**Assignment: State Management in React**

**Group 3**: Understanding and Implementing State Management in Functional Components

Objective

The purpose of this assignment is to explore **state management** within **React functional components**. It covers the foundational concepts of **state** and **React hooks** (specifically `useState`), as well as how to build and manage forms in React. This document provides a theoretical overview of state, hooks, and conditional rendering.

Key Concepts

1. **State in React**

- Definition: In React, "state" refers to data that changes over the component’s lifecycle. Managing state is crucial because it controls how data flows and updates within the component.

- Why It Matters: State enables components to dynamically reflect the latest data, responding to user actions or other triggers, ensuring the app updates in real-time.

- Example: In a form, the state could include the data entered in each input field, such as name, email, or age.

2. **Introduction to Hooks**

- What Are Hooks? React introduced hooks to provide functional components with state management and lifecycle features, which were previously only available in class components.

- useState Hook:

- Purpose: `useState` allows us to add state to functional components.

- How It Works: It returns an array containing two values:

1. The current state.

2. A function to update that state.

- *Syntax*:

javascript

const [state, setState] = useState(initialValue);

- `state`: Current value of the state.

- `setState`: Function to update the state.

- `initialValue`: The initial value of the state variable.

3. **Managing Component State and Rendering**

- React components automatically re-render when state updates occur.

- This reactivity allows us to update UI elements in response to user interactions or external data changes.

- For example, in a form component, updating the state with new input values will trigger a re-render, showing the latest data instantly.

**Practical Assignment**

As a practical exercise, the assignment involves creating a \*\*simple form\*\* in React with the following goals:

1. State Management with useState:

2. Handling Form Submission:

3. Conditional Rendering:

**Implementation Steps**

1. Setup and Import React: Initialize a basic React app and import `useState` from React.

2. Create the Form Component:

- Define a function-based component that includes form fields for name and email.

- Initialize state for form fields using `useState`.

3. Handle Form Changes:

- Define an `onChange` handler to update state as users input data.

4. Submit Form:

- Define an `onSubmit` handler to capture form data when submitted and update the submission state.

5. Implement Conditional Rendering:

- Use conditional rendering to show a thank-you message after the form is submitted.

References

1. \*\*React Documentation on State and Lifecycle\*\*: [https://reactjs.org/docs/state-and-lifecycle.html](https://reactjs.org/docs/state-and-lifecycle.html)

2. \*\*React Documentation on Hooks\*\*: [https://reactjs.org/docs/hooks-intro.html](https://reactjs.org/docs/hooks-intro.html)

3. \*\*React useState Hook\*\*: [https://reactjs.org/docs/hooks-reference.html#usestate](https://reactjs.org/docs/hooks-reference.html#usestate)

4. \*\*Understanding Conditional Rendering in React\*\*: [https://reactjs.org/docs/conditional-rendering.html](https://reactjs.org/docs/conditional-rendering.html)

---

This section includes references to key React documentation that can help you understand the concepts covered in this assignment more deeply.