COGNIZANT DN - 4.0 DEEP SKILLING

HANDS ON WEEK-6

REACT

1. **ReactJS-HOL**
2. Define SPA and its benefits

**Answer:**

A **Single Page Application (SPA)** is a type of web application or website that **interacts with the user by dynamically rewriting the current page** rather than loading entire new pages from the server. This means that most of the content is loaded **once** during the initial page load, and further interactions happen **without full page refreshes.**

**Benefits:**

* Faster User Experience.
* Reduced Server Load.
* Enhanced Responsiveness.
* Better Caching Capabilities.

1. Define React and identify its working.

**Answer:**

**React** is an open-source **JavaScript library** developed by **Meta (formerly Facebook)** for building **user interfaces,** especially for **single-page applications (SPAs)**.  
It allows developers to create **reusable UI components** and manage the state of the application efficiently.

React is mainly concerned with the **View layer (V)** of the **MVC (Model-View-Controller)** architecture.

1. Identify the differences between SPA and MPA.

**Answer:**

**Single Page Application (SPA):** it is a web application that loads a single HTML page and dynamically updates the content as the user interacts with it, without reloading the entire page. It uses technologies like JavaScript, AJAX, and client-side routing (e.g., React Router) to provide a fast and fluid user experience similar to desktop or mobile apps. SPAs are well-suited for highly interactive applications like Gmail, Facebook, and Trello. However, SPAs can be complex to develop, require additional libraries for routing and state management, and are generally less SEO-friendly unless server-side rendering is used.

**Multi Page Application (MPA):** it loads a new HTML page from the server every time the user navigates to a different section. Each route is handled by the server, making MPAs simpler to develop and more naturally optimized for SEO, as search engines can easily index separate pages. MPAs are ideal for content-heavy websites like Amazon, Wikipedia, or blogs and university websites. However, they can be slower in performance due to full-page reloads on each interaction and offer a less seamless experience compared to SPAs.

1. Explain Pros & Cons of Single-Page Application.

**Answer:**

**Pros of SPA:**

1. **Fast and Seamless User Experience:**

* SPAs load content dynamically without reloading the page, providing a smooth, app-like experience.

1. **Reduced Server Load:**

* Only one initial page load is required; later interactions request only data (usually via APIs), reducing server bandwidth usage.

1. **Efficient Caching:**

* Once the app is loaded, SPAs can cache data and use local storage, minimizing repeated network calls.

1. **Code Reusability:**

* Components and logic can be reused easily across different parts of the app.

### ****Cons of SPA:****

1. **Poor SEO Out-of-the-Box:**

* Since content is loaded dynamically with JavaScript, search engines may struggle to index pages properly (without SSR or pre-rendering).

1. **Initial Load Time:**

* The first load can be slower as it includes the entire JavaScript framework and application logic.

1. **Browser Compatibility Issues:**

* Some older browsers may not fully support modern JavaScript features used in SPAs.

1. **Client-Side Complexity:**

* More logic is handled on the frontend, increasing development and debugging complexity.

1. Explain about React.

Answer:

**React** is a popular **open-source JavaScript library** used for building **user interfaces**, especially for **Single Page Applications (SPAs)**. It was developed by **Meta (Facebook)** and is widely used in modern web development to create fast, interactive, and scalable web applications. React focuses on the **"View" layer** of the **MVC (Model View Controller)** architecture, meaning it helps build the visual part of a web application what the user sees and interacts with.

1. Define virtual DOM.

Answer:

The **Virtual DOM (Document Object Model)** is a **lightweight, in-memory copy of the actual browser DOM** used by libraries like **React** to optimize performance and rendering.

1. Explain Features of React.

Answer:

Here are the features of the React:

### 1. ****Component-Based Architecture****

React allows you to build your UI using **independent, reusable components**. Each component manages its own state and logic, making development **modular and maintainable**.

