**Day -1**

Learning objectives:

* JSX
* Components - Class and Function,props
* UseState

Task:

* Creating a stopwatch using these learnings.

JSX:

It is a syntax extension used in React that lets you write HTML-like code directly inside JavaScript. It makes building user interfaces simpler by combining the structure (like HTML) and logic (JavaScript) in one place. Instead of manually creating elements with JavaScript, JSX allows you to write tags such as <div>, <h1>, or custom components, making the code more readable and expressive.

JSX supports embedding JavaScript expressions inside curly braces {} to dynamically display values, run functions, or use variables within the markup. Although it looks like HTML, it is not exactly HTML — it compiles down to React’s createElement calls which generate virtual DOM elements.

Using JSX speeds up development and helps maintain a clean, declarative code style when building React components. It also supports attributes (like className, style) and conditional rendering, making UI creation flexible and intuitive.

Components:

In React, components are the fundamental building blocks used to create user interfaces. They are reusable, self-contained pieces of code that define how a part of the UI should look and behave. Components can be created as functions (functional components) or classes (class components), though functional components are more common in modern React due to the simplicity and use of hooks. Each component can accept input data called **props**, and functional components can also manage internal data using **state**. By breaking the UI into multiple components, React makes it easier to build, maintain, and reuse parts of an application efficiently.

UseState:

UseState is a core feature in React that allows functional components to hold and manage their own state, which means they can remember and respond to user interactions or other dynamic changes. State refers to any data that should change over time, like button clicks, form inputs, or toggling visibility. With useState, each component can store its own piece of information and update it independently. The hook provides two elements: the current value of the state and a function to change that value. When the function is called to update the state, React automatically re-renders the component so the UI stays in sync with the underlying data. This behavior makes the application feel interactive and responsive. Since useState is simple and powerful, it is one of the most commonly used tools in React, helping developers create components that are both flexible and easy to understand.