**DN 4.0 WEEK 6 MANDATORY PROBLEMS**

**MODULE 9: REACT JS**

1. **OBJECTIVES:**
   1. **Define SPA and its benefits**

SPA means Single Page Application that load a single HTML page and dynamically updates the entire page content without refreshing the whole again to see the change like in HTML

Benefits of SPA are Faster user experience and reduces server load.

* 1. **Define React and Identify its working**

React is a library developed using JavaScript where it is specially meant for developing user interfaces for single page applications.

Working:

Uses a virtual DOM to update only the necessary parts of the page

Manages the UI state using props and state

Uses components to break the UI inot reusable pieces

* 1. **Difference between SPA and MPA**

In SPA it loads only single HTML page and usually faster after initial load and handled by JavaScript.

In MPA it loads new HTML page with each request from the user and slower due to full page reloads and maximum it is handled by server.

* 1. **Pros and Cons of Single page Application**

Pros:

* Faster and interactive
* Better user experience
* Reduces server requests

Cons:

* Slower first load
* Browser history and back button need manual handling
* Security risks if not handled properly
  1. **Explain about React**

React is JavaScript Library which is used for building UI components and focuses on “What to show” rather than “How to show it”. It is fast due to it uses Virtual DOM

* 1. **Define Virtual DOM**

Virtual DOM is lightweight copy of the real DOM used by React

When the UI Changes then react updates the virtual DOM

Compares it with the previous version

Faster updates and better performance are the benefits

* 1. **Features of React**

Key features are:

* One way Data Binding
* Uses Virtual DOM
* Will have reusable components
* Easy integration with REST API’S

**PROGRAM:**

**APP.JS**

function FirstApp(){

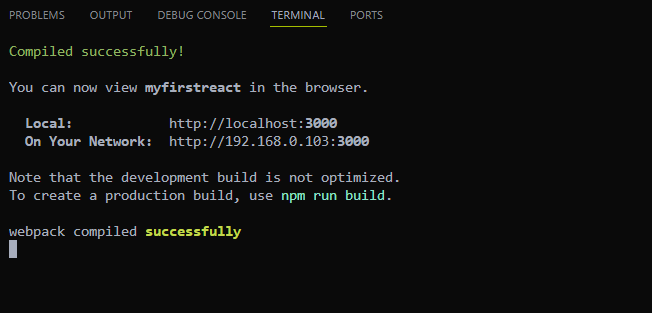
return(

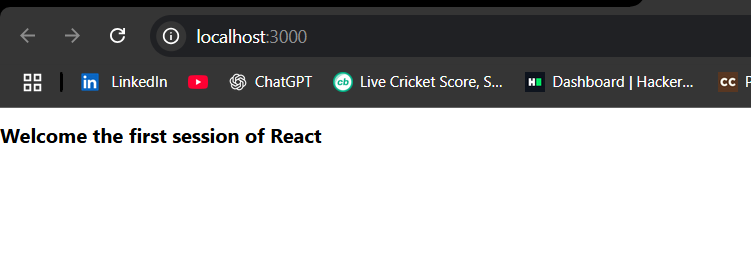
<h1>Welcome the first session of React</h1>

);

}

export default FirstApp;





1. **OBJECTIVES:**
   1. **EXPLAIN REACT COMPONENTS**

React Components are reusable building blocks of UI

Each component is like a small piece of the webpage like take a example like form, header, button, etc.

There are 2 main components:

Function Components: Use a simple function that returns JSX

Class Components: Use a class and must include a render() method.

* 1. **Difference between Components and JavaScript functions**

React components are used as building parts of a UI in web application this return HTML like code

JavaScript Functions are used to perform general tasks and return any data of any data type in JavaScript.

