REACT COURSE

1. Basics

1.1 Introduction to React

- **Topics**:
    - What is React?
    - Benefits of using React.
    - Key concepts: Components, JSX, Virtual DOM.
    - Comparison with other frameworks (e.g., Angular, Vue).
- **Exercise**: Research the history of React and write a brief report on why React became popular.

1.2 Setting Up Environment

- **Topics**:
    - Installing Node.js and npm.
    - Setting up VS Code.
    - Creating a React app with Create React App.
    - Folder structure overview.
- **Exercise**: Create your first React app using Create React App and explore the project structure.

1.3 JSX

- **Topics**:
    - Understanding JSX syntax.
    - Embedding expressions in JSX.
    - JSX vs. HTML.
    - JSX gotchas (e.g., className vs. class, self-closing tags).
- **Exercise**: Create a simple React component that displays your name and a list of your favorite hobbies using JSX.

1.4 Components

- **Topics**:
    - Function components vs. Class components.
    - Understanding props and how to pass data.

  - Managing state with useState.
  - Component lifecycle (Class components).
  - Splitting components into smaller parts.
- **Exercise**: Build a simple counter app with a button to increase and decrease the count, demonstrating state management.

1.5 Events & Handling

- **Topics**:
  - Handling events in React.
  - Passing arguments to event handlers.
  - Synthetic events in React.
- **Exercise**: Create a form with input fields and handle the form submission event to log the input data.

1.6 Conditional Rendering

- **Topics**:
  - Using if-else in JSX.
  - Ternary operators.
  - Logical && operator for conditional rendering.
- **Exercise**: Create a login form that shows either a "Welcome" message or a "Please login" message based on the state.

1.7 Lists & Keys

- **Topics**:
  - Rendering lists with map().
  - Understanding the importance of keys in lists.
  - Handling dynamic lists.
- **Exercise**: Build a to-do list app where users can add and remove items from a list.

---

2. Intermediate

2.1 Hooks

- **Topics**:
    - Introduction to Hooks.
    - useState: Managing state in functional components.
    - useEffect: Side effects, fetching data, cleanup functions.
    - Custom Hooks: Reusable logic across components.
- **Exercise**: Fetch data from an API (e.g., JSONPlaceholder) using useEffect and display it in a list.

2.2 Forms

- **Topics**:
    - Controlled vs. uncontrolled components.
    - Handling form submissions.
    - Validating form inputs.
    - Managing form state with useState.
- **Exercise**: Create a registration form with validation (e.g., required fields, email format) and display error messages.

2.3 Routing

- **Topics**:
    - Introduction to React Router.
    - Setting up routes, Route, Link components.
    - Nested routes.
    - URL parameters and query strings.
- **Exercise**: Build a multi-page app with React Router, such as a simple blog with a home page, about page, and individual post pages.

2.4 Context API

- **Topics**:
    - Introduction to the Context API.
    - Creating and providing context.
    - Consuming context in nested components.
    - Using context with functional components.

- **Exercise**: Implement a theme toggle feature in your app (light/dark mode) using the Context API.

2.5 Styling

- **Topics**:
  - Inline styles and CSS in JS.
  - CSS Modules for scoped styling.
  - Using Tailwind CSS for utility-first styling.
  - Styling libraries (e.g., styled-components).
- **Exercise**: Style a component using Tailwind CSS and compare it with regular CSS or CSS Modules.

---

3. Advanced

3.1 Higher-Order Components (HOCs)

- **Topics**:
  - Understanding HOCs.
  - Creating and using HOCs for reusability.
  - Common use cases for HOCs.
- **Exercise**: Create an HOC that adds authentication logic to protect certain routes in your app.

3.2 Render Props

- **Topics**:
  - Understanding the render props pattern.
  - Creating components that use render props.
  - When to use render props vs. HOCs.
- **Exercise**: Create a component that fetches data and passes it to children using a render prop.

3.3 Refs

- **Topics**:
  - Introduction to Refs in React.

   - Using useRef to access DOM elements.
   - Controlling focus and text selection with Refs.
   - Forwarding refs to child components.
- **Exercise**: Build a form where the first input field is automatically focused on page load using refs.

3.4 Portals

- **Topics**:
   - What are React Portals?
   - Creating and using Portals to render content outside the DOM hierarchy.
   - Common use cases: Modals, tooltips.
- **Exercise**: Implement a modal component using React Portals.

