**COGNIZANT - DIGITALNURTURE4.0**

**DEEPSKILLING JAVA FSE**

**WEEK 06: REACT**

**Exercise 1: SPA**

**1. Define SPA and Its Benefits**

* **SPA (Single Page Application):** A web application that loads a single HTML page and dynamically updates content as the user interacts with the app.
* **Benefits:**
  + Faster interactions (no full page reloads)
  + Seamless user experience
  + Better performance after the initial load
  + Easy to convert into mobile apps (using React Native)

**2. Define React and Identify Its Working**

* **React:** A JavaScript library developed by Facebook for building user interfaces, especially for SPAs.
* **Working of React:**
  + Uses components for UI
  + Utilizes Virtual DOM for efficient updates
  + Follows unidirectional data flow
  + Re-renders only what’s necessary

**3.** **Differences Between SPA and MPA**

| **Feature** | **SPA (Single Page Application)** | **MPA (Multi Page Application)** |
| --- | --- | --- |
| Page Loading | Loads once; dynamic updates | Loads new page for each request |
| Speed | Fast after initial load | Slower due to full reloads |
| SEO | Harder to optimize | Better for SEO |
| Development | Typically more complex setup | Simpler HTML + backend |
| Examples | Gmail, Facebook | Amazon, Wikipedia |

**4. Pros & Cons of SPA**

Pros:

* Fast navigation
* Reusable components
* Smooth user experience

Cons:

* Initial load may be slow
* Poor SEO if not handled properly
* JavaScript must be enabled

**5. Explain About React**

React is a declarative, component-based library used to build dynamic and interactive user interfaces. It simplifies UI development by breaking it into reusable components.

**6. Define Virtual DOM**

* A lightweight copy of the actual DOM
* React compares Virtual DOM with the real DOM using a process called **reconciliation**
* Only the parts that change are updated in the real DOM, improving performance

**7. Features of React**

* JSX (JavaScript + XML)
* Component-Based Architecture
* Virtual DOM
* Unidirectional data flow
* Fast rendering and performance
* Support for hooks and functional components

**//App.js**

import React from 'react';

