DAY 23 - JSX, COMPONENTS, PROP, LIST & CONDITIONAL RENDERING

1. Online Grocery Store Product List

Domain: E-commerce

Requirements:

1. **Functional Components**
   - ProductCard component to display each product.
   - ProductList to map through all products.
2. **JSX**
   - Use JSX for structured layout with className for styling.
   - Show price in bold, discount in a span with inline style (red).
3. **List Rendering**
   - Products array (name, price, availability, image).
   - Use .map() with unique key for rendering cards.
4. **Conditional Rendering**
   - If availability is true → Show "In Stock"
   - Else → Show "Out of Stock" with gray styling.
   - If discount > 0 → Show "On Sale" badge.
5. **Props**
   - ProductCard receives name, price, discount, availability, image as props.
   - Pass data from parent (App.jsx).

Real-life use: Mimics Flipkart/Amazon product listing.

2. Student Attendance Dashboard

Domain: Education

Requirements:

1. **Functional Components**
   - StudentCard component for each student.

   - AttendanceList to display all students.

2. **JSX**
   - Display student name, roll number, and attendance percentage.
   - Inline conditional styles → green text if attendance > 75, red otherwise.
3. **List Rendering**
   - Students array with (id, name, rollNo, attendance%).
   - .map() to render StudentCard.
4. **Conditional Rendering**
   - If attendance < 50 → Show “Warning: Low Attendance”.
   - If attendance > 90 → Show ⭐ “Excellent Student”.
5. **Props**
   - StudentCard receives name, rollNo, attendance as props.
   - Use children prop to show custom messages below card.

Real-life use: Similar to digital school attendance portals.

---

3. Job Board Application

Domain: Recruitment

Requirements:

1. **Functional Components**
   - JobCard to display job details.
   - JobList to render all jobs.
2. **JSX**
   - Title in <h2>, company name in <p>, apply button inside <button>.
   - Conditional inline style → Remote jobs highlighted with a green border.
3. **List Rendering**
   - Jobs array (id, title, company, location, isRemote, salary).
   - .map() to render multiple JobCards.
4. **Conditional Rendering**
   - If isRemote === true → Show “Remote Job” badge.

- If salary > 100000 → Show “High Paying Job” tag.
- Else show “Standard Salary”.

5. **Props**
   - JobCard accepts title, company, location, salary, isRemote as props.
   - Props are destructured in component.

Real-life use: Works like LinkedIn/Indeed job listings.

---

4. Movie Recommendation System

Domain: Entertainment

Requirements:

1. **Functional Components**
   - MovieCard component for each movie.
   - MovieList parent that maps through movies.
2. **JSX**
   - Poster image inside <img>, title inside <h3>, rating with stars.
   - Inline style to highlight rating above 8.0.
3. **List Rendering**
   - Movies array (id, title, genre, rating, isTrending, poster).
   - Use .map() to generate movie cards.
4. **Conditional Rendering**
   - If isTrending === true → Show 🔥 Trending badge.
   - If rating >= 8.5 → Show “Top Rated”.
   - If no movies exist → Show “No Movies Found”.
5. **Props**
   - MovieCard receives title, genre, rating, isTrending, poster.
   - children prop to add custom user reviews under the card.

Real-life use: Similar to Netflix/HBO/Prime movie listings.

---

5. Employee Directory System

Domain: HR Management

Requirements:

1. **Functional Components**
   - EmployeeCard to show each employee.
   - EmployeeList to map through employees.
2. **JSX**
   - Use <div> grid layout for directory.
   - Employee name in bold, role in italic, department in a <span>.
3. **List Rendering**
   - Employees array (id, name, role, department, isActive).
   - .map() with key for each EmployeeCard.
4. **Conditional Rendering**
   - If isActive === true → Show green “Active” label.
   - Else show red “Inactive”.
   - If department === "HR" → Highlight background color.
5. **Props**
   - EmployeeCard accepts name, role, department, isActive.
   - Default prop for role = "Employee" if not passed.

Real-life use: Company staff directory for HR systems.

Why These Projects Are Perfect

- They cover **all JSX rules** (className, expressions, inline styles).
- Each project uses **functional components** for modular design.
- **Lists with .map()** are central to all projects.
- **Conditional rendering** is implemented in multiple real cases (badges, warnings, empty states).
- **Props** are used for passing data, destructuring, default values, and children props.
- Each project simulates a **real-world app scenario** (e-commerce, HR, entertainment, education, jobs).
