1. **ReactJS-HOL**

**1)Define SPA and its benefits**

A Single Page Application (SPA) is a web application that interacts with users by dynamically rewriting the current page instead of loading new pages from the server. It uses AJAX and HTML5 to create smooth and responsive web apps without constant page reloads.

Benefits of SPA: - Faster navigation after the initial load - Better user experience since it avoids full-page reloads - Reduced server load - Easier development of frontend logic using frameworks like React

**2)Define React and identify its working**

React is a JavaScript library developed by Meta (Facebook) for building fast and interactive user interfaces. It uses a component-based design, allowing developers to create self-contained components that manage their own state.

How React works:

- Uses a virtual DOM to efficiently update only the necessary parts of the UI

- Components are written with JSX (JavaScript + XML)

- React apps consist of reusable components

- Follows a one-way data flow (props → children)

**3)Identify the differences between SPA and MPA**

|  |  |  |
| --- | --- | --- |
| **Feature** | **SPA (Single-Page Application)** | **MPA (Multi-Page Application)** |
| **Page Reload** | No full reload, uses dynamic routing | Full page reload for every request |
| **Speed** | Faster after initial load | Slower due to repeated loading |
| **SEO** | More complex to optimize | Easier to optimize (server-rendered content) |
| **Technologies** | React, Angular, Vue | JSP, PHP, ASP.NET, HTML |
| **Examples** | Gmail, Facebook, Trello | Amazon, Wikipedia |

**4)Explain Pros & Cons of Single-Page Application**

Pros:

- Smooth user experience

- Faster load times after the initial load

- Efficient rendering with virtual DOM

- Can work like native apps

Cons:

- SEO optimization is more difficult

- Initial load time can be significant

- Requires JavaScript to be enabled

- More complex client-side routing and state management

**5)Explain about React**

React is a declarative, component-based JavaScript library used to build interactive user interfaces. It divides the UI into small, isolated, and reusable parts called components. Core concepts: - JSX for template-like syntax - Components (functional and class) - State and Props for data flow - Lifecycle methods - React Hooks (for functional components)

**6)Define virtual DOM**

Virtual DOM is an in-memory representation of the actual DOM. React creates a virtual DOM tree and updates it when the state or props change. React compares the new virtual DOM with the previous one using a diffing algorithm and applies the smallest set of changes to the real DOM. Benefits: - Faster updates - Efficient rendering - Avoids unnecessary DOM changes

**7)Explain Features of React**

- Component-Based: Build self-contained components that manage their own state

- Virtual DOM: Improves performance by reducing real DOM changes

- JSX: Combines JavaScript and HTML for clearer component definition

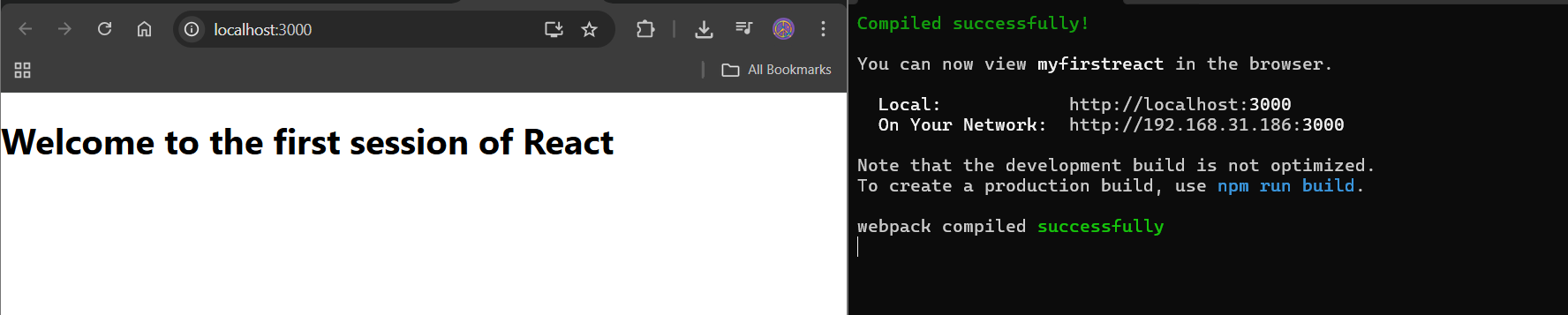
- One-Way Data Flow: Data flows from parent to child through props