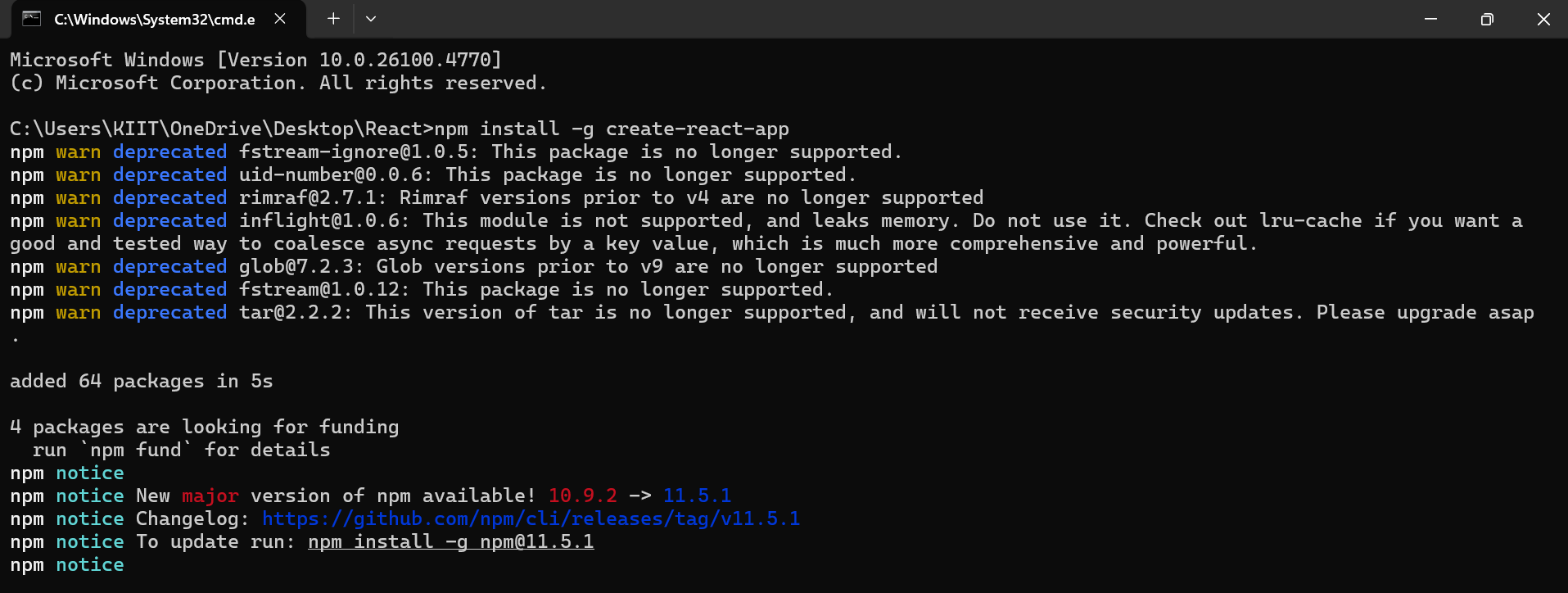
### 2. ****JSX (JavaScript XML)****

JSX is a **syntax extension** that lets you write HTML-like code inside JavaScript. It makes the structure of the UI easy to visualize and write, combining the power of JS and the simplicity of HTML.

### 3. ****Virtual DOM****

React uses a **Virtual DOM**, which is a lightweight copy of the real DOM. It updates the real DOM **efficiently** by calculating the **difference (diffing)** and applying **only the necessary changes**, improving performance.

1. Create a new React Application with the name “myfirstreact”, Run the application to print “welcome to the first session of React” as heading of that page.



**Code:**

*import* *React* *from* '*react*';

*function* App() {

*return* (

    <div>

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

    </div>

  );

}

*export* *default* App;



1. **ReactJS-HOL**
2. Explain React components.

**Answer:**

### ****Functional Components (Modern and Preferred)****

* These are simple JavaScript functions that return JSX.
* Can use **hooks** like useState, useEffect, etc., to manage state and side effects.

### ****Class Components (Older Style)****

* Defined using ES6 classes.
* Use this.state and lifecycle methods like componentDidMount().

