Case Study 1: Online Bookstore Application

Objective: Build a small bookstore app where users can browse, search, and add books to a shopping cart.

Concepts Covered:

- Class and Functional components
- Event handling (add to cart, search)
- Passing arguments between components (book details)
- Multiple components interacting (BookList, BookCard, Cart)
- Backend integration using JSON-server for books data

Tasks:

1. Create `BookList` functional component fetching data from JSON-server.
2. Use `BookCard` class component to display each book, with `Add to Cart` button.
3. Handle `onClick` event to add selected books to the cart.
4. Pass selected book info from `BookCard` to `Cart` component.
5. Implement search functionality to filter books by title or author.

Case Study 2: Employee Management Dashboard

Objective: Build a dashboard to manage employee profiles and leaves.

Concepts Covered:

- Class and Functional components
- Passing arguments between components
- Multiple components interaction (EmployeeList, EmployeeDetails, LeaveForm)
- Form handling using Formik and Yup for validation
- Backend integration using JSON-server

Tasks:

1. Create `EmployeeList` component to fetch and display employee names.
2. On clicking an employee, pass ID to `EmployeeDetails` component to show full profile.

3. Implement `LeaveForm` with Formik and Yup for adding leave requests.
4. Validate leave dates and reason using Yup schema.
5. Update JSON-server with new leave requests and reflect changes in the dashboard.

Case Study 3: Movie Booking Application

Objective: Build a movie booking app with real-time seat selection.

Concepts Covered:

- Event handling (seat selection, booking)
- Passing arguments between components (selected seats)
- Context API for global cart/seat selection state
- React Router for navigation between `Movies`, `Seats`, and `Payment` pages
- Backend integration for movie and seat data

Tasks:

1. Create `Movies` component fetching available movies from JSON-server.
2. Navigate to `Seats` page using React Router when a movie is selected.
3. Use Context API to store selected seats across components.
4. Handle seat selection event and disable booked seats.
5. Navigate to `Payment` component and display selected seats and total cost.

Case Study 4: Todo List with Higher-Order Components

Objective: Build a Todo List app with reusable functionality implemented via HOC.

Concepts Covered:

- Functional components
- Event handling (`add`, `delete`, `mark completed`)
- HOC for reusable functionality (e.g., logging actions or error handling)
- Multiple component interaction (TodoList, TodoItem, AddTodoForm)
- Backend integration for persisting todos

Tasks:

1. Create `TodoList` component displaying a list of todos.
2. Implement `AddTodoForm` component using Formik and Yup for validation.
3. Create a HOC `withLogger` that logs whenever a todo is added or removed.
4. Pass functions from parent to child components for event handling.
5. Save and fetch todos from JSON-server API.

Case Study 5: E-commerce Checkout Flow

Objective: Build a checkout flow for an e-commerce app.

Concepts Covered:

- Class and Functional components
- Event handling for quantity changes, apply coupon
- Passing data between multiple components (Cart, CheckoutForm, OrderSummary)
- Formik & Yup for checkout form validation
- React Router for multi-step checkout pages
- Context API for global cart state

Tasks:

1. Implement `Cart` component displaying all selected items and quantities.
2. Create `CheckoutForm` using Formik and Yup to capture shipping and payment info.
3. Pass cart items and total cost to `OrderSummary` component.
4. Use React Router to navigate between `Cart`, `CheckoutForm`, and `OrderSummary` pages.
5. Use Context API to maintain cart state across multiple pages.