3.5 Error Boundaries

- **Topics**:
   - Understanding error boundaries in React.
   - Catching errors in class components with error boundaries.
   - Displaying fallback UI for errors.
- **Exercise**: Implement an error boundary component that displays a custom error message when a component fails.

3.6 Suspense & Lazy Loading

- **Topics**:
   - Introduction to code splitting.
   - Using React.lazy() for lazy loading components.
   - Handling loading states with Suspense.
   - Combining Suspense with data fetching.
- **Exercise**: Implement lazy loading for routes in a React Router application.

---

4. Master Level

4.1 Performance Optimization

- **Topics**:
  - React.memo for memoizing components.
  - useMemo and useCallback for optimizing rendering.
  - Avoiding unnecessary re-renders.
  - Analyzing performance with React DevTools.
- **Exercise**: Optimize a list of items that re-renders on every input change by using React.memo and useCallback.

4.2 Testing

- **Topics**:
  - Introduction to testing in React.
  - Writing unit tests with Jest.
  - Testing components with React Testing Library.
  - Integration testing and mocking API calls.
- **Exercise**: Write unit tests for a simple React component and mock an API request in the tests.

4.3 Server-Side Rendering (SSR)

- **Topics**:
  - Introduction to SSR.
  - Differences between SSR and CSR.
  - Setting up a Next.js project.
  - Implementing SSR and static site generation (SSG).
- **Exercise**: Build a simple blog with Next.js, using SSR for the individual post pages.

4.4 State Management

- **Topics**:
  - Introduction to Redux.
  - Setting up Redux with React.
  - Using Redux Toolkit for better state management.

    - Asynchronous actions with Redux Thunk or Redux Saga.
- **Exercise**: Implement global state management for a shopping cart using Redux and Redux Toolkit.

4.5 React with TypeScript

- **Topics**:
    - Introduction to TypeScript in React.
    - Typing components and props.
    - Using interfaces and types.
    - Handling complex state and events with TypeScript.
- **Exercise**: Convert an existing React project to TypeScript and ensure all components are properly typed.

4.6 Custom Hooks

- **Topics**:
    - Building custom hooks for reusable logic.
    - Handling complex logic with custom hooks.
    - Sharing stateful logic across components.
- **Exercise**: Create a custom hook for form validation and reuse it in multiple forms.

4.7 React Native Basics

- **Topics**:
    - Introduction to React Native.
    - Setting up the React Native environment.
    - Core components and APIs.
    - Navigation in React Native apps.
- **Exercise**: Build a simple React Native app that displays a list of items and allows users to navigate between screens.

---

5. Projects

5.1 Basic Projects

- **Todo List**: Manage tasks with add, delete, and edit features.
- **Weather App**: Fetch and display weather data using a public API.

5.2 Intermediate Projects

- **E-commerce Site**: Product listing, filtering, and cart management.
- **Blog Platform**: CRUD operations for posts, comments, and user authentication.

5.3 Advanced Projects

- **Social Media Dashboard**: Analytics and data visualization with charts.
- **Real-time Chat App**: WebSockets or Firebase for real-time messaging.

5.4 Master Level Projects

1. Full-Stack E-commerce Platform

- **Description**: Build a complete e-commerce platform with a React frontend and Node.js/Express backend. Implement features such as product listings, user authentication, shopping cart, order management, payment gateway integration, and admin panel for managing products, users, and orders.
- **Key Concepts**:
  - **Frontend**: React, Redux Toolkit, Tailwind CSS for styling, React Router, and component libraries like Material UI.
  - **Backend**: Node.js, Express, MongoDB for database management, JWT for authentication, REST API design.
  - **Additional**: Payment integration (Stripe or PayPal), responsive design, error handling, and security best practices.

2. Progressive Web App (PWA)

- **Description**: Create a PWA using React that provides offline capabilities, push notifications, and fast load times. This app could be anything from a note-taking app to a task manager or news reader.
- **Key Concepts**:
  - **PWA Features**: Service workers for offline support, Web App Manifest for installation, caching strategies.
  - **Frontend**: React, React Router, Tailwind CSS, IndexedDB or local storage for offline data management.
  - **Additional**: Deploy the app with proper SSL, ensure cross-browser compatibility, and use Lighthouse for performance auditing.