**REACT  
  
1.ReactJS-HOL   
  
Define SPA and Its Benefits**

A Single Page Application (SPA) is a web application that loads a single HTML file and dynamically updates the page as the user interacts with it, without reloading from the server.

Benefits:-

Fast interactions after initial load

Smoother user experience (no full page reloads)

Efficient front-end routing using JavaScript

Ideal for mobile-friendly apps

**Define React and Identify Its Working**

React is a JavaScript library developed by Facebook for building dynamic user interfaces. It is:

**Component-based**: UI is broken into reusable building blocks

**Declarative**: Focuses on what UI should look like

**Efficient**: Uses Virtual DOM for faster rendering

How React Works:

Splits the UI into reusable components

Uses Virtual DOM to track changes

Updates only the changed parts of the real DOM

**Differences Between SPA and MPA**

| Feature | SPA | MPA |
| --- | --- | --- |
| Page Load | Loads single HTML file | Loads a new HTML file per page |
| Speed | Fast after initial load | Slower due to page reloads |
| Navigation | Client-side routing (JavaScript) | Server-side routing |
| SEO | Difficult without SSR | Easier to optimize for SEO |

**Pros & Cons of SPA**

Pros:

Fast and interactive user experience

Lower server load

Rich, app-like interface

Cons:

SEO optimization is difficult

Large initial load time

Complex app state management

**Explain About React**

React enables developers to create highly interactive web apps. It:

Emphasizes component reuse

Follows one-way data flow

Uses JSX to combine HTML with JavaScript

**What is Virtual DOM?**

The **Virtual DOM** is a lightweight in-memory copy of the actual DOM. React uses it to:

Quickly detect changes in the UI

Calculate the most efficient way to update the browser

Improve rendering performance

**Features of React**

JSX (JavaScript XML)

Component-based architecture

Virtual DOM

One-way data binding

Fast rendering

Rich ecosystem with tools like React Router, Redux

**App.js Code**

import React from 'react';

function App() {

return (

<div>

<h1>Welcome to the first session of React</h1>

</div>

);

}

export default App;

**2.ReactJS-HOL**

**What Are React Components?**

React components are reusable UI blocks that help break the interface into small, manageable parts.  
They come in two main types:

Class Components

Function Components

**Components vs. JavaScript Functions**

| Feature | JavaScript Functions | React Components |
| --- | --- | --- |
| Purpose | Perform logic or calculations | Build UI (return JSX) |
| Return Value | Any JS data | JSX or null |
| Lifecycle Methods | Not available | Available in class components |
| Reusability in UI | Limited | Designed to be reusable UI pieces |
|  |  |  |
| **Types of Components** |  |  |

Class Component: Uses class syntax, supports state and lifecycle methods.

Function Component: Uses simple JavaScript functions, supports hooks like useState, useEffect.

**Class Component Example**

// Home.js

import React, { Component } from 'react';

class Home extends Component {

render() {

return <h1>Welcome to the Home page</h1>;

}

}

export default Home;

**Function Component Example**

// Function version of Home

function Home() {

return <h1>Welcome to the Home page</h1>;

}

**Constructor in Class Components**

constructor(props) {

super(props);

this.state = { name: "Tamu" };

}

**render() Method**

Every class component must have a render() method, which returns the JSX to be displayed.

Home.js

import React, { Component } from 'react';

class Home extends Component {

render() {

return (

<div>

<h2>Welcome to the Home page of Student Management Portal</h2>

</div>

);

}

}

export default Home;

**About.js**

import React, { Component } from 'react';

class About extends Component {

render() {

return (

<div>

<h2>Welcome to the About page of the Student Management Portal</h2>

</div>

);

}

}

export default About;

**Contact.js**

import React, { Component } from 'react';

class Contact extends Component {

render() {

return (

<div>

<h2>Welcome to the Contact page of the Student Management Portal</h2>

</div>

);

}

}

export default Contact;

**App.js**

import React from 'react';

import Home from './Components/Home';

import About from './Components/About';

import Contact from './Components/Contact';

function App() {

return (

<div className="App">

<Home />

<About />

<Contact />

</div>

);

}

export default App;

**3.ReactJS-HOL**

**What are React Components?**

React components are independent, reusable pieces of UI.  
They can manage their own state and accept props for dynamic behavior.  
Components come in two forms:

* Class Components
* Function Components

**Components vs JavaScript Functions**

| **Feature** | **JavaScript Functions** | **React Components** |
| --- | --- | --- |
| Purpose | Perform logic or return data | Render UI using JSX |
| JSX Return | Not designed for JSX | Must return JSX or null |
| Reusability in UI | Not reusable as UI | Designed for reusable UI blocks |
| State / Props Usage | Not applicable | Accepts props, supports state |

**Types of Components**

ClassComponentsUse ES6 class syntax and include lifecycle methods like componentDidMount().FunctionComponentsUse simple JS functions and can use Hooks like useState, useEffect.

**What is a Class Component?**

A class-based component extends React.Component and must have a render() method to return JS

**What is a Function Component?**

A function component is a JavaScript function that returns JSX directly. It's the modern and preferred approach in React.

