**Week 7 HandsOn:**

**HandsOn (** filename **:- 9. reactJS-HOL)**

**List the features of ES6**

**=**

ES6 was a big update to JavaScript that made coding much cleaner and more powerful. It added things like let and const for safer variable handling, arrow functions for shorter syntax, template strings for easier text formatting, classes, modules, promises, and data structures like Map and Set.and more.

**· Explain JavaScript let**

**=**

**let** is used to declare a variable that can change later, but it only works within the block it’s defined in. It helps to avoid some of the confusion and bugs that came with the older var.

**· Identify the differences between var and let**

**= var** is function-scoped and can be re-declared, which often causes bugs. let is more modern and safer, since it’s block-scoped and doesn’t allow re-declaring the same variable in the same scope.

**· Explain JavaScript const**

**=**

Wee use **const** when we want a variable that never changes. It’s also block-scoped like let. If the value is an object or array, we can still change the contents but can’t assign a whole new object to it.

**· Explain ES6 class fundamentals**

**=**

ES6 introduced a simple and clear way to work with classes. A class groups together a constructor and methods . It is basically a cleaner way to write object-oriented code in JavaScript.

**· Explain ES6 class inheritance**

**=** Inheritance lets one class use features from another. In ES6, we can make a new class that extends an existing one, and use super() to call the parent’s constructor. This helps to reuse and organize code better.

**· Define ES6 arrow functions**

**=**

Arrow functions are a shortcut for writing functions. They're shorter and don’t have their own this, which can be super useful in some situations like callbacks.

**· Identify set(), map()**

**=**

* A Set is like a list, but it only stores unique values and no duplicates allowed.
* A Map is like an object, but better for key-value pairs, especially when keys aren't just strings. It keeps the order in which ywe add items and gives more useful methods.

Cricketapp:

**App.js :**

import React from 'react';

import OddPlayers from './OddPlayers';

import EvenPlayers from './EvenPlayers';

import ListOfPlayers from './ListOfPlayers';

import IndianPlayers from './IndianPlayers';

function App() {

  const flag = true;

  const players = [

    { name: 'Jack', score: 50 },

    { name: 'Michael', score: 70 },

    { name: 'John', score: 40 },

    { name: 'Ann', score: 61 },

    { name: 'Elizabeth', score: 61 },

    { name: 'Sachin', score: 95 },

    { name: 'Dhoni', score: 100 },

    { name: 'Virat', score: 84 },

    { name: 'Jadeja', score: 64 },

    { name: 'Raina', score: 75 },

    { name: 'Rohit', score: 80 }

  ];

  // ✅ Keeping your original if...else + map logic

  const players70 = [];

  players.map((item) => {

    if (item.score < 70) {

      players70.push(item);

    }

  });

  const T20Players = ['Sachin1', 'Dhoni2', 'Virat3'];

  const RanjiPlayers = ['Rohit4', 'Yuvaraj5', 'Raina6'];

  const IndianPlayersList = T20Players.concat(RanjiPlayers); // using concat

  return (

    <div>

      {flag === false ? (

        <>

          <h1>Odd Players</h1>

          <OddPlayers players={IndianPlayersList} />

          <h1>Even Players</h1>

          <EvenPlayers players={IndianPlayersList} />

          <h1>List of Indian Players Merged:</h1>

          <IndianPlayers players={IndianPlayersList} />

        </>

      ) : (

        <>

          <h1>List of Players:</h1>

          <ListOfPlayers players={players} />

          <h1>List of Players having Scores Less than 70</h1>

          <ul>

            {players70.map((item) => (

              <li key={item.name}>{item.name}</li>

            ))}

          </ul>

        </>

      )}

    </div>

  );

}

export default App;

**OddPlayers.js:**

import React from 'react';

export default function OddPlayers({ players }) {

  const [first, , third, , fifth] = players;

  return (

    <ul>

      <li>First : {first}</li>

      <li>Third : {third}</li>

      <li>Fifth : {fifth}</li>

    </ul>

  );

}

**EvenPlayers.js:**

import React from 'react';

export default function EvenPlayers({ players }) {

const [, second, , fourth, , sixth] = players;

return (

<ul>

<li>Second : {second}</li>

<li>Fourth : {fourth}</li>

<li>Sixth : {sixth}</li>

</ul>

);

