Introduction to React.js

THEORY EXERCISE

- **Question 1**: What is React.js? How is it different from other JavaScript frameworks and libraries?
- **Question 2**: Explain the core principles of React such as the virtual DOM and component-based architecture.
- **Question 3**: What are the advantages of using React.js in web development?

LAB EXERCISE

- Task:
  - Set up a new React.js project using `create-react-app`.
  - Create a basic component that displays "Hello, React!" on the web page.

2. JSX (JavaScript XML)

THEORY EXERCISE

- **Question 1**: What is JSX in React.js? Why is it used?
- **Question 2**: How is JSX different from regular JavaScript? Can you write JavaScript inside JSX?
- **Question 3**: Discuss the importance of using curly braces `{}` in JSX expressions.

LAB EXERCISE

- Task:
  - Create a React component that renders the following JSX elements:
    - A heading with the text "Welcome to JSX".
    - A paragraph explaining JSX with dynamic data (use curly braces to insert variables).

---

3. Components (Functional & Class Components)

THEORY EXERCISE

- **Question 1**: What are components in React? Explain the difference between **functional components** and **class components**.
- **Question 2**: How do you pass data to a component using props?
- **Question 3**: What is the role of `render()` in class components?

LAB EXERCISE

- Task 1:
  - Create a functional component `Greeting` that accepts a `name` as a prop and displays "Hello, [name]!".
- Task 2:
  - Create a class component `WelcomeMessage` that displays "Welcome to React!" and a `render()` method.

---

4. Props and State

THEORY EXERCISE

- **Question 1**: What are props in React.js? How are props different from state?
- **Question 2**: Explain the concept of state in React and how it is used to manage component data.
- **Question 3**: Why is `this.setState()` used in class components, and how does it work?

LAB EXERCISE

- Task 1:
  - Create a React component `UserCard` that accepts `name`, `age`, and `location` as props and displays them in a card format.
- Task 2:
  - Create a `Counter` component with a button that increments a count value using React state. Display the current count on the screen.

---

5. Handling Events in React

THEORY EXERCISE

- **Question 1**: How are events handled in React compared to vanilla JavaScript? Explain the concept of synthetic events.
- **Question 2**: What are some common event handlers in React.js? Provide examples of `onClick`, `onChange`, and `onSubmit`.
- **Question 3**: Why do you need to bind event handlers in class components?

LAB EXERCISE

- Task 1:
  - Create a button in a React component that, when clicked, changes the text from "Not Clicked" to "Clicked!" using event handling.
- Task 2:
  - Create a form with an input field in React. Display the value of the input field dynamically as the user types in it.

---

6. Conditional Rendering

THEORY EXERCISE

- **Question 1**: What is conditional rendering in React? How can you conditionally render elements in a React component?
- **Question 2**: Explain how `if-else`, ternary operators, and `&&` (logical AND) are used in JSX for conditional rendering.

LAB EXERCISE

- Task 1:
  - Create a component that conditionally displays a login or logout button based on the user's login status.
- Task 2:
  - Implement a component that displays a message like "You are eligible to vote" if theuser is over 18, otherwise display "You are not eligible to vote."

7. Lists and Keys

THEORY EXERCISE

- **Question 1**: How do you render a list of items in React? Why is it important to use keys when rendering lists?
- **Question 2**: What are keys in React, and what happens if you do not provide a unique key?

LAB EXERCISE

- Task 1:
  - Create a React component that renders a list of items (e.g., a list of fruit names). Use the `map()` function to render each item in the list.
- Task 2:
  - Create a list of users where each user has a unique `id`. Render the user list using React and assign a unique `key` to each user.

---

8. Forms in React

THEORY EXERCISE

- **Question 1**: How do you handle forms in React? Explain the concept of controlled components.
- **Question 2**: What is the difference between controlled and uncontrolled components in React?

LAB EXERCISE

- Task 1:
  - Create a form with inputs for `name`, `email`, and `password`. Use state to control the form and display the form data when the user submits it.
- Task 2:
  - Add validation to the form created above. For example, ensure that the `email` input contains a valid email address.

