**Hands On for React week 6**

**HandsOn 1:**

**Define SPA and its benefits**

**=**

Instead of loading whole new pages from the server, a Single-Page Application (SPA) dynamically rewrites the content of a single webpage. This indicates that only the required content is updated using JavaScript and that the page does not reload while being used.

Advantages of SPA:

Fast Performance: By updating only a portion of the page, load times are decreased.

Smooth User Experience: Because there are no full page reloads, it feels more like a native app.

Effective Data Handling: Uses APIs (usually REST or GraphQL) to communicate with the server.

Component-based development, like that found in React, is encouraged by modular development.

**· Define React and identify its working**

**=** React is a javascript library. React is used to create user interfaces, mostly for SPAs. React, created by Facebook, enables programmers to create reusable user interface elements that are capable of self-management.

How React Operates:

It divides the user interface into manageable chunks using components.

It uses a virtual DOM to effectively track modifications.

React compares changes in the virtual DOM and applies updates selectively, updating only the necessary portion of the DOM when a component's state or props change.

**· Identify the differences between SPA and MPA**

**=**

Spa is single page application that is it consists only one html page but Mpa has multiple html pages. Spa gives better user experience with no scope of reload but Mpa comparatively give lesser user experience. The speed of Spa in more compared to Mpa .

**· Explain Pros & Cons of Single-Page Application**

**=**

**Pros:**

Faster user experience after initial load

Efficient and interactive UI

Frontend and backend can be separated

Reusable and testable components

**Cons:**

Poor SEO without extra setup (e.g., SSR)

Initial load can be large and slow

JavaScript must work properly for app to function

Harder browser history and analytics tracking

**· Explain about React**

**=**

React is a declarative, component-based JavaScript library for building interactive UIs which was developed by Facebook and it uses a virtual DOM for efficient updates. It Works with tools like React Router, Redux, and Next.js for routing, state management, and server-side rendering.

**· Define virtual DOM**

**=**

React uses a lightweight, in-memory version of the real DOM called Virtual DOM (VDOM).When rendering components, React generates a virtual DOM tree.

When state or property changes, React makes a brand-new virtual DOM tree, uses a diffing algorithm to compare it to the prior one. It only modifies the modified portions of the actual DOM. By lowering the quantity of direct DOM manipulations, this enhances performance.

**· Explain Features of React**

**=**

React JSX (JavaScript XML) features include; a JavaScript syntax extension that enables the writing of HTML-like code.

Architecture Based on Components: UI is separated into components that can be reused.

Virtual DOM: improves performance while requiring fewer DOM changes.

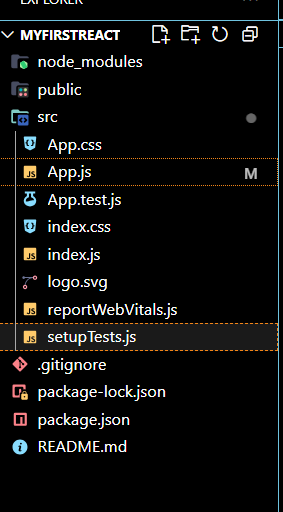
Props are used to transfer data from parent to child.

One-way Flow: improves control and debugging.

Declarative user interface: React modifies the DOM in accordance with developers' declarations of the UI state.

React Hooks: UseState, useEffect, and other hooks allow functional components to manage state and lifecycle.

**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.**

**=**

**React app created . These are the files included.**

**App.js:**

import React from 'react';