}

**IndianPlayers.js:**

import React from 'react';

export default function IndianPlayers({ players }) {

return (

<ul>

{players.map((\_, index) => (

<li key={index}>Mr. {["First", "Second", "Third", "Fourth", "Fifth", "Sixth"][index]} Player</li>

))}

</ul>

);

}

**ListOfPlayers.js:**

import React from 'react';

function ListOfPlayers({ players }) {

  return (

    <ul>

      {players.map((item) => (

        <li key={item.name}>

          Mr. {item.name} <span>({item.score})</span>

        </li>

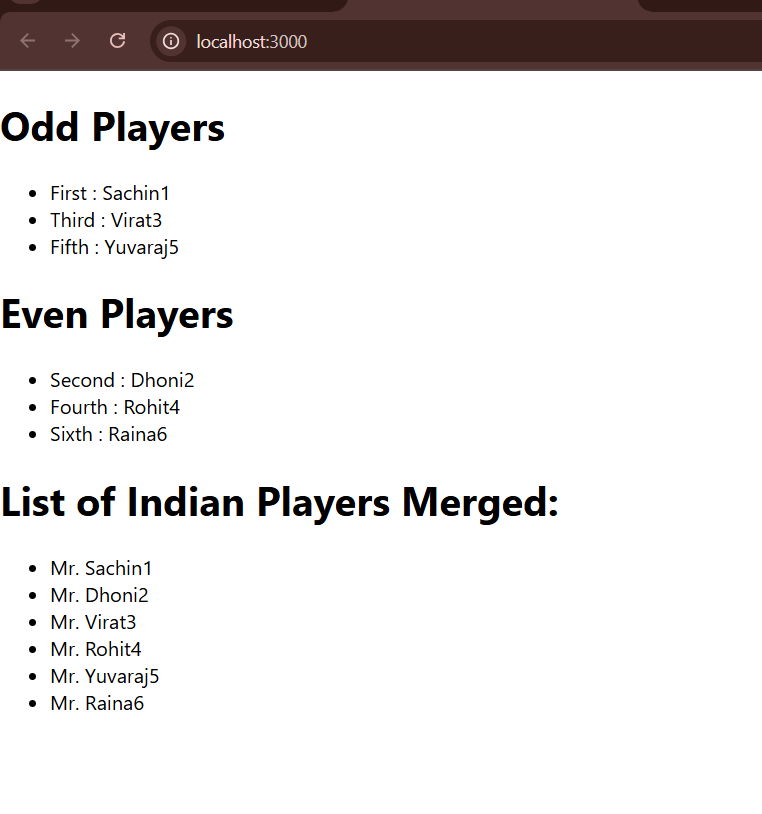
      ))}

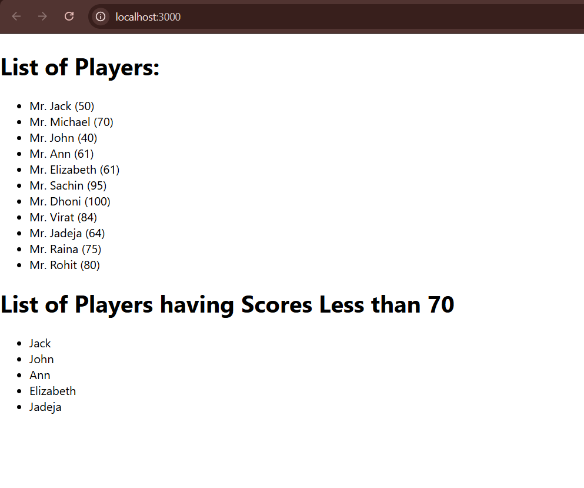
    </ul>

  );

}

export default ListOfPlayers;

**Output when flag =true : Output when flag = false :**

****

**HandsOn (** filename**:- 10 reactJS-HOL ) :**

**Define JSX**

JSX (JavaScript XML) is a syntax that lets us write HTML-like code in JavaScript. It makes React code easier to read and write. Instead of calling React.createElement() every time, we can just write something like <h1>Hello</h1>.