function App() {

  return (

    <div>

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

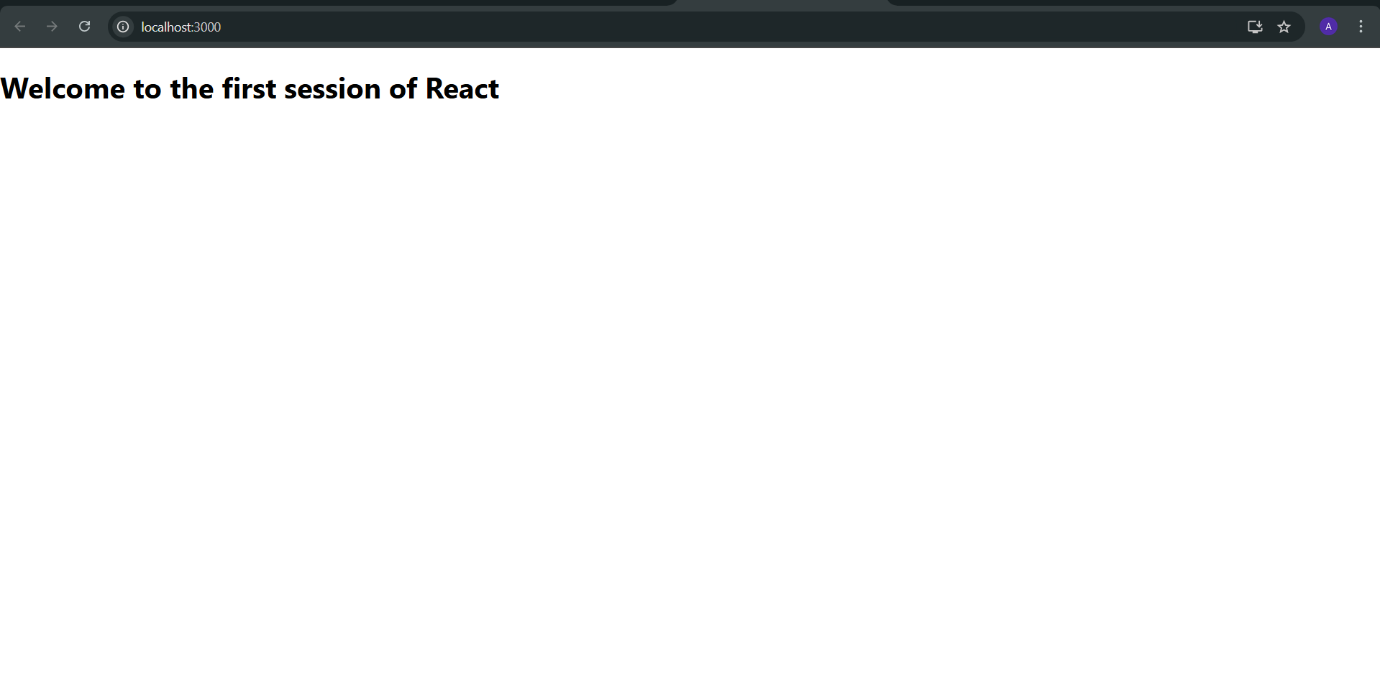
    </div>

  );

}

export default App;

**Output:**



**Exercise 2: React Components**

**1. Explain React Components**

* React Component: A reusable, self-contained piece of UI (like a button, form, or page section).
* Components can be either:
  + Function components
  + Class components

**2. Difference: Components vs JavaScript Functions**

| **Feature** | **React Component** | **JavaScript Function** |
| --- | --- | --- |
| Purpose | Used to return JSX/HTML | Executes logic |
| Reusability | Can maintain state & lifecycle | Just logic, no rendering |
| JSX Return | Yes | No (unless used in React) |
| Lifecycle Methods | Available in class components | Not available |

**3. Types of React Components**

* Class Component
* Function Component

**4. Class Component**

* Syntax: Uses ES6 class syntax.
* Can have state, lifecycle methods.

import React, { Component } from 'react';

class Home extends Component {

render() {

return <h2>Welcome to the Home page</h2>;

}

}

**5. Function Component**

* Simple function that returns JSX.

function About() {

return <h2>Welcome to the About page</h2>;

}

**6. Define Component Constructor**

* Special method used in class components to initialize state or bind methods.

constructor(props) {

super(props);

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

}

7. Define render() Function

* A required method in class components which returns the UI (JSX) for that component.

**//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;

**//Home.js**

import React, { Component } from 'react';

class Home extends Component {

render() {

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

}

}

export default Home;

**//About.js**

import React from 'react';

function About() {

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

}

export default About;

**//Contact.js**

import React from 'react';