function App() {

  return (

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

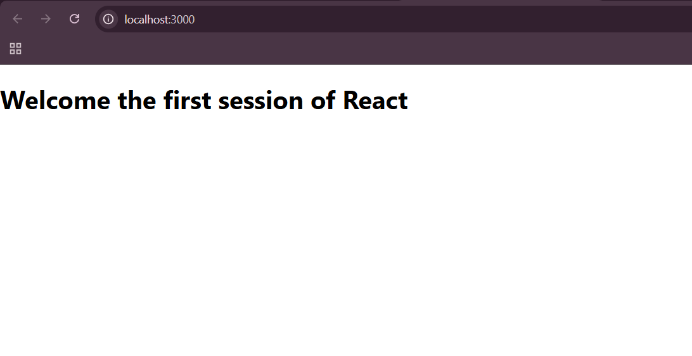
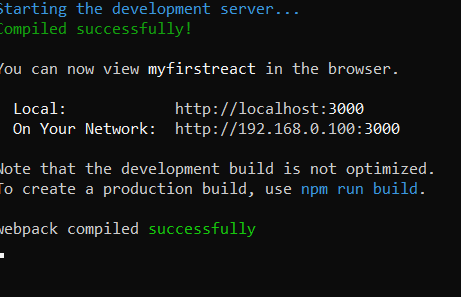
  );

}

export default App;

**Command Run**: npm start

**Output:**

****

**Hands On 2:**

* **Explain React components**

**=** React components are the building blocks of a React application.  
They are reusable pieces of UI that can accept input props and manage their own internal state.

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

**=**

React components differ from regular JavaScript functions in purpose and behavior. While JavaScript functions are used for performing logic or calculations, React components are specifically designed to build and manage parts of the user interface. Components return JSX can manage internal state, accept props, and respond to user interactions. They also support lifecycle methods and work with the React rendering . But, regular functions don't handle UI updates or have access to React's reactivity features.

* **Identify the types of components**

**=**

There are mainly two types: Class components and Function components. Class components use ES6 class syntax and can use lifecycle methods and state. But, Function components are simpler and use hooks to do the same things in a more modern way.

* **Explain class component**

**=** A class component is written using a JavaScript class that extends React.Component. It has a special method called render() which tells React what to show on the screen. We can also use this.state to store data and this.props to get input values from parent components.

* **Explain function component**

**=** A simple JavaScript function that returns JSX is called a function component. Compared to a class component, it is shorter and easier. It can also handle data and side effects with the aid of React Hooks like useState. Function components are used more frequently these days.

* **Define component constructor**

**=**

The constructor is a unique function that is used to initialize class components. If necessary, it binds methods and establishes the initial state. To ensure that the component inherits from React correctly, you always call super(props) first.

* **Define render() function**

**=**

The render() function is responsible for returning the JSX that React should display. It is found inside class components . It's called automatically whenever the component needs to show or update the UI.

**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.**

**=**

**Home.js:**

import React from 'react';

class Home extends React.Component {

render() {

return (

<div>

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

</div>

);

}

}

export default Home;

**About.js :**

import React from 'react';

class About extends React.Component {

  render() {

    return (

      <div>

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

      </div>

    );

  }

}

export default About;

**Contact.js :**

import React from 'react';

class Contact extends React.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 './App.css';

import Home from './Components/Home';

import About from './Components/About';

import Contact from './Components/Contact';