- Hooks: Allow functional components to use state and lifecycle methods

- React Developer Tools: Helpful for debugging in the browser

- Unidirectional Binding: Helps keep data behavior predictable

**OUTPUT:**

****

1. **ReactJS-HOL**

**1)Explain React components**

React components are independent, reusable blocks of code that represent parts of a user interface. A component can be as simple as a button or as complex as an entire form. Each component can manage its own state and props; it can be rendered inside other components. React apps consist of many components working together to create a dynamic UI.

**2)Identify the differences between components and JavaScript functions**

|  |  |  |
| --- | --- | --- |
| **Feature** | **React Components** | **JavaScript Functions** |
| Purpose | Used to render UI in React | Used to perform tasks or return values |
| Return Type | Returns JSX (HTML-like structure) | Returns data (number, string, object, etc.) |
| Integration | Can be rendered in the DOM | Not directly rendered in the UI |
| React Specific APIs | Can use useState, useEffect, etc. | Cannot use React features unless it's a component |

**3)Identify the types of components**

**Class Components**

- Use ES6 classes

- Can hold state and lifecycle methods

- Syntax: class ComponentName extends React.Component

**Function Components**

- Use plain JavaScript functions

- Have a simpler and cleaner syntax

- Can use hooks (useState, useEffect) to handle state and lifecycle

**4)Explain class component**

A class component is a React component defined using an ES6 class that extends React.Component. It must include a render() method that returns JSX.

**5)Explain function component**

A function component is a simpler way to write a component using a plain JavaScript function that returns JSX. It’s ideal for UI pieces that don’t need lifecycle methods or complex state management.

**6)Define component constructor**

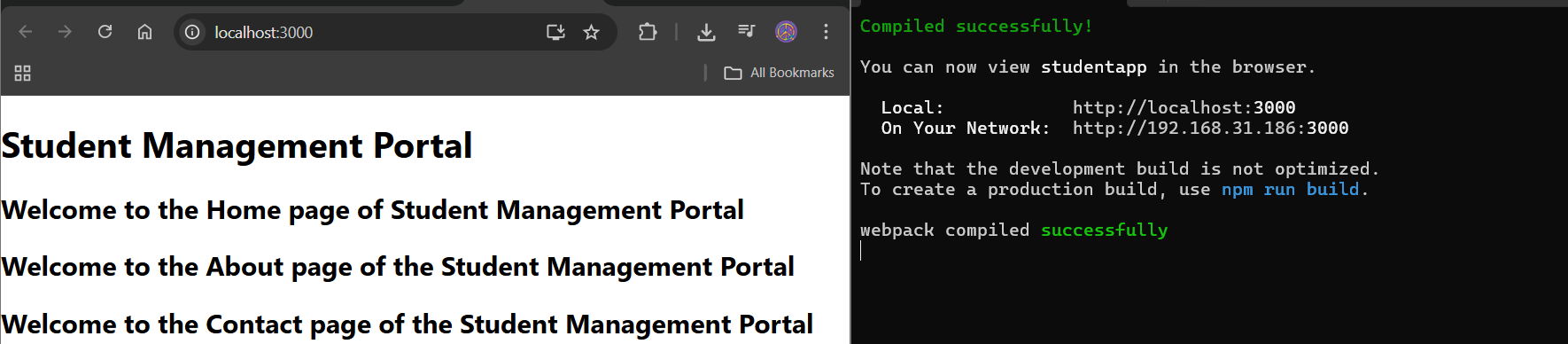
The constructor is a special function in class components that initializes state and binds methods.

