Week 1.1
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.

Examples of JSX :-

```
import React from 'react';
import ReactDOM from 'react-dom';
Const ele = <h1> Hello World </H1>;
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 components. Props allow components to be dynamic & 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 remembers the component's to reflect the new state.

Ex. of States:-

```
class clock extends React.Component
{ constructor (props)
  { super (props);
    this.state = { date: new Date() };
  }
componentDidMount()
{ this.timerID = setInterval(()=>
    this.tick(), 1000);
}
componentWillUnmount()
{ clearInterval (this.timerId);
}
tick()
{ this.setState ({
    date: newDate()
  });
}
```