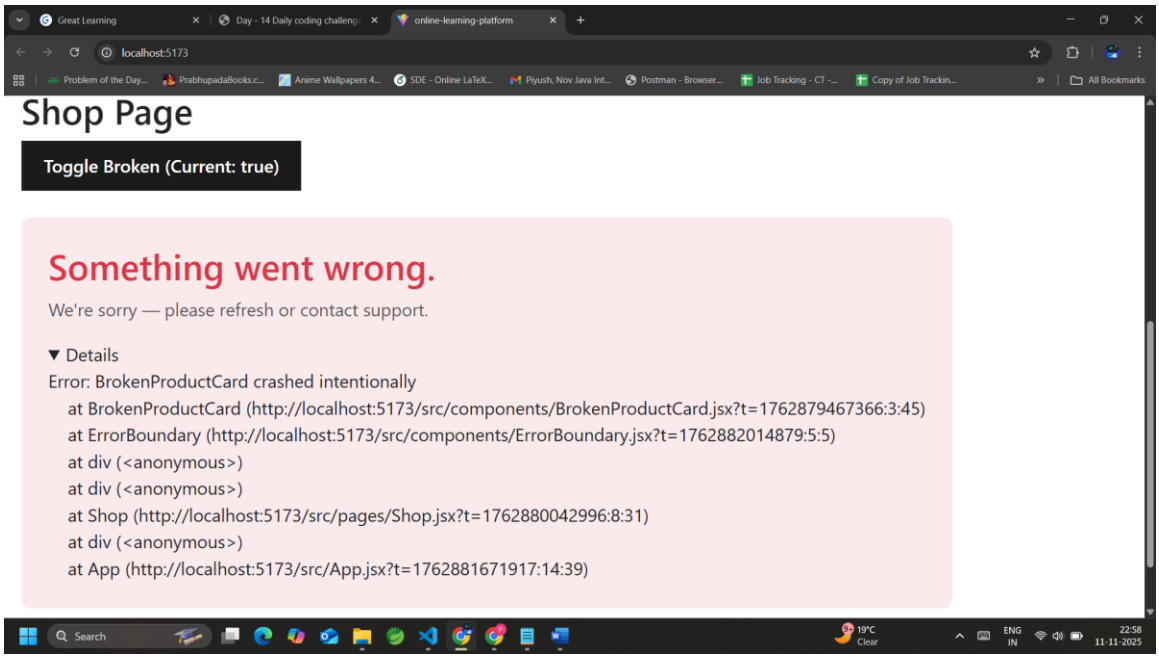
```
import React, { useState } from "react";
import StatsCard from "../components/StatsCard";

// Dashboard page demonstrates Pure Components behavior
export default function Dashboard() {
  const [cards, setCards] = useState([
    { id: 1, title: "Users", value: 1200, lastUpdated: "2025-10-10" },
    { id: 2, title: "Courses", value: 35, lastUpdated: "2025-10-10" },
    { id: 3, title: "Revenue", value: "$8,400", lastUpdated: "2025-10-10"
},
  ]);

  const simulateUpdate = (single = true) => {
    setCards(prev =>
      prev.map(c => (single && c.id !== 1 ? c : { ...c, value: c.value + "
*", lastUpdated: new Date().toLocaleString() }))
    );
  };

  return (
    <div>
      <h2>Dashboard</h2>
      <button onClick={() => simulateUpdate(true)}>Update One</button>
```

```
      <button onClick={() => simulateUpdate(false)} style={{ marginLeft:
10 }}>Update All</button>
      <div className="d-flex flex-wrap gap-3 mt-4">
        {cards.map(c => (
          <StatsCard key={c.id} {...c} />
        ))}
      </div>

    </div>
  );
}
```

6. Shop Page

Demonstrates the ErrorBoundary in action. Users can toggle an error in BrokenProductCard to view the fallback UI.

