NORAIZ RANA

Task #9

Topic

React Props & State Management basics

Date:

14 July 2025

SUBMITTED TO:

APPVERSE TECHNOLOGIES

📓 Learning Report: React Props and State Management Basics

1. Introduction to React Props and State

In React, **props** and **state** are two core concepts that enable component-based architecture, interactivity, and data flow in applications.

- **Props (short for properties)**: Used to pass data from one component to another, typically from parent to child.
- **State**: Refers to data that is managed within a component and can change over time due to user actions or other events.

Understanding and correctly using props and state is essential for building dynamic, interactive, and reusable React components.

---

2. What are Props in React?

Props are **read-only** objects passed to components to customize or configure their behavior or appearance.

Characteristics of Props:

- Props are **immutable**: once passed, they cannot be modified by the receiving component.
- Props help in **reusing components** by giving them different data.
- Props are passed from **parent to child**.

Example:

```
function Welcome(props) {
  return <h1>Hello, {props.name}!</h1>;
```

```
}

<Welcome name="Noraiz" />
```

In the example above:

- The `Welcome` component receives a `name` prop.
- It renders a message dynamically based on the passed value.

---

3. What is State in React?

State is a built-in object in React components that holds dynamic data or information that affects how the component behaves or renders.

Key Points:

- State is **mutable** and can be updated.
- State changes trigger **re-rendering** of the component.
- Each component can maintain its own **local state** using the `useState()` hook (in functional components).

Example using `useState`:

```
import { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <>
      <p>Count: {count}</p>
```

```
      <button onClick={() => setCount(count +
1)}>Increase</button>
    </>
  );
}
```

In this example:

- `useState(0)` initializes the `count` state with a value of 0.
- Clicking the button updates the state, causing the UI to re-render.

---

4. Difference Between Props and State

Feature	Props	State
Mutability	Immutable	Mutable
Ownership	Passed from parent	Managed within the component
Use Case	Configure child components	Handle dynamic behavior
Changes	Cannot be changed internally	Can be updated using `setState()`