**Experiment 8**

**Aim:**

**Tools:** React, Visual Studio Code

**Learning Objectives:**

* Proficiency in React Basics: Understand and apply fundamental concepts of React, including components, props, and state.
* Dynamic Data Rendering: Demonstrate the ability to dynamically render a list of items using React’s mapping function.**Tools:** HTML, CSS, JavaScript, Text Editor/IDE

**Theory:**

**Introduction to React**

React is a popular JavaScript library developed by Facebook for building user interfaces, particularly single-page applications. Its component-based architecture allows developers to create reusable UI components that manage their own state, leading to better organization of code and improved performance.

**Core Concepts**

1. **Components**: The building blocks of any React application are components. They can be class-based or functional. Functional components are simpler and often preferred in modern development due to their readability and the introduction of hooks.
2. **JSX**: React uses JSX (JavaScript XML) to describe what the UI should look like. JSX allows developers to write HTML-like syntax within JavaScript, making the code more intuitive.
3. **Props**: Short for properties, props are the mechanism by which data is passed from parent to child components. They are immutable and help maintain the flow of data in a unidirectional manner.
4. **State**: Unlike props, state is mutable and is managed within the component. State allows components to respond to user input and change over time, making the app dynamic.

**Creating a React App**

To create a new React app, developers often use Create React App (CRA), a tool that sets up a new React project with sensible defaults. The command npx create-react-app my-app initializes a new project folder, complete with the necessary configurations.

Once the app is created, developers can focus on structuring their components. For displaying a list of items, the following steps are essential:

1. **Component Structure**: Define a main component (e.g., App) that will hold the state for the list of items. Each item can be represented by a separate component (e.g., Item).
2. **State Management**: Use React’s useState hook to create state for the items array. For example:

javascript

Copy code

const [items, setItems] = useState(['Item 1', 'Item 2', ..., 'Item 10']);

1. **Rendering Lists**: Utilize the map function to iterate over the array of items and render them:

javascript

Copy code

return (

<div>

{items.map((item, index) => (

<Item key={index} name={item} />

))}

</div>

);

1. **Styling and Interactivity**: Style components using CSS and add interactivity, such as buttons to add or remove items from the list.

**Learning Outcomes:**

* Component Creation: Successfully create functional components in React to display data.
* State Management: Utilize React state to manage and update the list of items efficiently.

**Implementation Code:**

**index.js**

**import React from 'react';**

**import ReactDOM from 'react-dom/client';**

**import './index.css';**

**import App from './App';**

**const root = ReactDOM.createRoot(document.getElementById('root'));**

**root.render(**

**<React.StrictMode>**

**<App />**

**</React.StrictMode>**

**);**

**app.js**

**import './App.css';**

**export default function App() {**

**return (**

**<div className="App">**

**<h1>Here are the products that we sell:</h1>**

**<ul>**

**<li>Electronics</li>**

**<li>Fashion</li>**

**<li>Home Appliances</li>**

**<li>Furniture</li>**

**<li>Beauty & Personal Care</li>**

**<li>Sports & Fitness</li>**

**<li>Books & Stationary</li>**

**<li>Groceries & Essentials</li>**

**<li>Toys & Games</li>**

**<li>Automative Accessories</li>**

**</ul>**

**</div>**

**);**

**}**

**app.css**

**body {**

**display: flex;**

**align-items: center;**

**justify-content: center;**

**font-family: 'Roboto', sans-serif;**

**}**

**Implentation Output:**