function App() {

  return (

    <div className="App">

      <Home />

      <About />

      <Contact />

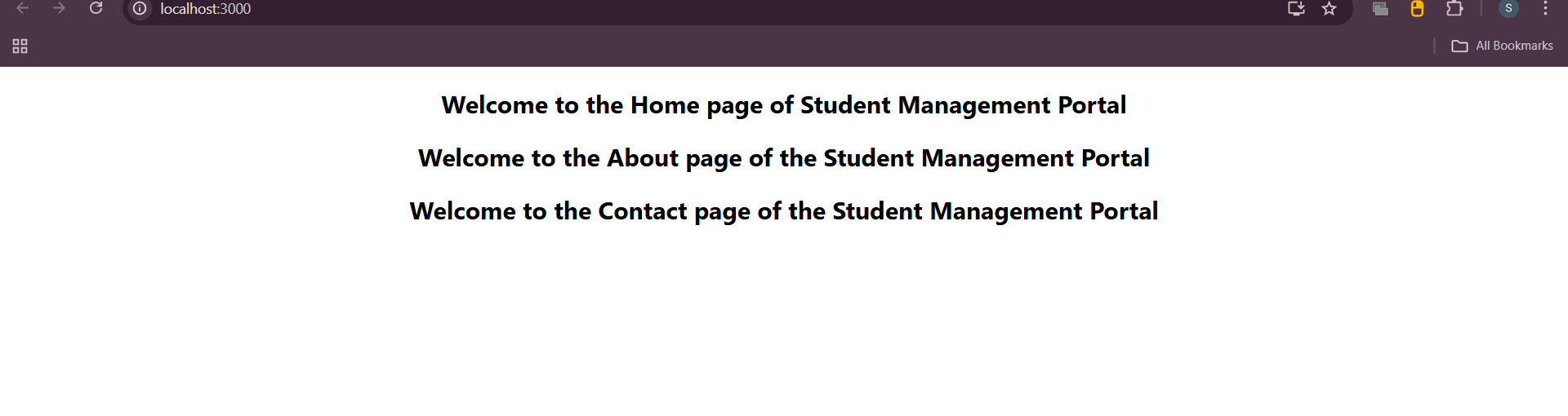
    </div>

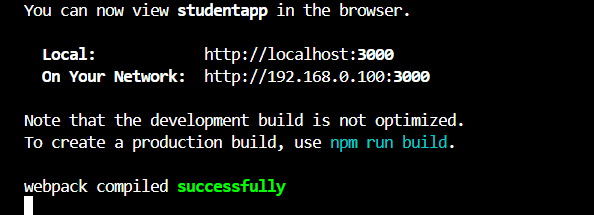
  );

}

export default App;

**Outpt:**

****

****

**PTO**

**HANDS ON 3**

**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}

      />

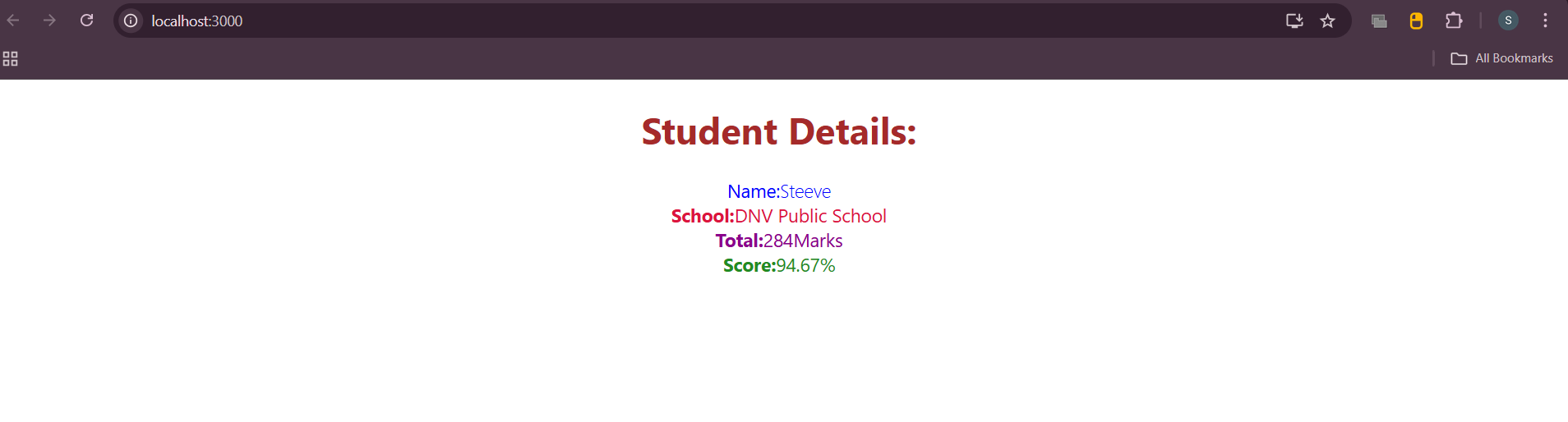
    </div>

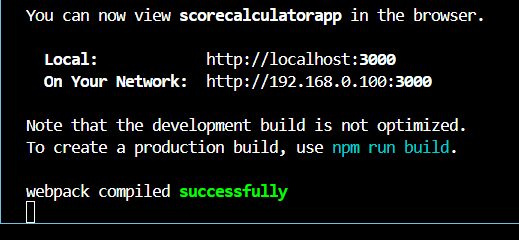
  );