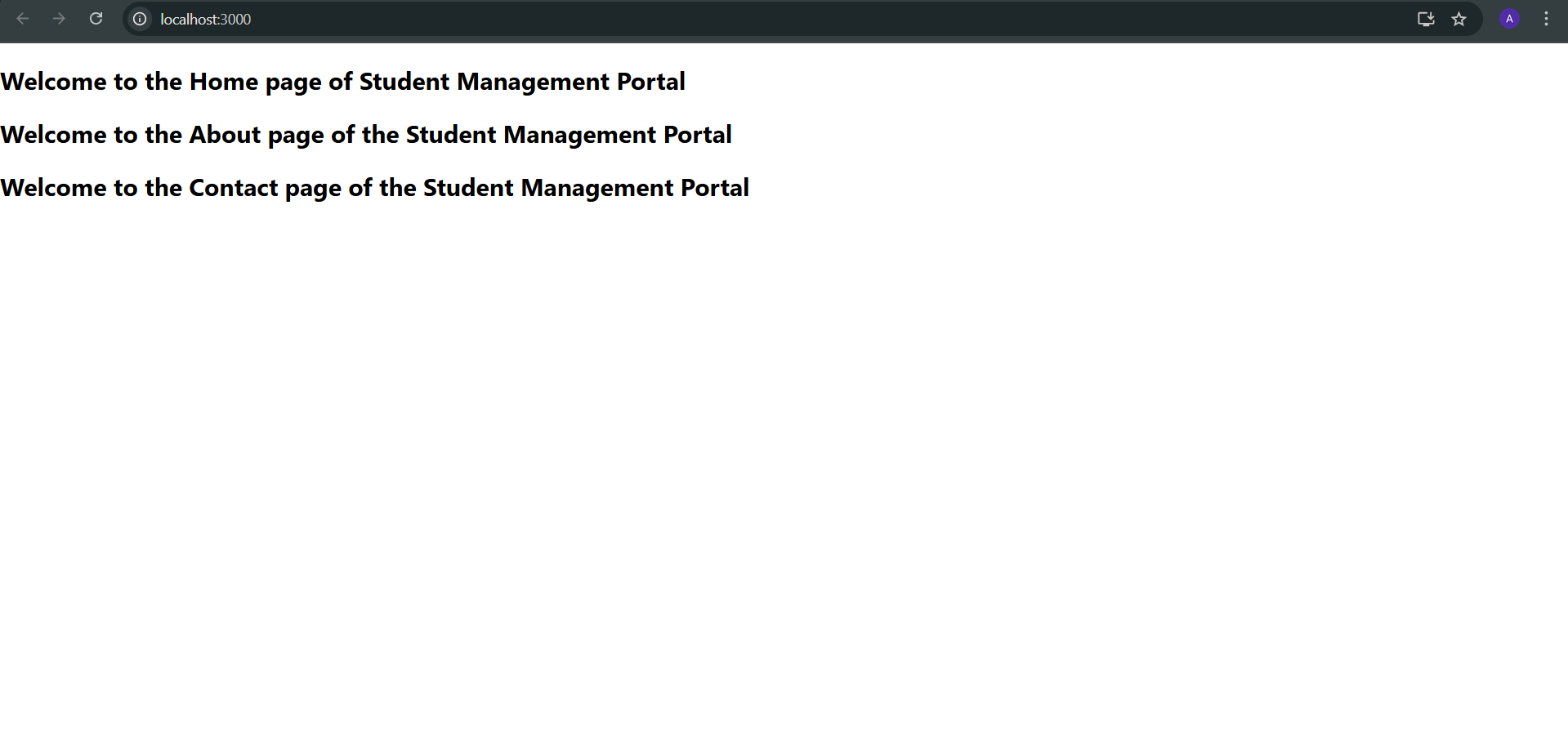
function Contact() {

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

}

export default Contact;

**Output:**

****

**Exercise 3: React Components**

**//App.js**

import React from 'react';

import './App.css';

import CalculateScore from './Components/CalculateScore';

function App() {

  return (

    <div className="App">

      <CalculateScore name="Ashritha" school="RMK Engineering College" total={450} goal={5} />

    </div>

  );

}

export default App;

**//mystyle.css**

.score-box {

  background-color: #f0f8ff;

  padding: 20px;

  margin: 30px auto;

  width: 50%;

  border-radius: 10px;

  box-shadow: 0 0 10px #ccc;

  font-family: Arial, sans-serif;

}

.score-box h2 {

  color: #2a52be;

  text-align: center;

}

.highlight {

  color: green;

  font-weight: bold;