**7)Define render() function**

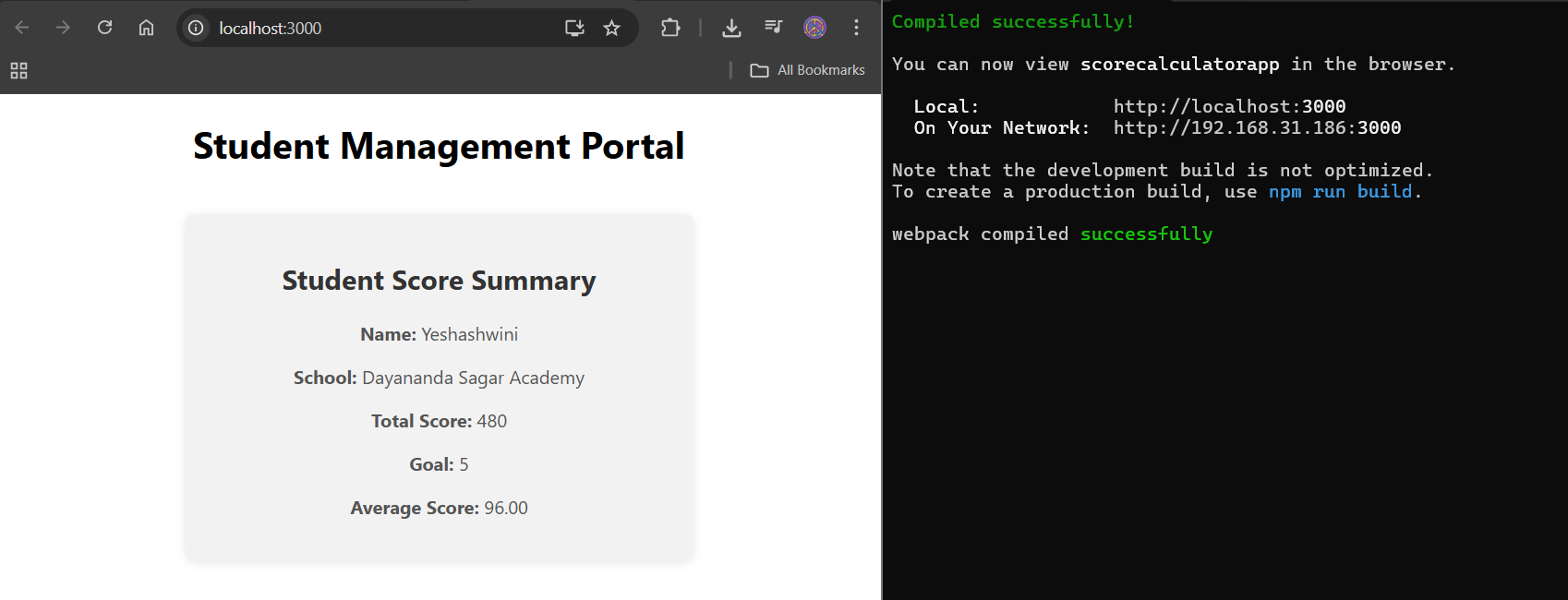
The render() method is required in class components. It returns the JSX to be displayed on the screen.

It is called every time the component needs to render or re-render.

**OUTPUT:**

****

1. **ReactJS-HOL**

****

1. **ReactJS-HOL**

**1)Explain the need and Benefits of component life cycle**

React components go through a lifecycle from creation to deletion. Understanding this lifecycle allows developers to:

* **Control behavior at specific stages** (e.g., when a component mounts, updates, or unmounts)
* **Fetch data at the right time** (e.g., after a component has rendered)
* **Improve performance** by minimizing unnecessary updates
* **Clean up resources** (like timers or event listeners)
* **Handle errors** gracefully with lifecycle hooks

**Benefits:**

* Efficient resource usage
* Better debugging and control
* Smooth data integration with APIs
* Error boundary handling

**2)Identify various life cycle hook methods**

Lifecycle methods are available in **class components**. They are grouped by phase:

**Mounting (when component is created and inserted into the DOM)**

* constructor()
* static getDerivedStateFromProps()
* render()
* componentDidMount()

**Updating (when props or state changes)**

* static getDerivedStateFromProps()
* shouldComponentUpdate()
* render()
* getSnapshotBeforeUpdate()
* componentDidUpdate()

**Unmounting (when component is removed from the DOM)**

* componentWillUnmount()

**Error Handling**

* componentDidCatch()
* static getDerivedStateFromError()

**3)List the sequence of steps in rendering a component**

When a component is first created (mounting phase), the following sequence of methods is called:

1. constructor()  
   → Used to initialize state and bind event handlers.
2. getDerivedStateFromProps() *(optional)*  
   → Syncs state with props if needed.
3. render()  
   → Returns the JSX to be rendered in the UI.
4. componentDidMount()  
   → Called once after the component is mounted to the DOM. Ideal for data fetching.