[Screenshot Placeholder Here]



- Code Snippet:

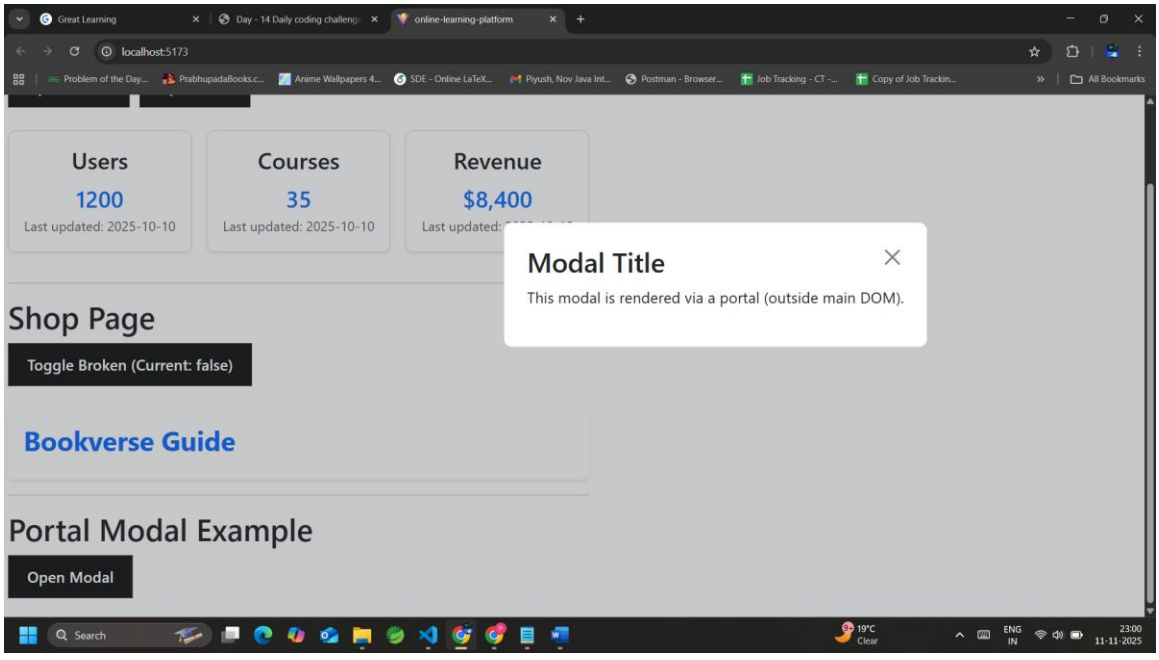
```
import React, { useState } from "react";
import ErrorBoundary from "../components/ErrorBoundary";
import BrokenProductCard from "../components/BrokenProductCard";

// Shop page demonstrating Error Boundary functionality
export default function Shop() {
  const [broken, setBroken] = useState(false);
```

```
  const product = { name: "Bookverse Guide", shouldThrow: broken };

  return (
    <div>
      <h2>Shop Page</h2>
      <button onClick={() => setBroken(b => !b)}>
        Toggle Broken (Current: {String(broken)})
      </button>
      <div className="mt-4">
        <ErrorBoundary>
          <BrokenProductCard product={product} />
        </ErrorBoundary>
      </div>
    </div>
  );
}
```

7. ExampleWithModal Page

Illustrates how to open and close a Bootstrap-styled modal rendered via React Portal.

[Screenshot Placeholder Here]



- Code Snippet:

```
import React, { useState } from "react";
import Modal from "../components/Modal";
```

```jsx
// Example page to demonstrate Portal-based Modal
export default function ExampleWithModal() {
  const [open, setOpen] = useState(false);
  return (
    <div>
      <h2>Portal Modal Example</h2>
      <button onClick={() => setOpen(true)}>Open Modal</button>
      <Modal open={open} onClose={() => setOpen(false)}>
        <h3>Modal Title</h3>
        <p>This modal is rendered via a portal (outside main DOM).</p>
      </Modal>
    </div>
  );
}
```