**Explain ECMA Script**

ECMAScript is the standard behind JavaScript. ES6 added many cool features like let, const , arrow functions, classes, and modules that modern JavaScript heavily relies on.

**Explain React.createElement()**

This is the function React uses under the hood to create virtual DOM elements. For example, React.createElement('h1', null, 'Hello') creates an <h1>Hello</h1> element

**Explain how to create React nodes with JSX**

With JSX, we create nodes (elements) just by writing HTML-like tags in your JavaScript. Example: const element = <h1>Hello, world!</h1>; This creates a React node that we can render to the DOM.

**Define how to render JSX to the DOM**

We render JSX using ReactDOM.render(). For example: ReactDOM.render(<App />, document.getElementById('root'));

This tells React to take the <App /> component and display it inside the element with id="root".

**Explain how to use JavaScript expressions in JSX**

JSX lets you embed any JavaScript expression inside curly braces {}. Example: const name = "Swarnadri"; const element = <h1>Hello, {name}</h1>; This will render: ***Hello, Swarnadri***

**Explain how to use inline CSS in JSX**

You can use inline CSS by passing a JavaScript object to the style attribute. The CSS properties must be in camelCase. Example: const headingStyle = { color: 'green', fontSize: '20px' }; const element = <h1 style={headingStyle}>Hi Guys</h1>; This will apply the styles directly to the <h1> element.

**App.js:**

import React from 'react';

import './App.css';

function App() {

const element = "Office Space";

const itemName = [

{ Name: "DBS", Rent: 50000, Address: "Chennai", imgSrc: "https://cdn.pixabay.com/photo/2016/10/16/10/29/office-space-1744801\_1280.jpg" },

{ Name: "ABC", Rent: 72000, Address: "kOLKATA", imgSrc: "https://cdn.pixabay.com/photo/2015/04/20/06/46/office-730681\_1280.jpg" },

{ Name: "DEF", Rent: 59000, Address: "PUNE", imgSrc: "https://images.unsplash.com/photo-1593642532973-d31b6557fa68" },

{ Name: "MNO", Rent: 38000, Address: "Delhi", imgSrc: "https://cdn.pixabay.com/photo/2020/02/10/22/48/business-4837890\_1280.jpg" },

];

return (

<div className="App">

<h1>{element} , at Affordable Range</h1>

{itemName.map((office, index) => {

const jsxat = <img src={office.imgSrc} width="25%" height="25%" alt="Office Space" />;

let colors = [];

if (office.Rent <= 60000) {

colors.push("textRed");

} else {

colors.push("textGreen");

}

return (

<div key={index} style={{ marginBottom: '40px' }}>

{jsxat}

<h2>Name: {office.Name}</h2>

<h3 className={colors[0]}>Rent: Rs. {office.Rent}</h3>

<h3>Address: {office.Address}</h3>

</div>

);

})}

</div>

);

}

export default App;

**index.css:**

.App {

padding: 25px;

}

.textRed {

color: red;

}

.textGreen {

color: green;