}

**//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-box">

      <h2>Student Score Details</h2>

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

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

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

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

      <p className="highlight"><strong>Average Score:</strong> {average}</p>

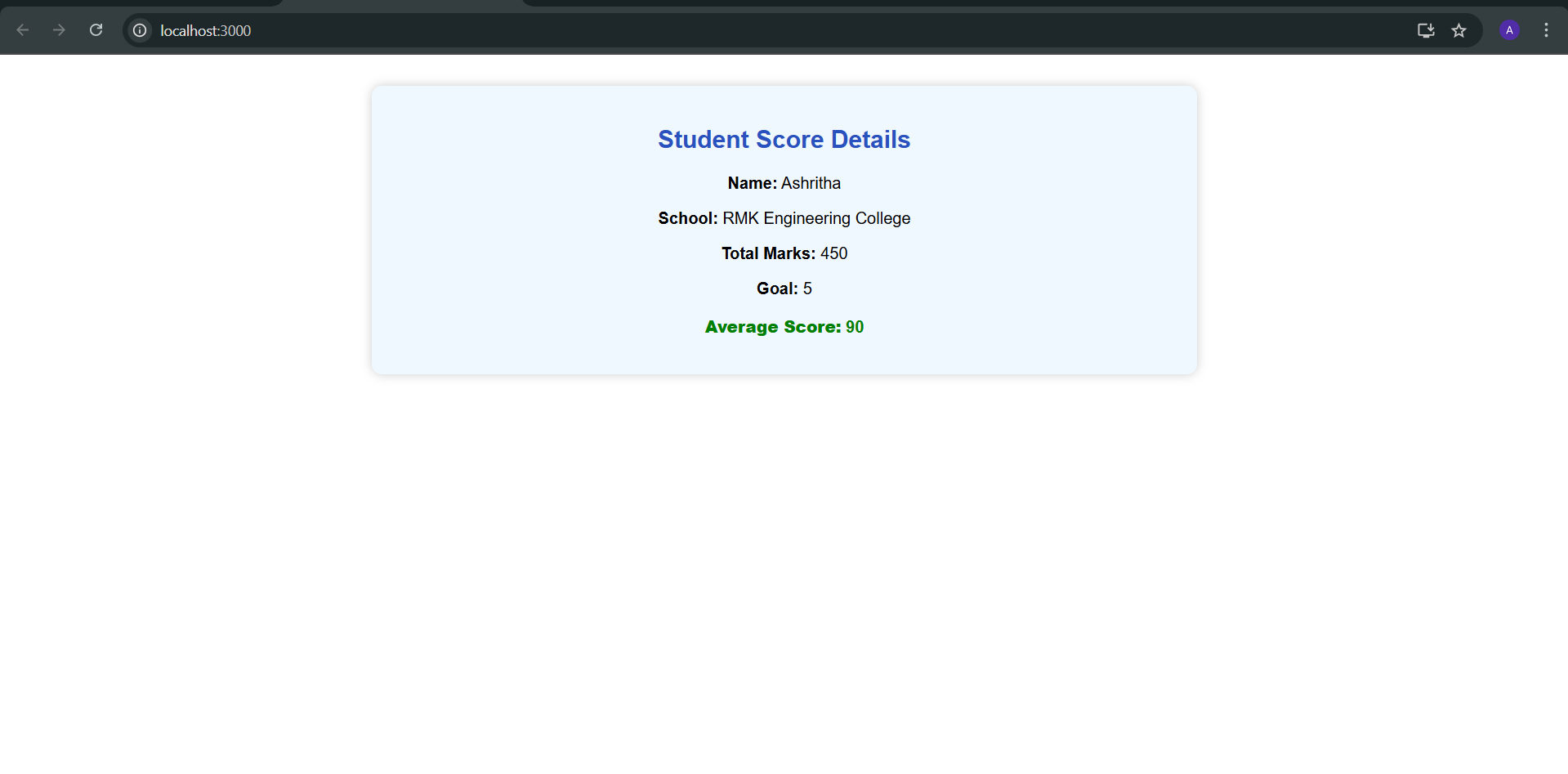
    </div>

  );

}

export default CalculateScore;

**Output**



**Exercise 4: Component Lifecycle**

**1. Explain the Need and Benefits of Component Lifecycle**

Lifecycle methods help in:

* + Managing state and side effects
  + Interacting with external APIs
  + Performing cleanup tasks
  + Handling errors gracefully

**2. Identify Various Lifecycle Hook Methods (Class Component)**

* Mounting: constructor, render, componentDidMount
* Updating: shouldComponentUpdate, componentDidUpdate
* Unmounting: componentWillUnmount
* Error Handling: componentDidCatch, getDerivedStateFromError

**3. Sequence of Rendering a Component**

1. constructor()
2. render()
3. componentDidMount() (after first render)
4. Updates → render() → componentDidUpdate()
5. Errors → componentDidCatch()

**//Post.js**

import React from 'react';

class Post extends React.Component {

render() {

const { title, body } = this.props;

return (

<div style={{ marginBottom: '20px' }}>

<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 = async () => {

try {

const response = await fetch('https://jsonplaceholder.typicode.com/posts');

const data = await response.json();

this.setState({ posts: data.slice(0, 5) }); // Limit to 5 posts

} 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:", error);

console.error("Info:", info);

this.setState({ hasError: true });

}

render() {

if (this.state.hasError) {

return <h2>Something went wrong while displaying posts.</h2>;

}

return (

<div>

<h2>Blog Posts</h2>

{this.state.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">

<h1>Welcome to BlogApp</h1>

<Posts />

</div>

);

}

export default App;

**Output:**



**Exercise 5: Cohort Dashboard Styling Lab**

**Key Concepts**

* Need for Styling React Components  
  Styling makes React components visually appealing, improves user experience, and distinguishes data clearly.
* CSS Modules  
  Local-scoped CSS styling, avoids class name collisions in large apps.
* Inline Styles  
  Define styles directly in the component using JavaScript objects.

**//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;

}

**//CohortDetails.js**

import React from 'react';

import styles from './CohortDetails.module.css'; // link CSS module

function CohortDetails({ cohort }) {

return (

<div className={styles.box}>

{/\* h3 title color depends on cohort.status \*/}

<h3 style={{ color: cohort.status === 'ongoing' ? 'green' : 'blue' }}>

{cohort.name}

</h3>

{/\* Description List: dt = term, dd = value \*/}

<dl>

<dt>Start Date:</dt>

<dd>{cohort.startDate}</dd>

<dt>End Date:</dt>

<dd>{cohort.endDate}</dd>

<dt>Status:</dt>

<dd>{cohort.status}</dd>

</dl>

</div>

);