* 1. **Explain Class Component**

Import React from ‘react’;

Class Home extends React.Component{

Render(){

Return <h1>Welcome to the Home Page</h1>;

}

}

Here it should extend react.component and must have a render() method

* 1. **Explain Function Component**

Function Home(){

Return <h1>Welcome to the home page</h1>;

}

Simpler and no need to extend classes

* 1. **Define Component Constructor**

Constructor(props){

Super(props);

This.state={name:’Student’};

}

Here constructor() is used to initialize sate and bind event handlers

* 1. **Define render() function**

Render(){

Return <h1>hello</h1>;

}

Here render() tells react what to show on the screen

It returns a single element

**PROGRAM:**

**ABOUT.JS**

import React, { Component } from 'react';

class About extends Component {

  render() {

    return (

      <div>

        <h3>Welcome to the About page of the Student Management Portal</h3>

      </div>

    );

  }

}

export default About;

**CONTACT.JS**

import React, { Component } from 'react';

class Contact extends Component {

  render() {

    return (

      <div>

        <h3>Welcome to the Contact page of the Student Management Portal</h3>

      </div>

    );

  }

}

export default Contact;

**HOME.JS**

import React, { Component } from 'react';

class Home extends Component {

  render() {

    return (

      <div>

        <h3>Welcome to the Home Page of Student Management Portal</h3>

      </div>

    );

  }

}

export default Home;

After doing this we have to modify the app.js code also

**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 />

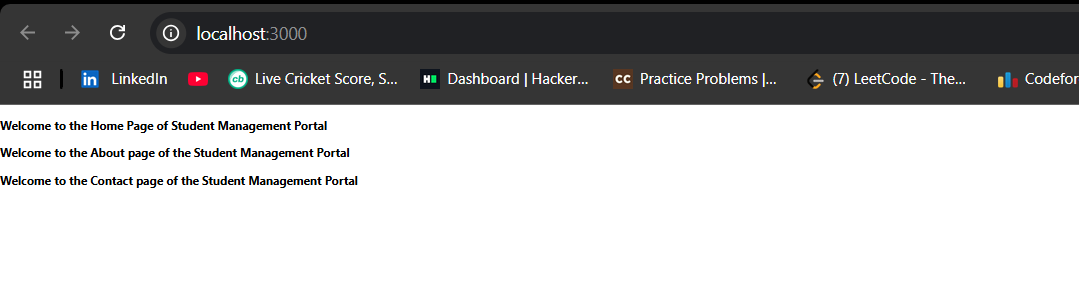
    </div>

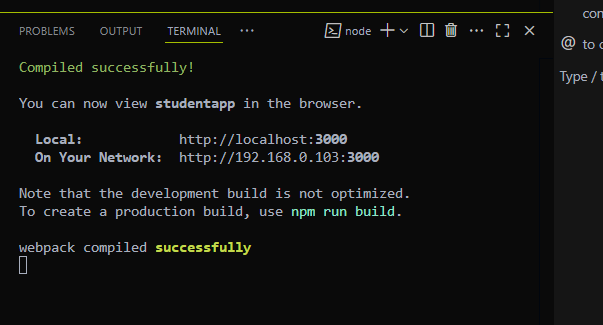
  );

}

export default App;

**OUTPUT:**

****



1. **OBJECTIVES**

**ALL THE OBJECTIVES DISCUSSED IN THE ABOVE QUESTIONS ARE DISCUSSED HERE AS WELL**

**PROGRAM:**

Following the same process that we did in the above code we need to create a separate folder named components and stylesheets and in components there is file name CalculateScore

**CalculateScore.js**

import React from 'react';

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

function CalculateScore(){

    const name="Suneel";

    const school="Gandhi Institute of Technolgy";

    const total=450;

    const goal=500;

    const average=(total/goal)\*100;

    return(

        <div className="score-box">

            <h2>Student Score Calculator</h2>

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

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

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

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

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

        </div>

        );

}

export default CalculateScore;

Now in the Stylesheets folder create a file with name mystyle.css

**Mystyle.css**

.score-box {

  padding:20px;

  margin:20px;

  border:2px *solid* #007bff;

  border-radius:10px;

  background-color:#f0f8ff;

  font-family:*Times* New Roman,*sans-serif*;