1. Identify the differences between components and JavaScript functions.

**Answers:**

**React component:**  It is a special type of JavaScript function designed specifically to build and manage parts of a web application's user interface. It returns **JSX**, a syntax that looks like HTML and is used to describe what should appear on the screen. React components must start with a capital letter and are used like custom HTML tags, such as <Header /> or <UserCard />. They can maintain their own state, use hooks like useState or useEffect, and are an essential part of how React builds interactive, dynamic UIs.

**Regular JavaScript function**: it is a general-purpose block of code that performs specific tasks like calculations, data processing, or logic handling. These functions return standard JavaScript values such as numbers, strings, or objects, and are called using the function name followed by parentheses, for example add(2, 3). Unlike React components, regular functions do not interact with the UI or use JSX, and they do not have access to React-specific features like hooks or component lifecycle.

1. Define component constructor.

**Answer:**

In **class-based React components,** the **constructor** is a special method used to **initialize the component’s state** and **bind methods.**

It is the **first method called** when a component instance is created and is part of the component’s **lifecycle.**

1. Define render() function.

Answer:

The render() **function** is a **mandatory method** in **class-based React components.** It isused to **describe what should be displayed on the screen.**

This function returns **JSX**, which React converts into real DOM elements and inserts into the browser’s web page.

1. Create a react app for Student Management Portal named StudentApp and create a component named Home which will display the Message “Welcome to the Home page of Student Management Portal”. Create another component named About and display the Message “Welcome to the About page of the Student Management Portal”. Create a third component named Contact and display the Message “Welcome to the Contact page of the Student Management Portal”. Call all the three components.

**Code:**

**Home.js**

*import* *React* *from* '*react*';

*function* Home() {

*return* (

    <div>

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

    </div>

  );

}

*export* *default* Home;

**About.js**

*import* *React* *from* '*react*';

*function* About() {

*return* (

    <div>

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

    </div>

  );

}

*export* *default* About;

**Contact.js**

*import* *React* *from* '*react*';

*function* Contact() {

*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>

      <Home />

      <About />

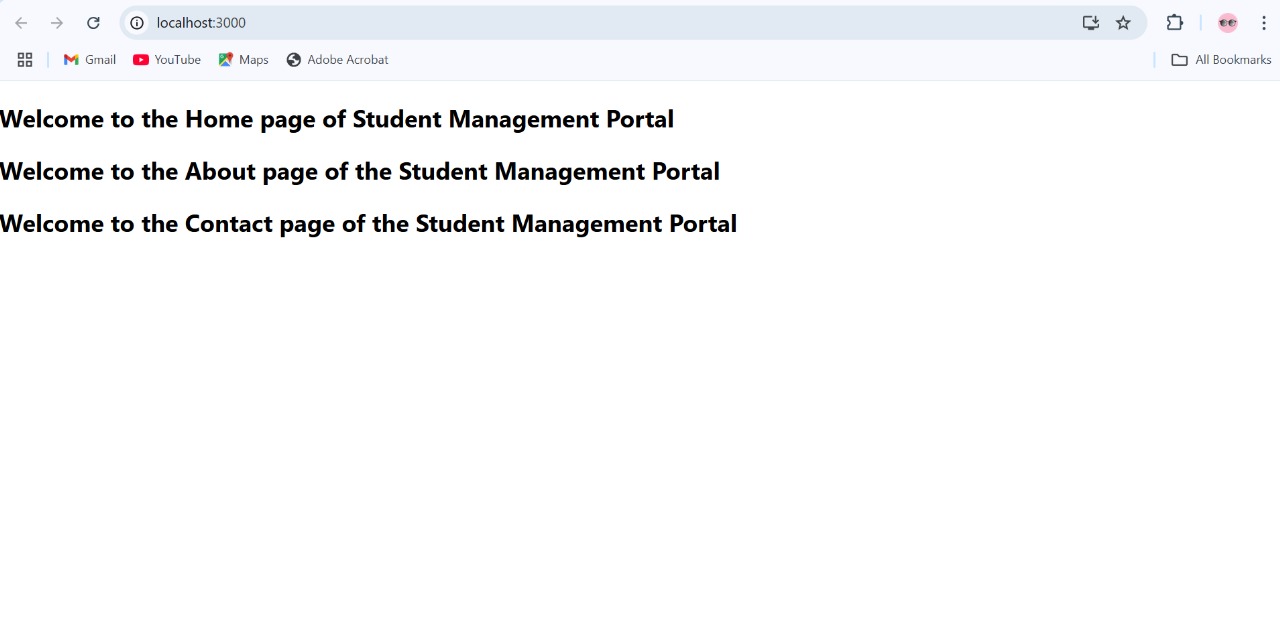
      <Contact />

    </div>

  );