---

9. Lifecycle Methods (Class Components)

THEORY EXERCISE

- **Question 1**: What are lifecycle methods in React class components? Describe the phases of a component's lifecycle.
- **Question 2**: Explain the purpose of `componentDidMount()`, `componentDidUpdate()`, and `componentWillUnmount()`.

LAB EXERCISE

- Task 1:
  - Create a class component that fetches data from an API when the component mounts using `componentDidMount()`. Display the data in the component.
- Task 2:
  - Implement a component that logs a message to the console when it updates using `componentDidUpdate()`. Log another message when the component unmounts using `componentWillUnmount()`.

---

10. Hooks (useState, useEffect, useReducer, useMemo, useRef, useCallback)

THEORY EXERCISE

- **Question 1**: What are React hooks? How do `useState()` and `useEffect()` hooks work in functional components?
- **Question 2**: What problems did hooks solve in React development? Why are hooks considered an important addition to React?
- **Question 3**: What is useReducer ? How we use in react app?
- **Question 4:** What is the purpose of useCallback & useMemo Hooks?
- **Question 5:** What's the Difference between the useCallback & useMemo Hooks?
- **Question 6 :** What is useRef ? How to work in react app?

LAB EXERCISE

- Task 1:
  - Create a functional component with a counter using the `useState()` hook. Include buttons to increment and decrement the counter.
- Task 2:
  - Use the `useEffect()` hook to fetch and display data from an API when the component mounts.
- Task 3:
  - Create react app with use of useSelector & useDispatch.
- Task 4:
  - Create react app to avoid re-renders in react application by useRef ?

---

11. Routing in React (React Router)

THEORY EXERCISE

- **Question 1**: What is React Router? How does it handle routing in single-page applications?
- **Question 2**: Explain the difference between `BrowserRouter`, `Route`, `Link`, and `Switch` components in React Router.

LAB EXERCISE

- Task 1:
  - Set up a basic React Router with two routes: one for a Home page and one for an

About page. Display the appropriate content based on the URL.

- Task 2:
  - Create a navigation bar using React Router's `Link` component that allows users to switch between the Home, About, and Contact pages.

12. React – JSON-server and Firebase Real Time Database

THEORY EXERCISE

- **Question 1:** What do you mean by RESTful web services?
- **Question 2:** What is Json-Server? How we use in React ?
- **Question 3**: How do you fetch data from a Json-server API in React? Explain the role of `fetch()` or `axios()` in making API requests.
- **Question 4:** What is Firebase? What features does Firebase offer?
- **Question 5**: Discuss the importance of handling errors and loading states when working withAPIs in React

LAB EXERCISE

- Task 1:
  - Create a React component that fetches data from a public API (e.g., a list of users) and displays it in a table format.
  - Create a React app with Json-server and use Get , Post , Put , Delete & patch method on Json-server API.
- Task 2:
  - Create a React app crud and Authentication with firebase API.
  - Implement google Authentication with firebase API.
- Task 3:
  - Implement error handling and loading states for the API call. Display a loading spinner while the data is being fetched.

---

13. Context API

THEORY EXERCISE

- **Question 1**: What is the Context API in React? How is it used to manage global state across multiple components?
- **Question 2**: Explain how `createContext()` and `useContext()` are used in React for sharing state.

LAB EXERCISE

- Task 1:
  - Create a simple theme toggle (light/dark mode) using the Context API. The theme state should be shared across multiple components.
- Task 2:
  - Use the Context API to create a global user authentication system. If the user is

logged in, display a welcome message; otherwise, prompt them to log in.

---

14. State Management (Redux, Redux-Toolkit or Recoil)

THEORY EXERCISE

- **Question 1**: What is Redux, and why is it used in React applications? Explain the core concepts of actions, reducers, and the store.
- **Question 2**: How does Recoil simplify state management in React compared to Redux?

LAB EXERCISE

- Task 1:
  - Create a simple counter application using Redux for state management. Implement actions to increment and decrement the counter.
- Task 2:
  - Build a todo list application using Recoil for state management. Allow users to add, remove, and mark tasks as complete.
- Task 3:
  - Build a crud application using Redux-Toolkit for state management. Allow users to add,remove, delete and update.