Week-7.1 Deliverable.1

* React : Fundamental concept -

→ React is a popular javascript library for building user interface, particularly, single-page applications, where efficiency, speed, & dynamic content updates are crucial. It was developed by Facebook & is maintained by a community of developers. React's core philosophy is based on the principles of declarative programming, component-based architecture, & efficient rendering using a virtual DOM. This report will cover the fundamental concept of React, including JSX, components, states, & props, along with relevant code examples.

JSX (JavaScript XML) -

JSX is a syntax extension for javascript that looks similar to XML or HTML. It is used with React to describe the UI structure. JSX allows us to write HTML elements in JavaScript & place them in DOM without using methods. ~~like~~

Examples of JSX -

```
import React from 'react';
import ReactDOM from 'reactdom';
const ele = <h1>Hello World </h2>;
ReactDOM.render(element, document.getElementById('root'));
```

Key Features :

1. Embedding Expressions
2. JSX is an Expression too.
3. Specifying Attributes with JSX
4. children in JSX.

components -

components are the building blocks of a React application. A component is a javascript function or class that optionally accepts inputs (known as "props") & returns a React element that describes how a section of UI should appear.

Types of Components:

1. Function Components - These are javascript functions that return JSX.

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

2. Class Components - These are ES6 classes that extend 'React. Component' & have a 'render' method.

Code -
```
class Welcome extends React. Component {
  render(){
     return <h1> Hello, {this.props.name}</h1>)
  }
}
```

Using Components -

components can be used within other components to build complex UI. This process is called "composition".

Code -
```
function App(){
  return (
     <div>
       <welcome name = "Alice" />
       <Welcome name = "Bob" />
       <Welcome name = "charlie" />
  );
}
```

#props →

props (short for "properties") are ready-only attributes passed from a parent component to a child component. Props allow components to be a dynamec & reusable.

Examples of Props -
```
function Greetings (props) {
    return <h1> Hello, {props.name} </h1>
}
function App() {
   return (
      <div>
         <Greeting name= "Alice" />
         <Greeting name= "Bob" />
      </div>
   );
}
```

State →

State is a builtin object used to store data that affects the rendering of a component. Unlike props, which are immutable, state can be changed asynchronously. when the state of a component changes, React re-renders the components to reflect the new state.

Example of State -
```
class clock extends React.Component {
   constructor (props) {
   super(props);
  this.state = {date: new Date()};
}
component DidMount () {
this.timerID = setInterval (() =>
      this.tick(), 1000);
}
componentWillUnmount() {
 clearInterval (this.timerID);
}
tick() {
 this.setState({
    date: new Date()
  });
}
```

Using State in Function Components:-

```
import React, {useState} from 'react';
function Counter(){
   const [count, setCount] = useState(0);
return (
   <div>
    <p> You clicked {count} times </p>
    <button onClick ={() => setCount (count+1) }>
      click me
      </button>
      </div>
    );
}
```