}

img {

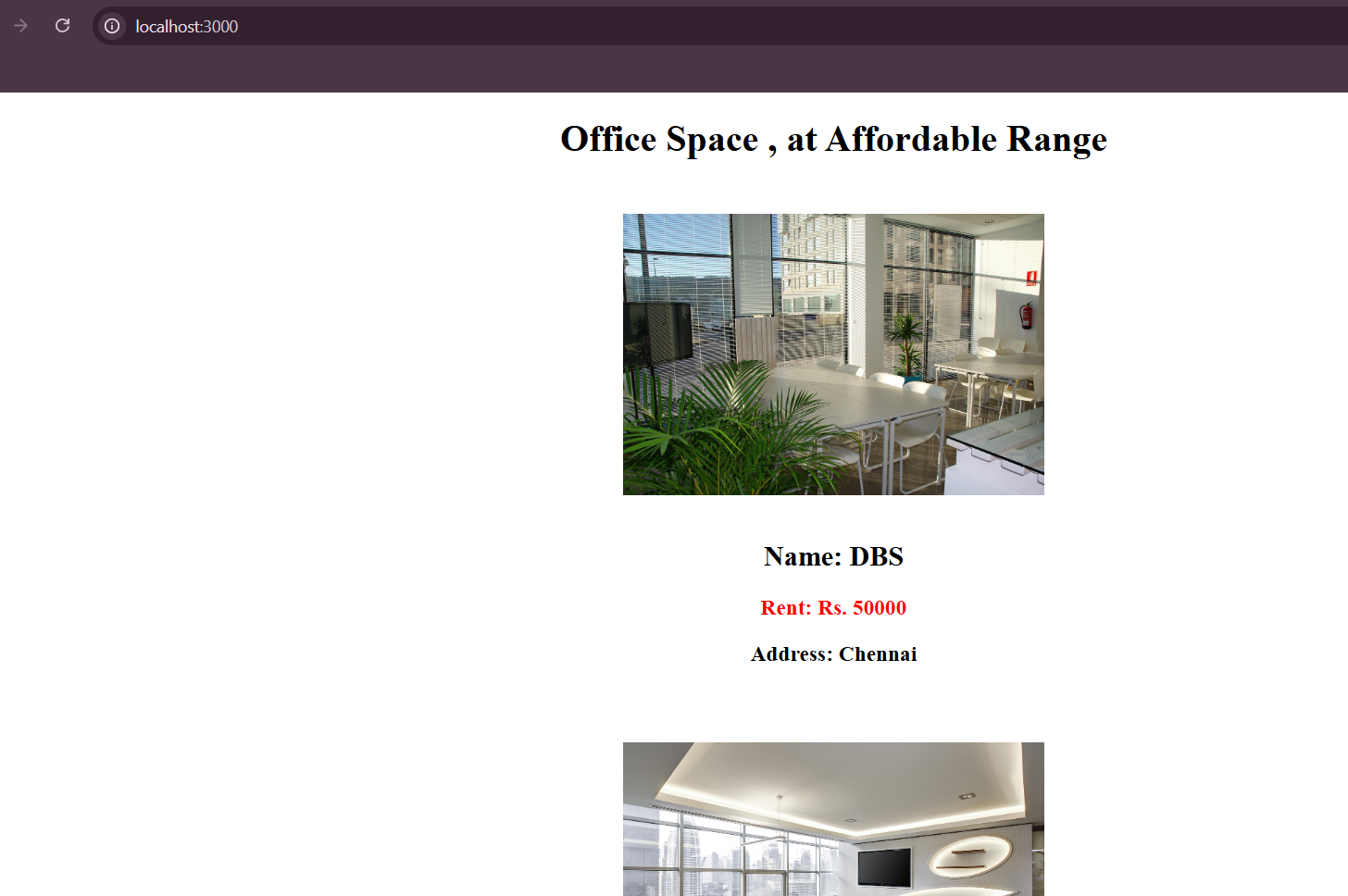
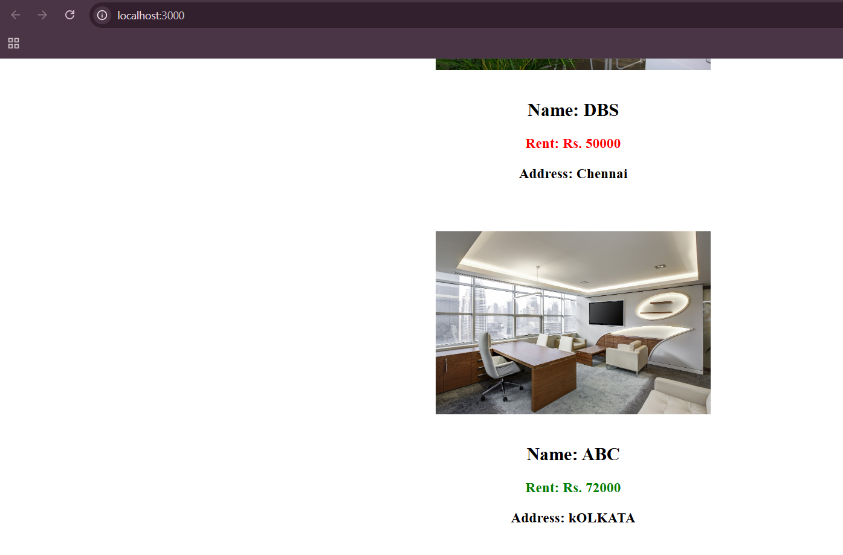
border-radius: 10px;

margin-top: 10px;

padding: 15px;

};

**Output:**

****

Output is like this…..