**What is a Component Constructor?**

Used only in class components to:

* Initialize state
* Bind methods

Not needed in functional components.

**What is the render() Function?**

In class components, the render() method is required and returns JSX.  
Not needed in function components.

**CalculateScore.js**

import React from 'react';

import '../Stylesheets/mystyle.css';

function CalculateScore(props) {

const { name, school, total, goal } = props;

const average = total / goal;

return (

<div className="score-card">

<h2>Student Score Details</h2>

<p><strong>Name:</strong> {name}</p>

<p><strong>School:</strong> {school}</p>

<p><strong>Total Score:</strong> {total}</p>

<p><strong>Goal:</strong> {goal}</p>

<p><strong>Average Score:</strong> {average.toFixed(2)}</p>

</div>

);

}

export default CalculateScore;

**mystyle.css**

**.**score-card {

background-color: #f9f9f9;

border: 2px solid #4CAF50;

border-radius: 10px;

padding: 20px;

margin: 30px auto;

width: 60%;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

}

.score-card h2 {

color: #4CAF50;

text-align: center;

}

.score-card p {

font-size: 16px;

line-height: 1.6;

}

**App.js**

import React from 'react';

import './App.css';

import CalculateScore from './Components/CalculateScore';

function App() {

return (

<div className="App">

<CalculateScore name="Alice Smith" school="Greenwood High" total={450} goal={5} />

</div>

);

}

export default App;

**OUTPUT:**

Student Score Details

Name: Alice Smith

School: Greenwood High

Total Score: 450

Goal: 5

Average Score: 90.00

**4.ReactJS-HOL: Component Lifecycle, Data Fetching, and Error Handling**

**Why Are Component Lifecycles Important?**

React components go through various phases (creation, update, destruction). Understanding these lifecycles helps in:

Fetching data at the right time (componentDidMount)

Cleaning up before unmounting (componentWillUnmount)

Catching and handling errors gracefully (componentDidCatch)

Optimizing performance by controlling re-renders

**Lifecycle Methods Summary**

| Phase | Method | Purpose |
| --- | --- | --- |
| Mounting | constructor() | Initialize state and bind methods |
|  | render() | Render UI (JSX) |
|  | componentDidMount() | Fetch data after initial render |
| Updating | shouldComponentUpdate() | Control re-render logic |
|  | componentDidUpdate() | Act after updates (DOM or props/state) |
| Unmounting | componentWillUnmount() | Clean up (timers, listeners) before component removal |
| Error Handling | componentDidCatch() | Catch and log errors in child components |
|  |  |  |

**Lifecycle Execution Order**

constructor()

render()

DOM gets updated

componentDidMount()

On state/props change, render() again

Errors trigger componentDidCatch()

**Post.js**

import React from 'react';

function Post({ title, body }) {

return (

<div className="post">

<h3>{title}</h3>

<p>{body}</p>

</div>

);

}

export default Post;

**Posts.js**

import React, { Component } from 'react';

import Post from './Post';

class Posts extends Component {

constructor(props) {

super(props);

this.state = {

posts: [],

hasError: false

};

}

loadPosts = () => {

fetch('https://jsonplaceholder.typicode.com/posts')

.then(response => response.json())

.then(data => this.setState({ posts: data }))

.catch(error => {

console.error("Fetch Error:", error);

this.setState({ hasError: true });

});

};

componentDidMount() {

this.loadPosts();

}

componentDidCatch(error, info) {

alert("An error occurred in the Posts component.");

console.error("Error caught in component:", error, info);

}

render() {

const { posts, hasError } = this.state;

if (hasError) {

return <h2>Something went wrong. Please try again later.</h2>;

}

return (

<div>

<h2>Blog Posts</h2>

{posts.map(post => (

<Post key={post.id} title={post.title} body={post.body} />

))}

</div>

);

}

}

export default Posts;

**App.js**

import React from 'react';

import './App.css';

import Posts from './Posts';

function App() {

return (

<div className="App">

<Posts />

</div>

);

}

export default App;

**5.ReactJS-HOL React component with CSS Modules**

**What Does This Component Do?**

The CohortDetails component displays information about a training batch or learning group (called a "cohort") like:

Cohort name

Status (e.g., "Ongoing" or "Completed")

Start and End dates

The text color of the cohort title changes based on the status:

Green if ongoing

Blue if completed

**Output :**

<CohortDetails

name="React Bootcamp"

status="Ongoing"

startDate="2025-07-01"

endDate="2025-08-31"

/>

**CSS Styling from CohortDetails.module.css**

**.**box {

width: 300px;

display: inline-block;

margin: 10px;

padding: 10px 20px;

border: 1px solid black;

border-radius: 10px;

}

dt {

font-weight: 500;

}

**App.js**

import React from 'react';

import CohortDetails from './Components/CohortDetails';

function App() {

return (

<div className="App">

<CohortDetails

name="React Bootcamp"

status="Ongoing"

startDate="2025-07-01"

endDate="2025-08-31"

/>

<CohortDetails

name=".NET Masterclass"

status="Completed"

startDate="2025-05-01"

endDate="2025-06-15"

/>

</div>

);

}

export default App;