}

Now after doing this we need to modify the app,js and compile using npm start to get the output in localhost:3000

**App.js**

import React from 'react';

import CalculateScore from './Components/CalculateScore';

function App(){

  return(

    <div>

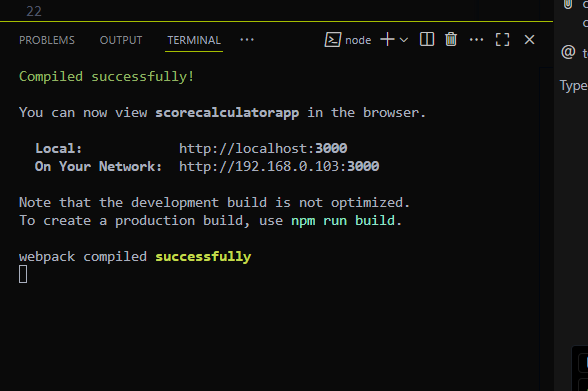
      <CalculateScore />

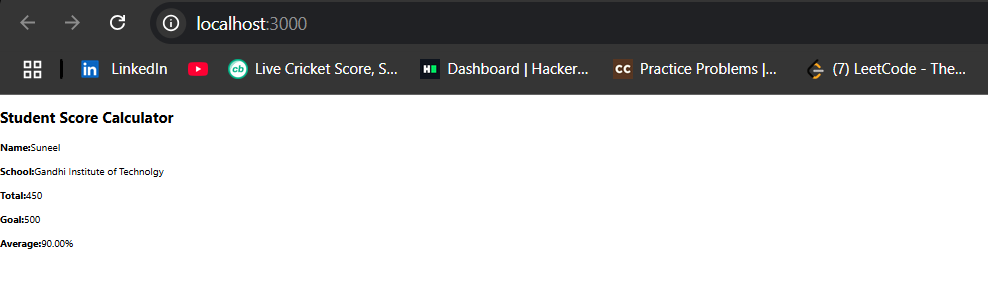
    </div>

  );

}

export default App;

****

****

1. **OBJECTIVES**
   1. **Explain the need and benefits of Component LifeCycle**

So, Component life cycle in react means that every class component goes through a series of phases from the moment it is created, updated and then removed from the DOM of the react this whole process is know as Component Life Cycle.

This process is needed

* To perform actions at specific points like fetch the data
* To clean up resources
* To optimize performance by controlling re renders.
* Helps manage side effects in a controlled way.
  1. **Identify Various Life Cycle Hook Methods**

Life cycle methods are available only in the class components

Mounting Phase

* Constructor()
* Static getDerivedStateFromProps()
* Render()
* componentDidMount()

Updating Phase

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

Unmounting Phase

* componentWillUnmount()
  1. **List the sequence of steps in rendering a component**

When a React class component is rendered it goes through a sequence of lifecycle steps.

Mounting Phase

* Constructor():- Initializes the component state and binds methods
* getDerivedStateFromProps():- Used to update the state based on incoming props.
* Render():- returns the html code to display on the screen.
* componentDidMount():- called after the component is rendered on the screen

Updating Phase

* render():- re renders the component with new data
* componentDidUpdate():- called after the update is completed

Unmounting Phase

* componentWillUnmount:- Used to clean up and prevent memory leaks.

**PROGRAM:**

Now here the goal is to show the blog post using sap and map

For this I created two files in the react app which is named blogapp

Those are post.js and posts.js

**Post.js**

import React from 'react';

function Post(props) {

  return (

    <div>

      <h3>{props.title}</h3>

      <p>{props.body}</p>

    </div>

  );

}

export default Post;

now posts.js

**Posts.js**

import React from 'react';

import {Post} from './Post';

class Posts extends React.Component{

    constructor(props){

*super*(props);

*this*.state={

            posts:[],

        };

    }

    loadPost(){

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

        .then(res=>res.json())

        .then(data=>{

            const postList=data.map(p=>new Post(p.id,p.title.p.body));