}

*export* *default* App;

****

1. **ReactJS-HOL**
   1. Create a react app for Student Management Portal named scorecalculatorapp and create a function component named “CalculateScore” which will accept Name, School, Total and goal in order to calculate the average score of a student and display the same.

**Calculatescore.js**

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

*const* percentToDecimal = (*decimal*) *=>* {

*return* decimal.toFixed(2) + '*%*';

};

*const* calcScore = (*total*, *goal*) *=>* {

*return* percentToDecimal(total / goal);

};

*export* *const* CalculateScore = ({ *Name*, *School*, *total*, *goal* }) *=>* (

  <div *className*="*formatstyle*">

    <h1><font *color*="*Brown*">Student Details:</font></h1>

    <div *className*="*Name*">

      <b><span> Name: </span></b>

      <span>*{*Name*}*</span>

    </div>

    <div *className*="*School*">

      <b><span> School: </span></b>

      <span>*{*School*}*</span>

    </div>

    <div *className*="*Total*">

      <b><span> Total: </span></b>

      <span>*{*total*}*</span>

      <span>Marks</span>

    </div>

    <div *className*="*Score*">

      <b>Score:</b>

      <span>*{*calcScore(total, goal)*}*</span>

    </div>

  </div>

);

**MyStyle.css**

.*Name* {

*font-weight*: 300;

*color*: blue;

}

.*School* {

*color*: crimson;

}

.*Total* {

*color*: darkmagenta;

}

.*formatstyle* {

*text-align*: center;

*font-size*: large;

}

.*Score* {

*color*: forestgreen;

}

**App.js**

*import* { *CalculateScore* } *from* '*./Components/CalculateScore*';

*function* App() {

*return* (

    <div>

      <CalculateScore

*Name*="*Steeve*"

*School*="*DNV public school*"