}

export default App;

**Output:**

****

****

**PTO**

**HANDS ON 4**

**· Explain the need and Benefits of component life cycle**

**=**

The component lifecycle in React helps developers understand how a component behaves from creation to deletion. Lifecycle methods let us run specific code at each stage, like fetching data when the component mounts or cleaning up timers before it’s removed. This improves performance, makes code predictable, and helps manage side effects like API calls or DOM changes more cleanly.

**· Identify various life cycle hook methods**

**=**

In React, lifecycle hook methods are special functions that get called at different stages of a class component's life. These include mounting (e.g., constructor(), componentDidMount()), updating (e.g., shouldComponentUpdate(), componentDidUpdate()), and unmounting (componentWillUnmount()). These hooks let us control what happens when a component is created, updated, or removed, such as fetching data, setting timers, or cleaning up. In functional components, similar behavior is handled using the useEffect() hook.

**· List the sequence of steps in rendering a component**

**=**

The sequence of steps in rendering a React component is like this:

1. Constructor – Initializes the component's state and binds methods. Next ,
2. Render – Returns the JSX to be displayed in the browser.
3. componentDidMount – It is called after the component is added to the DOM
4. componentDidUpdate – It runs after the component updates due to props or state changes.
5. componentWillUnmount – Called before the component is removed from the DOM.

**Create a new react application using create-react-app tool with the name as “blogapp”**

**=**

**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: [],

      error: null

    };

  }

  loadPosts() {

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

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

      .then((data) => {

        const posts = data.map(post => new Post(post.id, post.title, post.body));

        this.setState({ posts });

      })

      .catch((error) => {

        this.setState({ error });

      });

  }

  componentDidMount() {

    this.loadPosts();

  }

  componentDidCatch(error, info) {

    alert("An error occurred: " + error);

  }

  render() {

    const { posts, error } = this.state;

    if (error) {

      return <div>Error: {error.message}</div>;

    }

    return (

      <div>

        <h1>Blog Posts</h1>

        {posts.map(post => (

          <div key={post.id}>

            <h2>{post.title}</h2>

            <p>{post.body}</p>

            <hr />

          </div>

        ))}

      </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>

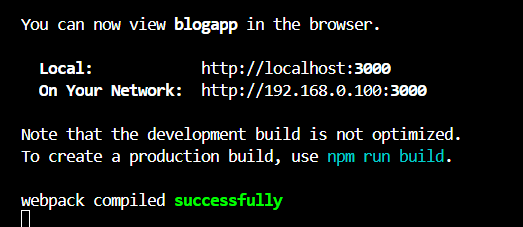
  );

}

export default App;

**Output :**

****

** PTO**

**HANDS ON 5 :**

**My Academy team at Cognizant want to create a dashboard containing the details of ongoing and completed cohorts. A react application is created which displays the detail of the cohorts using react component. You are assigned the task of styling these react components.**

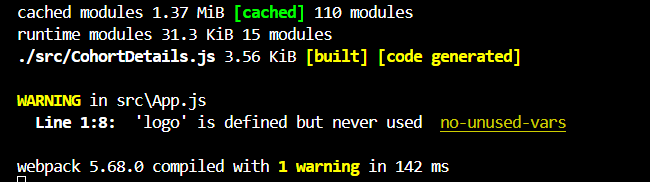
**=**

**Zip attatched :**

****

**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 ( updated for box and the green or blue feature) :**

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

function CohortDetails(props) {

    return (

        <div className={styles.box}>

            <h3 style={{ color: props.cohort.currentStatus === 'Ongoing' ? 'green' : 'blue' }}>

                {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;