*this*.setState({posts:postList});

        })

        .catch(err=>{

            alert("Something is wrong");

            console.log(err);

        });

    }

    componentDidMount(){

*this*.loadPost();

    }

    render() {

    return (

        <div>

        <h1>Blog Posts</h1>

        {*this*.state.posts.map((item) => (

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

        ))}

        </div>

    );

    }

    componentDidCatch(error,info){

        alert("Error");

        console.log(error,info);

    }

}

export default Posts;

Here I created different components lifecycle from mounting to updating phase to finally unmounting phase

Now I updated the app.js as it needs to be updated according to how we want to display

**App.js**

import React from 'react';

import Posts from './Posts';

function App(){

  return(

    <div>

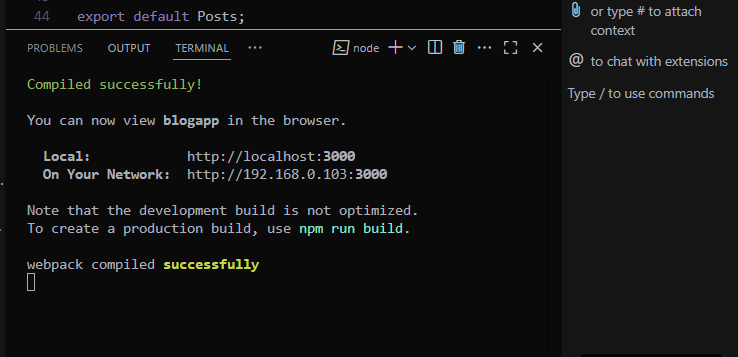
      <Posts />

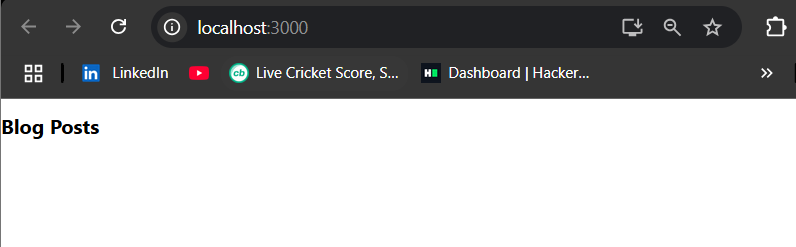
    </div>

  );

}

export default App;

****

****

1. **OBJECTIVES**
   1. **Understanding the need for styling react components**

As we know that React is used to build user interfaces so styling the look is what make those interfaces look good and user friendly and easy to navigate and responsive

So react is a component based its better to style components intependently so that user will feel easy to handle and each component will be self contained and easy to manage.

* 1. **Working with CSS module and inline styles**

CSS modules like in html it is also used for the same purpose in the react also like this allow you to write css and that is scoped to one component.

We will use the same conventions like we used for styling html but in react we import like this

Import styles from “ that address”

This way we can avoid name conflicts and keeps styling clean and organized

When coming to Inline styles we give the styling in the react code itself

Example

Function mycomponent(){

Const style={color:’green’};

Return <h1 style={style}> hello</h1>

}

This gives the output like the hello will printed in green color

In inline reusability will be very less and it will be local to that tag or elment.

**PROGRAM:**

After extracting the folder given in the word document of React 5

Then use npm install this will install the react.modules

Now change the given codes accordingly and create a cohortDetails.module.css

**CohortDetails.Module.css**

.box {

  width: 300px;

  display: *inline-block*;

  margin: 10px;

  padding: 10px 20px;

  border: 1px *solid* *black*;

  border-radius: 10px;

}

Change the app.js according to the cohort

**App.js**

import './App.css';

import { CohortsData } from './Cohort';

import CohortDetails from './CohortDetails';

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

function App() {

  return (

    <div className="App">

      <h1>Cohorts Details</h1>

      <div className="cohort-container">

        {CohortsData.map((co, i) => (

          <CohortDetails key={i} cohort={co} />

        ))}

      </div>

    </div>

  );