*total*=*{*284*}*

*goal*=*{*3*}*

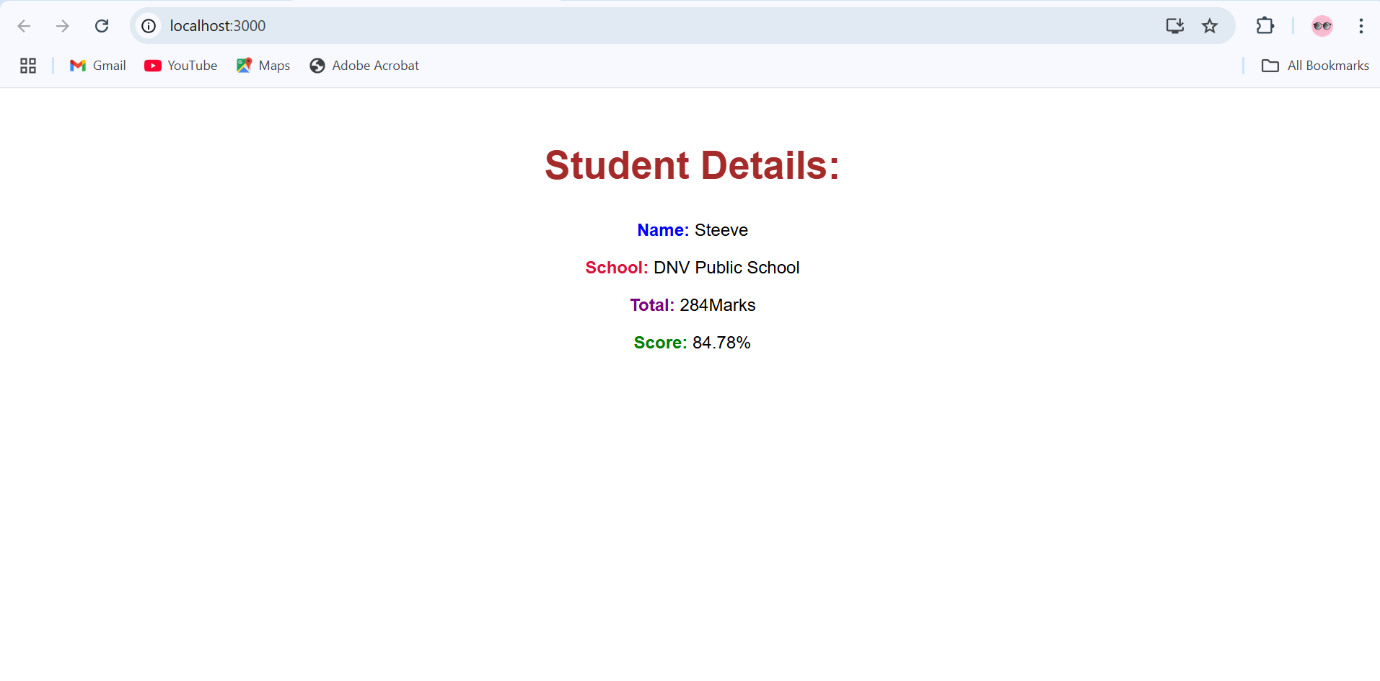
      />

    </div>

  );

}

*export* *default* App;

****

1. **ReactJS-HOL**
2. Explain the need and Benefits of component life cycle

**Answer:**

The component lifecycle refers to the sequence of phases a component goes through from its creation to its removal from the DOM. These phases include:

1. **Mounting** (component is created and inserted into the DOM)
2. **Updating** (component updates due to state/props changes)
3. **Unmounting** (component is removed from the DOM)
4. Identify various life cycle hook methods.

**Answer:**

In React, lifecycle hook methods are essential for managing the different phases a component goes through **mounting, updating,** and **unmounting**. In **class components**, several built-in lifecycle methods are available. During the mounting phase, methods like constructor(), getDerivedStateFromProps(), render(), and componentDidMount() are used to initialize the component, set up state, and fetch data. In the updating phase, React uses methods such as shouldComponentUpdate(), getSnapshotBeforeUpdate(), and componentDidUpdate() to respond to state or prop changes and optimize rendering. Finally, componentWillUnmount() is called before the component is removed from the DOM, commonly used to clean up timers, subscriptions, or listeners.

1. List the sequence of steps in rendering a component.

Answer:

### ****For Class Components:****

1. **constructor()**

* Called when the component is initialized.
* Used to set the initial state and bind methods.

1. **getDerivedStateFromProps()** (optional)

* Called before every render.
* Syncs state with changes in props.

1. **render()**

* **Required method** that returns JSX (UI to display).
* Pure and should not modify the state.

1. **componentDidMount()**

* Called **after the component is mounted** (rendered into the DOM).
* Ideal for API calls, subscriptions, or DOM manipulation.

**Lab Hands On:**

**Post.js**

*class* Post {

  constructor(*id*, *title*, *body*) {

*this*.id = id;

*this*.title = title;

*this*.body = body;

  }

}

*export* *default* Post;

**Posts.js**

*import* *React* *from* '*react*';

*import* *Post* *from* '*./Post*';

*class* Posts *extends* React.Component {

  constructor(*props*) {

*super*(props);

*this*.state = { *posts*: [], *loading*: *true*, *error*: *null* };

  }

  loadPosts() {

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

      .then(*res* *=>* {

*if* (!res.ok) *throw* *new* Error('*Network error*');

*return* res.json();

      })

      .then(*data* *=>* *this*.setState({

*posts*: data.map(*p* *=>* *new* Post(p.id, p.title, p.body)),

*loading*: *false*

      }))

      .catch(*err* *=>* *this*.setState({ *error*: err.message, *loading*: *false* }));

  }

  componentDidMount() {

*this*.loadPosts();

  }

  componentDidCatch(*error*, *info*) {

    console.error(error, info);