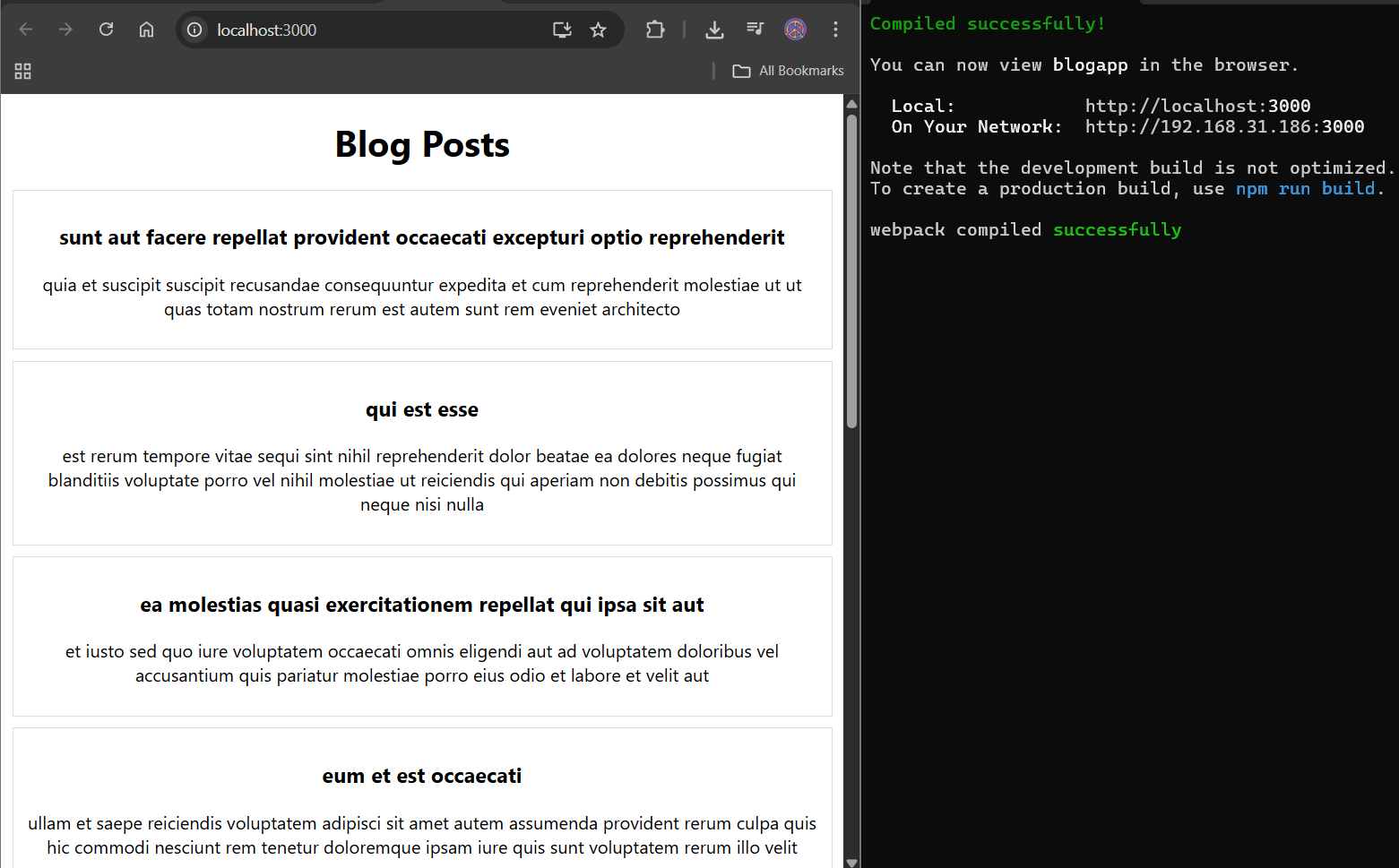
When the component **updates**, this sequence occurs:

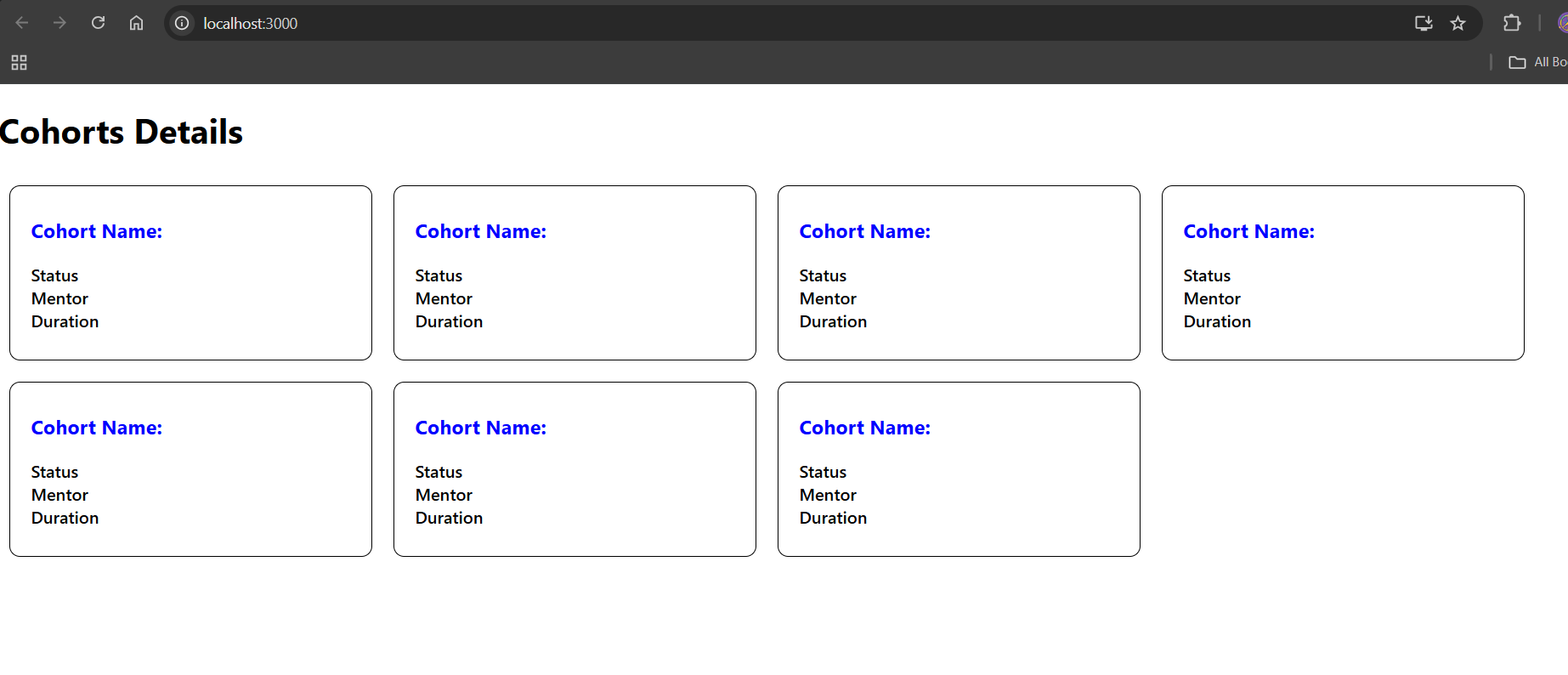
1. getDerivedStateFromProps()
2. shouldComponentUpdate() *(optional)*
3. render()
4. getSnapshotBeforeUpdate() *(optional)*
5. componentDidUpdate()

When the component **unmounts**:

* componentWillUnmount()  
  → Ideal for cleanup tasks like removing timers or event listeners.

**OUTPUT:**



1. **ReactJS-HOL  
     
   **