**HandsOn (**filename **:- 11. reactJS – HOL ) :**

**Explain React Events**

React events work similarly to browser DOM events , but with some differences in syntax and behavior. We can handle events in React using JSX, and they are wrapped in something called a synthetic event to make them consistent across all browsers.

**Explain Event Handlers**

Event handlers are functions we define to respond to events like clicks or form submissions. In React, we pass them as props in JSX. For example:

<button onClick={handleClick}>Click </button>

Here, handleClick is the event handler that runs when the button is clicked.

**Define Synthetic Event**

A Synthetic Event is React’s wrapper around the browser’s native event system. It works the same way as native events but provides consistent behavior across all browsers. React pools these events for performance, so they behave uniformly regardless of the platform.

**Identify React Event Naming Convention**

In React, event names use camelCase instead of lowercase. For example, we use onClick instead of onclick, and onChange instead of onchange. Also, we pass the handler function directly, not as a string.

**Q> Create “Increment” button to increase the value of the counter and “Decrement” button to decrease the value of the counter. The “Increase” button should invoke multiple methods. a. To increment the value b. Say Hello followed by a static message.**

**2. Create a button “Say Welcome” which invokes the function which takes “welcome” as an argument.**

**3. Create a button which invokes synthetic event “OnPress” which display “I was clicked” Create a “CurrencyConvertor” component which will convert the Indian Rupees to Euro when the Convert button is clicked. Handle the Click event of the button to invoke the handleSubmit event and handle the conversion of the euro to rupees**

**=**

**App.js :**

import React, { useState } from 'react';

import './App.css';

function CurrencyConvertor() {

const [amount, setAmount] = useState('');

const [currency, setCurrency] = useState('Euro');

function handleSubmit(event) {

event.preventDefault();

if (currency === 'Euro') {

const rupeeRate = 80;

const result = parseFloat(amount) \* rupeeRate;

alert(`Converting to Euro. Amount is ${result}`);

} else {

alert("Unsupported currency selected");

}

}

return (

<div>

<h2 className="green" >Currency Convertor!!!</h2>

<form onSubmit={handleSubmit}>

<label>Amount: </label>

<input

type="number"

value={amount}

onChange={(e) => setAmount(e.target.value)}

required

/>

<br /><br />

<label>Currency: </label>

<select value={currency} onChange={(e) => setCurrency(e.target.value)}>

<option value="Euro">Euro</option>

</select>

<br /><br />

<button type="submit">Submit</button>

</form>

</div>

);

}

function App() {

const [count, setCount] = useState(0);

function incrementValue() {

setCount(count + 1);

}

function sayHello() {

alert("Hello! My name is Swarnadri");

}

function handleIncrement() {

incrementValue();

sayHello();

}

function handleDecrement() {

setCount(count - 1);

}

function sayWelcome(message) {

alert(message);

}

function handleClick(event) {

alert("I was clicked");

}

return (

<div className="App">

<h1>{count}</h1>

<button onClick={handleIncrement}>Increment</button>

<br />

<button onClick={handleDecrement}>Decrement</button>

<br />

<button onClick={() => sayWelcome("Welcome to You, Hi")}>Say welcome</button>

<br />

<button onClick={handleClick}>Click on me</button>

<CurrencyConvertor />

</div>

);

}

export default App;

**App.css :**

.App {

text-align: center;

padding: 20px;

font-family: Arial, sans-serif;

}

button {

margin: 5px;

padding: 10px;

font-size: 16px;