*this*.setState({ *error*: '*Something went wrong.*' });

  }

  render() {

    const { posts, loading, error } = *this*.state;

*if* (error) *return* <div>Error: *{*error*}*</div>;

*if* (loading) *return* <div>Loading...</div>;

*return* (

      <div>

*{*posts.map(*post* *=>* (

          <div *key*=*{*post.id*}*>

            <h2>*{*post.title*}*</h2>

            <p>*{*post.body*}*</p>

          </div>

        ))*}*

      </div>

    );

  }

}

*export* *default* Posts;

**App.js**

*import* *React* *from* '*react*';

*import* *Posts* *from* '*./Posts*';

*function* App() {

*return* (

    <div *className*="*App*">

      <h1>Blog Posts</h1>

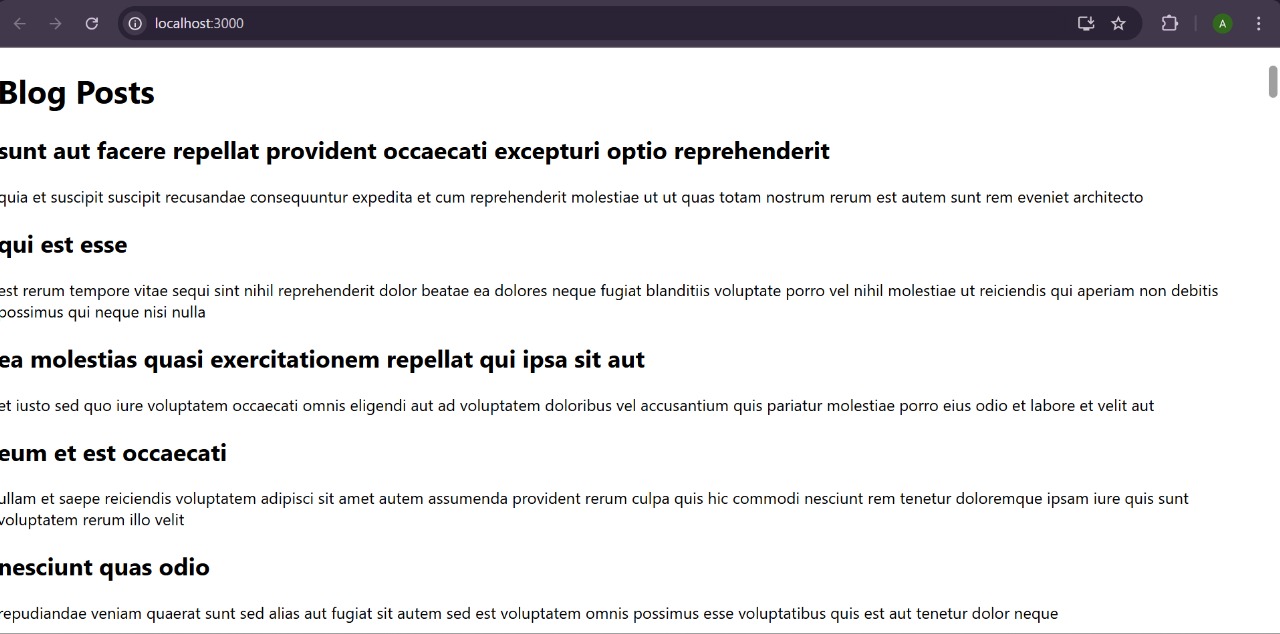
      <Posts />

    </div>

  );

}

*export* *default* App;

****

1. **ReactJS-HOL**

**CohortDetails.js**

*import* *React* *from* '*react*';

*import* *styles* *from* '*./CohortDetails.module.css*';

*function* CohortDetails(*props*) {

*const* { cohort } = props;

*return* (

    <div *className*=*{*styles.box*}*>

      <h3 *className*=*{*

        cohort.currentStatus === '*Ongoing*'