}

export default App;

**CohortDetails.js**

This is already given the extracted folder

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

function CohortDetails(props) {

  let col = "blue";

  if(props.cohort.currentStatus === "Ongoing") {

    col = "green";

  }

  return (

    <div className={styles.box}>

      <h3 style={{ color: col }}>

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

      </h3>

      <dl>

        <dt>Started On</dt>

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

        <dt>Current Status</dt>

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

        <dt>Coach</dt>

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

        <dt>Trainer</dt>

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

      </dl>

    </div>

  );

}

export default CohortDetails;

**Cohort.js**

**Same goes with this also this is also given in the extracted folder**

class Cohort {

    constructor(cohortCode,

        startDate,

        technology,

        trainerName,

        coachName,

        currentStatus) {

*this*.cohortCode = cohortCode;

*this*.coachName = coachName;

*this*.trainerName = trainerName;

*this*.technology = technology;

*this*.startDate = startDate;

*this*.currentStatus = currentStatus;

    }

}

const CohortsData =[

    new Cohort('INTADMDF10','22-Feb-2022', '.NET FSD', 'Jojo Jose','Aathma', 'Scheduled'),

    new Cohort('ADM21JF014','10-Sep-2021', 'Java FSD', 'Elisa Smith','Apoorv', 'Ongoing'),

    new Cohort('CDBJF21025','24-Dec-2021', 'Java FSD', 'John Doe','Aathma', 'Ongoing'),

    new Cohort('INTADMJF12','22-Feb-2022', 'Java FSD', 'To Be Assigned','Ibrahim', 'Scheduled'),

    new Cohort('CDE22JF011','24-Dec-2021', 'Java FSD', 'Emma Swan','Apoorv', 'Ongoing'),

    new Cohort('INTADMDF09','22-Feb-2022', 'Dataware Housing', 'Babjee Rao','Aathma', 'Scheduled'),

    new Cohort('ADM22DF001','10-Sep-2021', '.NET FSD', 'Marie Curie','Ibrahim', 'Ongoing'),

];

export {Cohort, CohortsData};

**App.css**

**Same with this small part is appended in the last of this text other all are already given in the folder which I extracted**

.App {

  text-align: *center*;

}

.App-logo {

  height: 40vmin;

  pointer-events: *none*;

}

@media (prefers-reduced-motion: no-preference) {

  .App-logo {

    animation: App-logo-spin *infinite* 20s *linear*;

  }

}

.App-header {

  background-color: #282c34;

  min-height: 100vh;

  display: *flex*;

  flex-direction: *column*;

  align-items: *center*;

  justify-content: *center*;

  font-size: calc(10px + 2vmin);

  color: *white*;

}

.App-link {

  color: #61dafb;

}

@keyframes App-logo-spin {

  from {

    transform: rotate(0deg);

  }

  to {

    transform: rotate(360deg);