}

input {

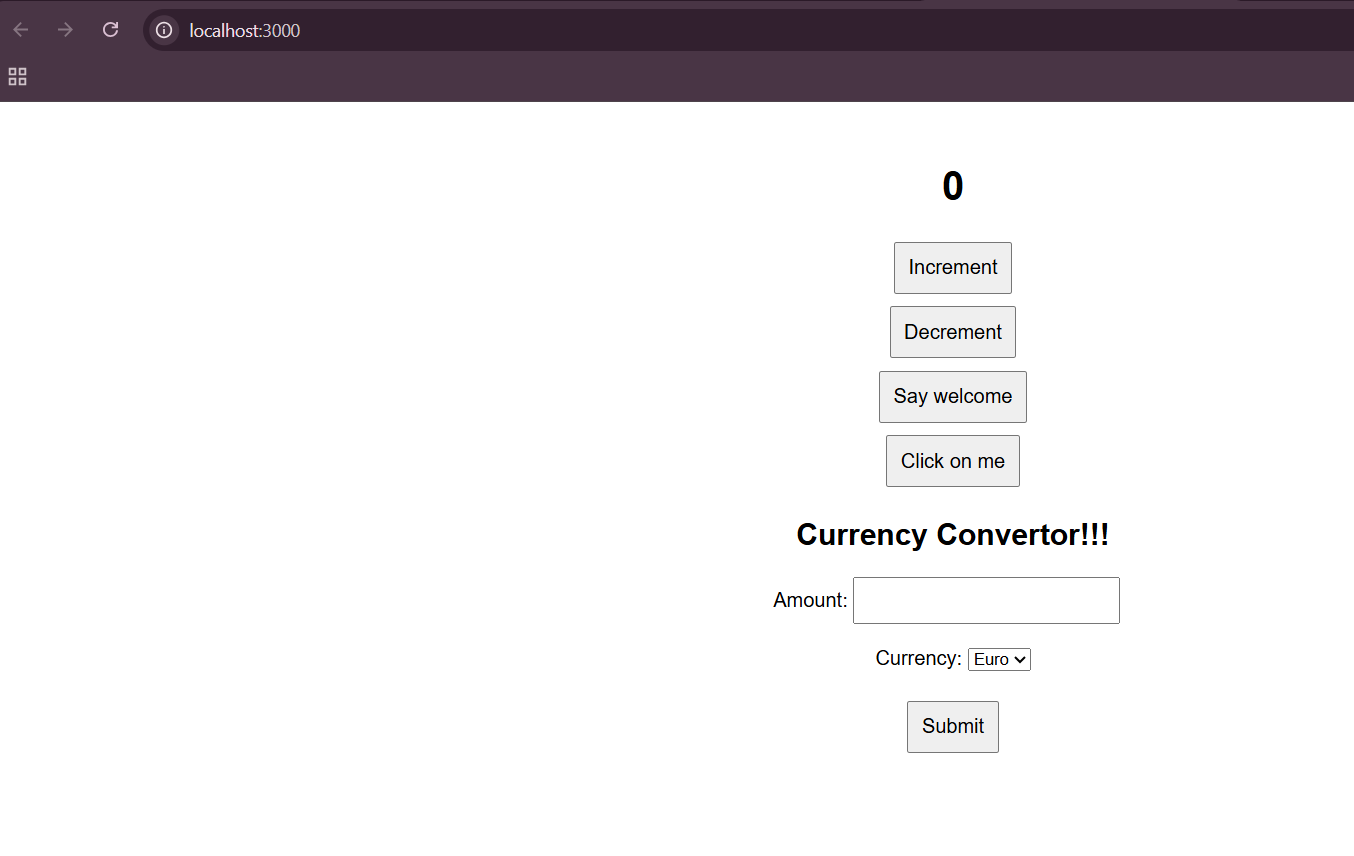
padding: 8px;

font-size: 16px;

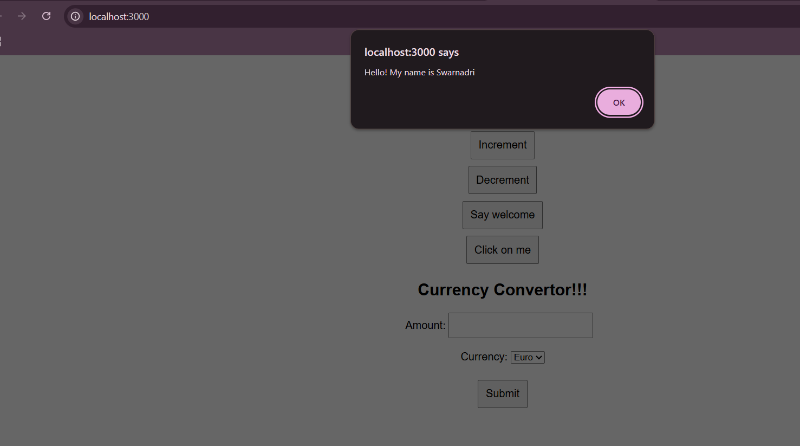
margin-right: 10px;