          ? styles.titleGreen

          : styles.titleBlue

*}*>

*{*cohort.cohortCode*}* - <span>*{*cohort.technology*}*</span>

      </h3>

      <dl>

        <dt *className*=*{*styles.dt*}*>Started On</dt>

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

        <dt *className*=*{*styles.dt*}*>Current Status</dt>

        <dd>*{*cohort.currentStatus*}*</dd>

        <dt *className*=*{*styles.dt*}*>Coach</dt>

        <dd>*{*cohort.coachName*}*</dd>

        <dt *className*=*{*styles.dt*}*>Trainer</dt>

        <dd>*{*cohort.trainerName*}*</dd>

      </dl>

    </div>

  );

}

*export* *default* CohortDetails;

**CohortDetails.module.css**

.*box* {

*width*: 300*px*;

*display*: inline-block;

*margin*: 10*px*;

*padding*: 10*px* 20*px*;

*border*: 1*px* solid black;

*border-radius*: 10*px*;

}

.*dt* {

*font-weight*: 500;

}

.*titleGreen* {

*color*: green;

}

.*titleBlue* {

*color*: blue;

}

**App.js**

*import* *logo* *from* '*./logo.svg*';

*import* '*./App.css*';

*import* { *CohortsData*} *from* '*./Cohort*'

*import* *CohortDetails* *from* '*./CohortDetails*';

*function* App() {

*return* (

  <div>

    <h1>Cohorts Details</h1>

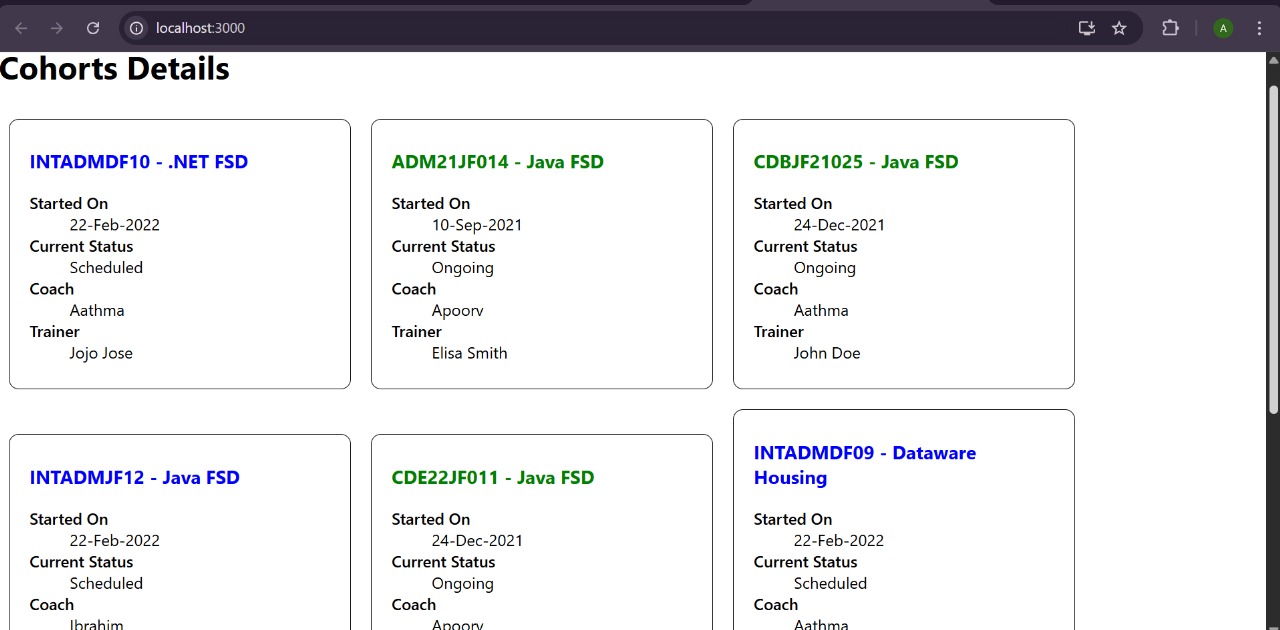
*{*CohortsData.map(*cohort* *=>* <CohortDetails *cohort*=*{*cohort*}*/>)*}*

  </div>

  );

}

*export* *default* App;

****