**Exercise 1: User Directory with Lists and Keys**

**Question:**

1. Create a React application to display a list of users.
2. The application should contain three components: App, UserList, and UserItem.
   * App should hold an array of user objects with properties id, name, and age.
   * Pass the list of users to UserList as a prop.
   * UserList should iterate over the list using .map() and render each user using the UserItem component.
3. Each UserItem should display the user's name and age.
4. Use id as the key for each user.
5. Style each component with its own separate CSS file to make the app visually appealing.

**Exercise 2: Extended User Directory with Filter**

**Question:**

1. Extend the previous application to include a search feature.
2. Add an input field in the UserList component to filter users by name.
3. The filtered list should update in real-time as the user types in the input field.
4. Use a controlled input for managing the search term state.

**Exercise 3: Toggle User Details**

**Question:**

1. Update the UserItem component to include a "Show Details" button.
2. When clicked, toggle the visibility of additional user information (e.g., email and profession).
3. The additional information should only be visible when the button is clicked.

**Additional User Data in App.js**

const users = [

{ id: 1, name: "Alice", age: 25, email: "alice@outlook.com", profession: "Engineer" },

{ id: 2, name: "Bob", age: 30, email: "bob@gmail.com", profession: "Designer" },

{ id: 3, name: "Charlie", age: 35, email: "charlie@outlook.com", profession: "Manager" },

];

**Exercise 4: Add User Functionality**

**Question:**

1. Add functionality in the App component to allow the user to add a new user to the list.
2. Include a form with inputs for name and age.
3. The new user should be appended to the list and displayed in the UserList.

**Exercise 5: Delete User Functionality**

**Question:**

1. Add a "Delete" button to each UserItem.
2. Clicking the button should remove the user from the list.
3. The updated list should reflect immediately.

**Exercise 6: Toggle User Details**

**Question**

Create a React application with the following requirements:

1. Display a list of users with their names.
2. Include a button for each user that toggles the visibility of their details (age and email).
3. Use at least three components:
   * App to manage the state.
   * **UserList** to display the list of users.
   * **UserItem** to handle individual user details.
4. Add CSS for each component in separate files.

**Exercise 7: Product Catalog with Categories**

**Question**

Create a React application with the following requirements:

Display a list of **product categories**.

When a category is clicked, show a list of **products** for that category.

Allow users to click on a **product** to view its details.

**Components:**

**CategoryList** to display product categories.

**ProductList** to display products within a category.

**ProductDetails** to display details of the selected product.

Include a separate CSS file for each component.

**Exercise 8: Task Management Dashboard**

**Question**

Build a Task Management Dashboard in React with the following functionality:

1. Display multiple task lists (e.g., "To Do," "In Progress," "Completed").
2. Each task list should show tasks under it.
3. Allow the user to add new tasks to a list.
4. Implement functionality to move tasks between lists (drag-and-drop not required; use buttons).
5. Include a detailed view of a selected task, where the user can update its description or delete it.
6. Use at least 10 components:
   * App to manage the global state.
   * TaskList for individual task lists.
   * TaskItem for each task.
   * TaskDetails for viewing/updating a task.
   * AddTaskForm for adding new tasks.
   * Additional components to organize and display the UI as needed.
7. Each component should have its own CSS file.

**Additional Components:**

Include AddTaskForm.js and TaskDetails.js, along with their corresponding CSS files. Let me know if you'd like to see these components in detail!

**Exercise 9: Project Management Dashboard**

Build a React application that simulates a **Project Management Dashboard**. The application should include the following features:

1. A list of projects with details such as name, description, and assigned team members.
2. A form to add new projects.
3. Task management within each project, including adding, editing, and deleting tasks.
4. Display project details with a list of associated tasks and team members.
5. Assign team members to projects from a predefined list.
6. Use at least **10 components**, each with a dedicated CSS file.

**Directory Structure**

src/

├── components/

│ ├── AddProjectForm.js

│ ├── AddTaskForm.js

│ ├── AssignTeamMembers.js

│ ├── Navbar.js

│ ├── ProjectCard.js

│ ├── ProjectDetails.js

│ ├── ProjectList.js

│ ├── TaskDetails.js

│ ├── TeamMemberCard.js

│ ├── TeamMemberList.js

├── css/

│ ├── AddProjectForm.css

│ ├── AddTaskForm.css

│ ├── AssignTeamMembers.css

│ ├── Navbar.css

│ ├── ProjectCard.css

│ ├── ProjectDetails.css

│ ├── ProjectList.css

│ ├── TaskDetails.css

│ ├── TeamMemberCard.css

│ ├── TeamMemberList.css

├── App.js

├── App.css

**Exercise 10: Human Resource Management System (HRMS)**

Build a React application that simulates a **Human Resource Management System (HRMS)**. The application should include the following features:

**HRMS App Example**

This HRMS app will include the following components:

* **Navbar**: For navigation between different sections.
* **EmployeeList**: Displays a list of employees.
* **AddEmployeeForm**: A form to add new employees.
* **EmployeeDetails**: Displays the details of an employee.
* **DepartmentList**: Displays a list of departments.
* **AddDepartmentForm**: A form to add new departments.
* **AttendanceList**: Displays employee attendance.
* **AddAttendanceForm**: A form to record attendance.
* **EmployeeCard**: A card for each employee in the list.
* **DepartmentCard**: A card for each department in the list.
* **HRDashboard**: Main dashboard showing an overview of employees, departments, and attendance.

Here’s a more complete HRMS example with **10 components**, each following the **Lists and Keys** concept in React. Along with the components, I'll include their corresponding **CSS files** for styling. The structure is as follows:

**Project Structure:**

hrms-app/

├── public/

│ ├── index.html

│ └── favicon.ico

├── src/

│ ├── components/

│ │ ├── EmployeeList.js

│ │ ├── DepartmentList.js

│ │ ├── AttendanceList.js

│ │ ├── SalaryList.js

│ │ ├── TaskList.js

│ │ ├── LeaveList.js

│ │ ├── ProjectList.js

│ │ ├── TrainingList.js

│ │ ├── DepartmentHeads.js

│ │ ├── JobTitleList.js

│ ├── css/

│ │ ├── EmployeeList.css

│ │ ├── DepartmentList.css

│ │ ├── AttendanceList.css

│ │ ├── SalaryList.css

│ │ ├── TaskList.css

│ │ ├── LeaveList.css

│ │ ├── ProjectList.css

│ │ ├── TrainingList.css

│ │ ├── DepartmentHeads.css

│ │ ├── JobTitleList.css

│ ├── App.js

│ ├── index.js

│ ├── index.css

└── package.json