**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;

**Cohort. js ( as it was) :**

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'),

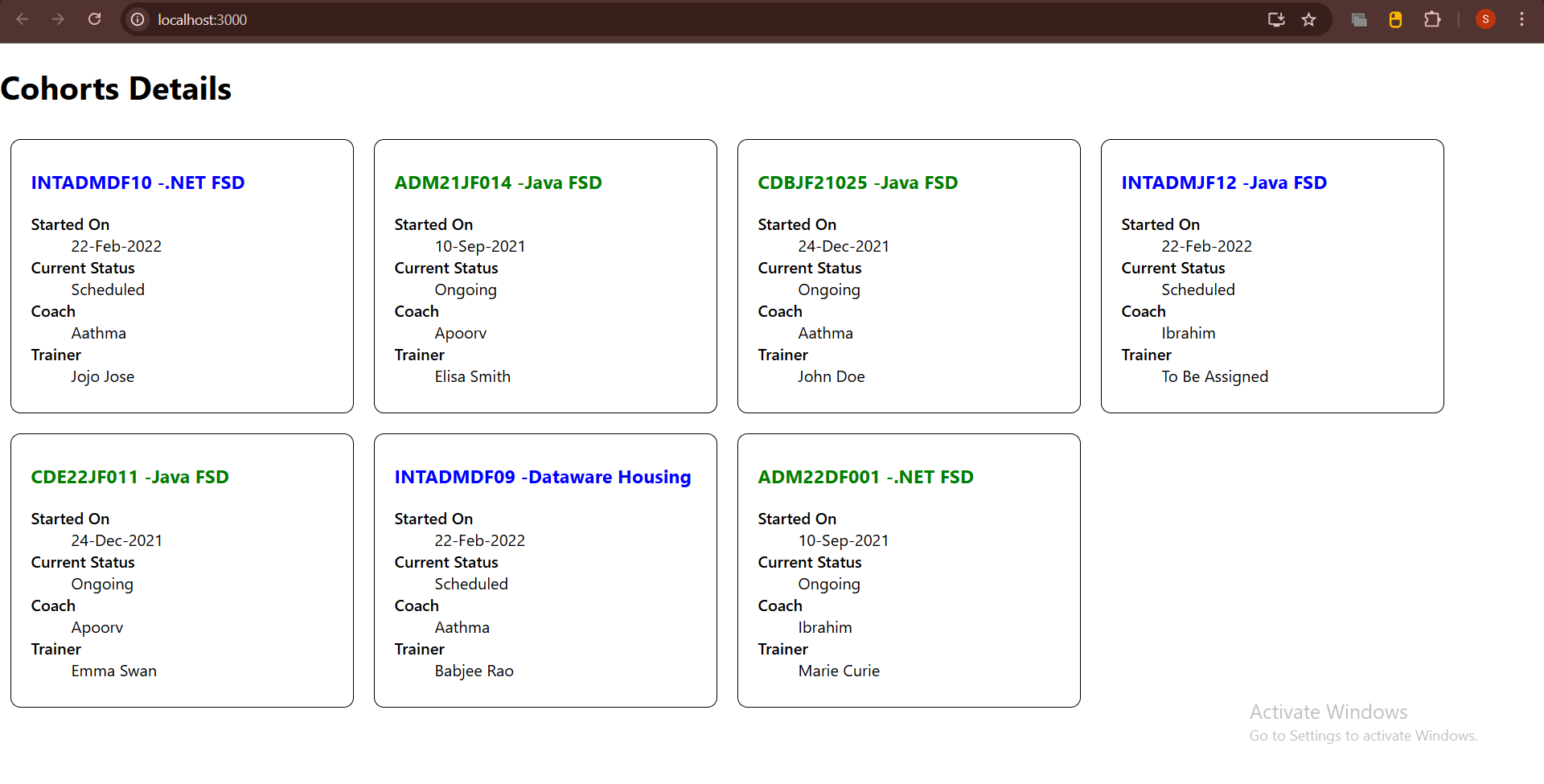
    new Cohort('INTADMDF09','22-Feb-2022', 'Dataware Housing', 'Babjee Rao','Aathma', 'Scheduled'),

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

];

export {Cohort, CohortsData};

**OUTPUT :**

****

**Assignment details :**

**Name : Swarnadri Sekhar Mukherjee**