React Training Plan for Trainees

Purpose

This training plan is designed to upskill trainees from basic React knowledge to production-ready application development. The focus is on **clean architecture, scalability, and real-world best practices**, not just component building.

Learning Outcomes

By the end of the training, trainees will be able to:

- Build React applications with confidence
- Organize code using feature-based architecture
- Manage state effectively at different levels
- Separate UI, logic, and data layers cleanly
- Write production-ready, maintainable React code

Curriculum Breakdown

1. React Fundamentals

- JSX syntax and rules
- Functional components
- Props and component composition
- State management using useState
- Side effects using useEffect
- Conditional rendering and lists
- Basic routing using react-router-dom

Hands-on:

- Build reusable UI components
- Simple multi-page React app

2. Application Structure & Architecture

- Common mistakes in beginner folder structures
- Feature-based (domain-driven) project organization
- Understanding features, shared modules, and app layer
- Deciding what belongs inside a feature
- Avoiding over-shared components

Hands-on:

- Refactor an existing app into feature-based structure
- Design folder structure for a SaaS-style application

3. State Management Strategy

- Types of state:
    - Local UI state
    - Client/global state
    - Server state
- When to lift state and when not to
- Context API: correct and incorrect use cases
- Redux Toolkit fundamentals
- Feature-based Redux slices

Hands-on:

- Implement Context for authentication/user state
- Implement Redux Toolkit for global state

4. Logic & UI Separation

- Identifying business logic vs presentation
- Writing custom hooks (useAuth, useFetch, etc.)
- Keeping components focused on rendering only
- Avoiding API calls and heavy logic inside components
- Reusability through hooks

Hands-on:

- Extract logic from components into custom hooks
- Refactor bloated components into clean UI components

5. Data Access & Services Layer

- Importance of a dedicated data layer
- Difference between lib/ and services/
- Axios setup with interceptors
- Centralized API configuration
- Feature-level service files
- Error handling and loading patterns

Hands-on:

- Configure Axios with interceptors
- Move all API calls into services
- Connect services → hooks → components

6. Production Readiness

- Naming conventions and coding discipline
- File and export consistency
- Performance basics:

    - Memoization
    - Lazy loading
    - Avoiding unnecessary re-renders
- Handling loading, empty, and error states
- Error boundaries
- Testing basics (unit & component testing)
- Accessibility fundamentals

Hands-on:

- Add loading & error states across the app
- Optimize selected components
- Write basic tests

Evaluation Criteria

Trainees will be evaluated on:

- Code readability and structure
- Correct state management decisions
- Separation of concerns
- Folder and naming discipline
- Ability to explain architectural decisions

Final Assignment

Build a small production-style React application with:

- Feature-based folder structure
- Centralized API services
- Custom hooks for business logic
- Global and local state used appropriately
- Proper loading, error, and empty states

Trainer Notes

- Encourage **why-based thinking**, not copy-paste coding
- Perform regular code reviews
- Focus on real-world scenarios and trade-offs
- Emphasize maintainability over shortcuts

Expected Skill Level After Training

Trainees completing this plan will be capable of:

- Working on mid-to-large React codebases
- Following team-level architectural standards
- Writing clean, scalable, and maintainable React applications