  }

}

**.cohort-container {**

**display: *flex*;**

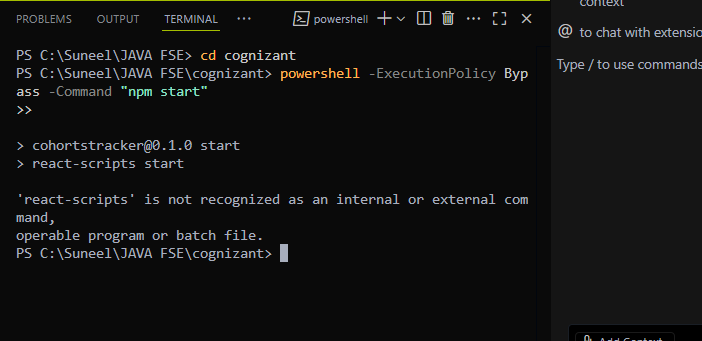
**flex-wrap: *wrap*;**

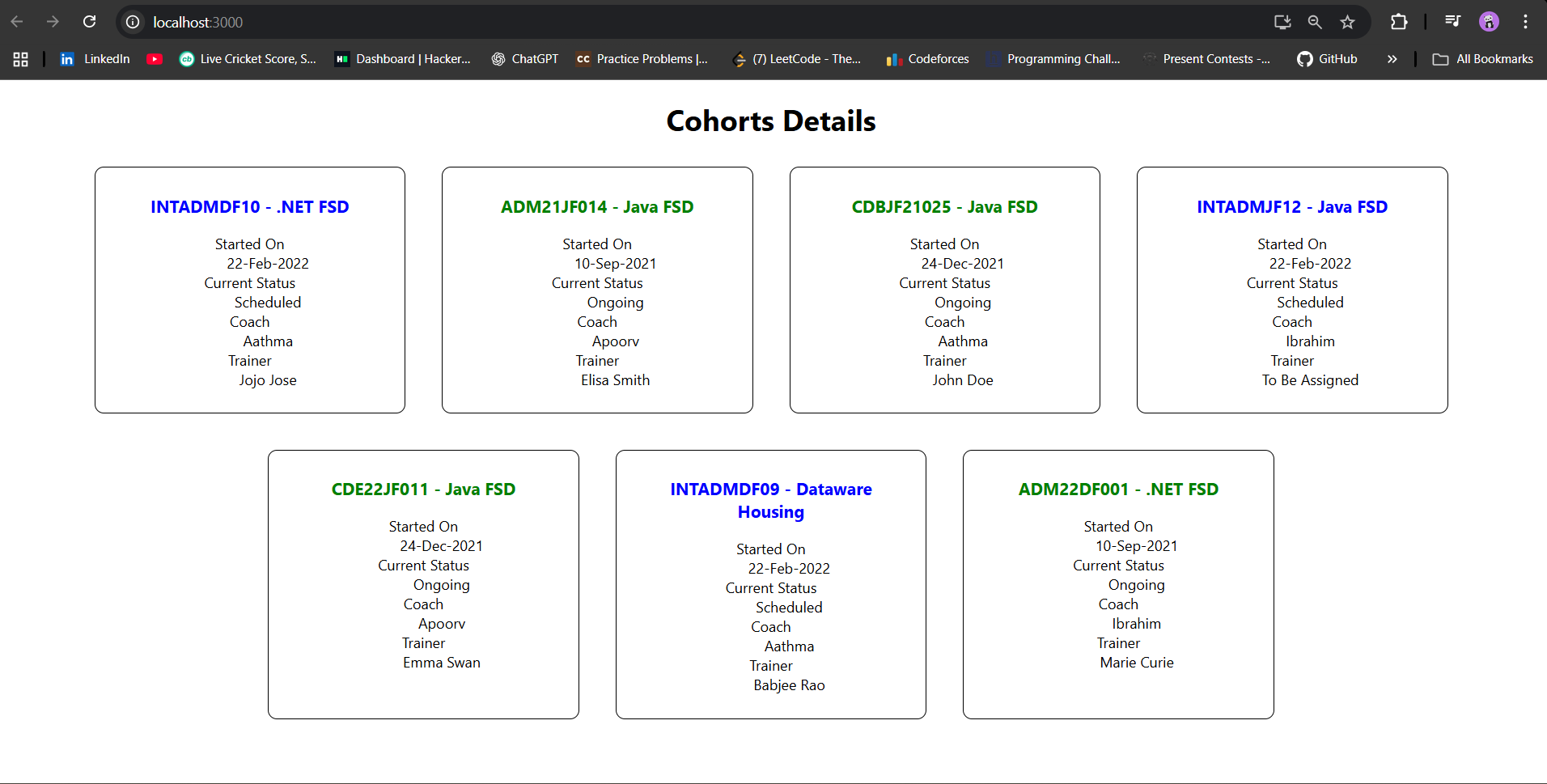
**justify-content: *center*;**

**gap: 20px;**

**margin-top: 20px;**

**}**

****

****