**WEEK 6**

**5.ReactJS-HOL**

**OBJECTIVES:**

**1.Understanding the Need for Styling React Components**

**Styling in React is essential to:**

* Make UI visually appealing and user-friendly
* Provide a consistent layout and branding
* Highlight interactive elements like buttons or form fields
* Improve accessibility and usability

In React, unlike traditional HTML/CSS websites, components are modular and reusable. So, styling must also follow this modular approach to:

* Avoid style conflicts between components
* Enable component-specific styles
* Make components self-contained and portable

React supports various styling methods, such as:

* Traditional CSS files
* CSS Modules
* Inline styles
* Styled-components (CSS-in-JS libraries)

**2.Working with CSS Modules**

CSS Modules allow you to scope CSS locally to a specific component. This prevents global CSS conflicts and is useful in large applications.

**How it works:**

1. Create a CSS file with the .module.css extension (e.g., App.module.css)
2. Import it in your component:

import styles from './App.module.css';

1. Apply styles using:

<div className={styles.container}></div>

CSS Modules automatically generate unique class names behind the scenes, like App\_container\_\_2hXtr.

**Working with Inline Styles**

Inline styles in React are written as JavaScript objects and assigned to components using the style attribute.

**Example:**

const headingStyle = {

color: 'blue',

fontSize: '24px',

marginBottom: '10px'

};

function MyComponent() {

return <h1 style={headingStyle}>Hello, styled with inline styles!</h1>;

}

 Property names are **camelCased** (e.g., backgroundColor, not background-color)

 Values must be in **quotes or variables**, especially for strings