}

export default CohortDetails;

**//App.js**

import React from 'react';

import './App.css';

import CohortDetails from './components/CohortDetails';

function App() {

const cohort1 = {

name: 'React Fundamentals',

startDate: '2024-01-01',

endDate: '2024-06-01',

status: 'ongoing'

};

const cohort2 = {

name: 'Node.js Bootcamp',

startDate: '2023-06-01',

endDate: '2023-12-01',

status: 'completed'

};

return (

<div className="App">

<CohortDetails cohort={cohort1} />

<CohortDetails cohort={cohort2} />

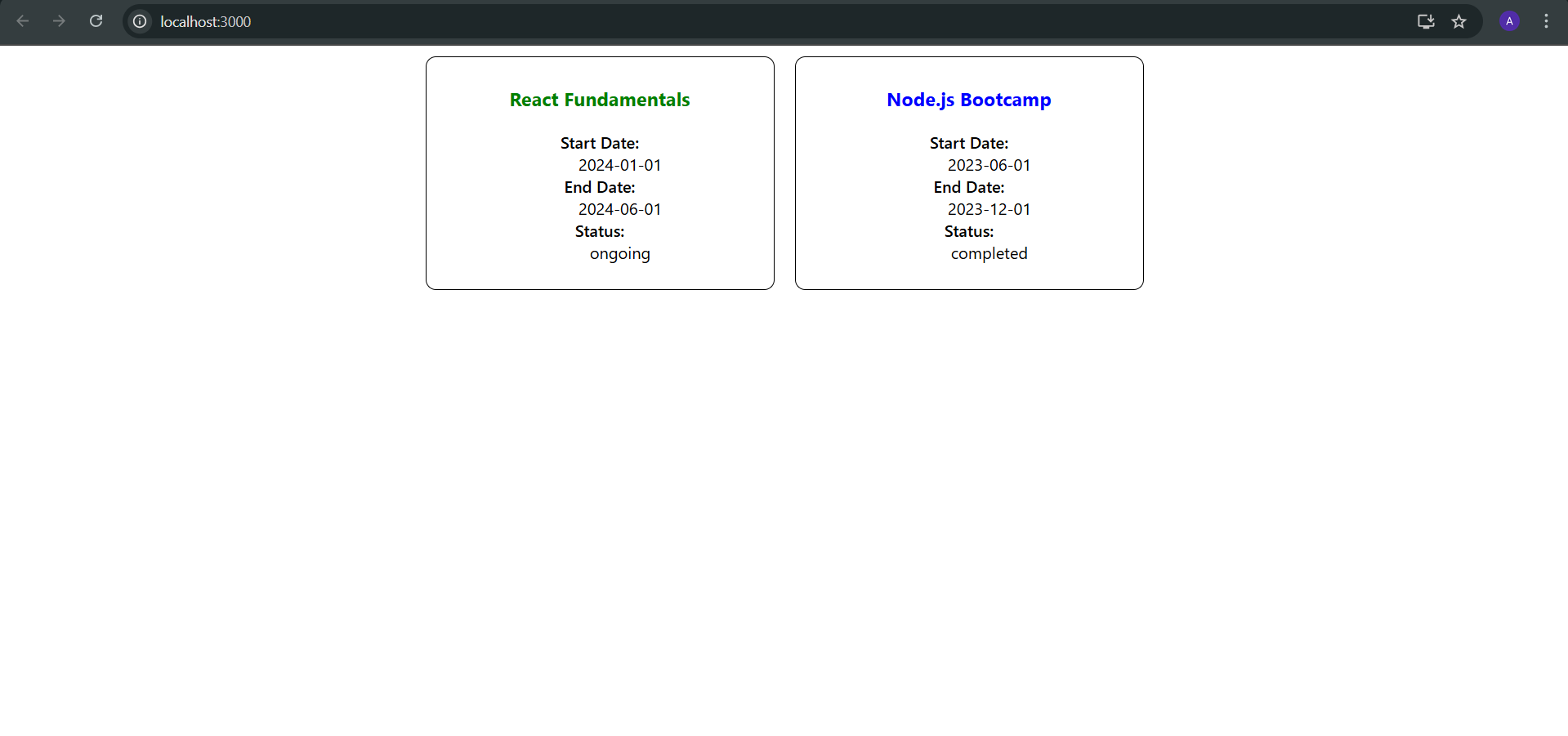
</div>

);

}

export default App;

**Output:**



**NAME: HARINI G  
SUPERSET ID: 6385037**