}

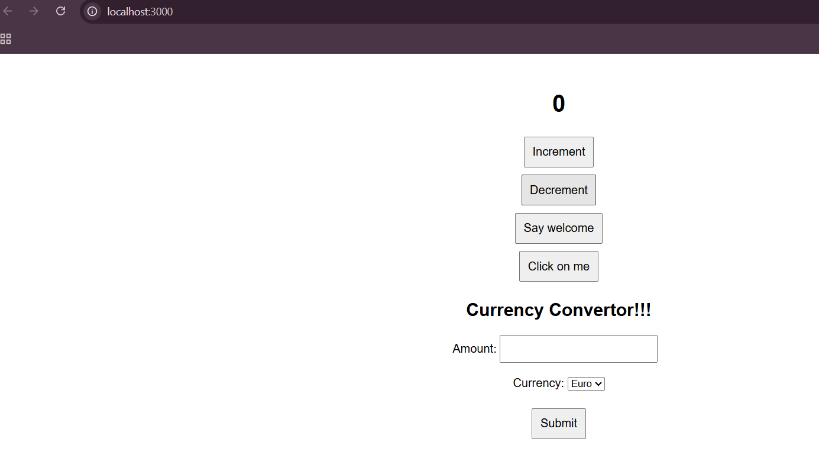
**Output:**



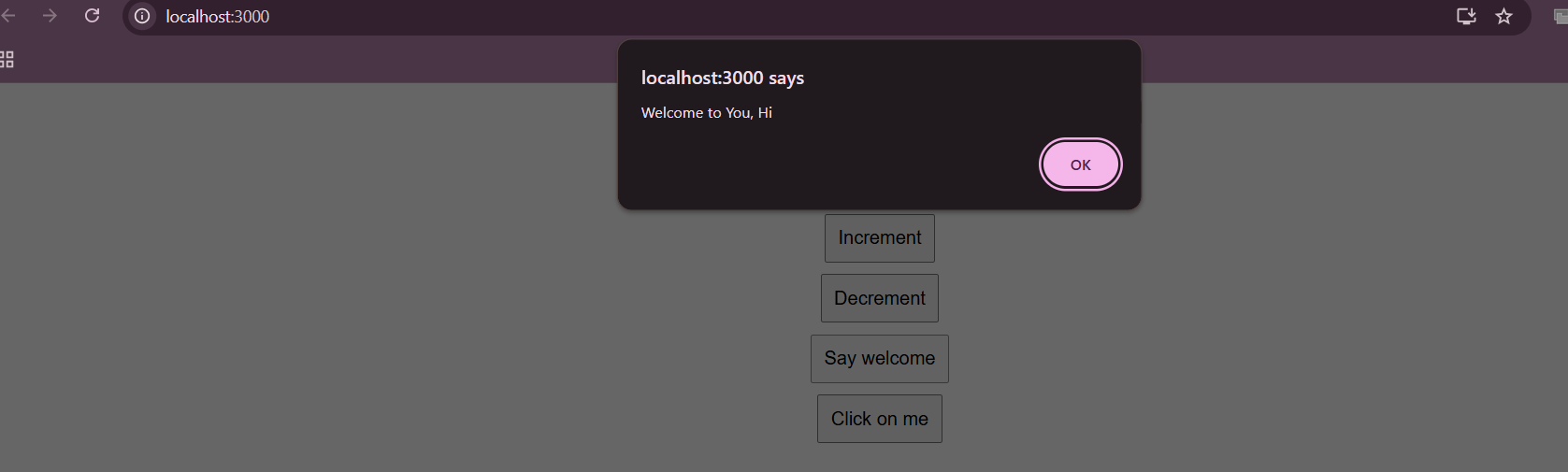
1. ** Increment :**



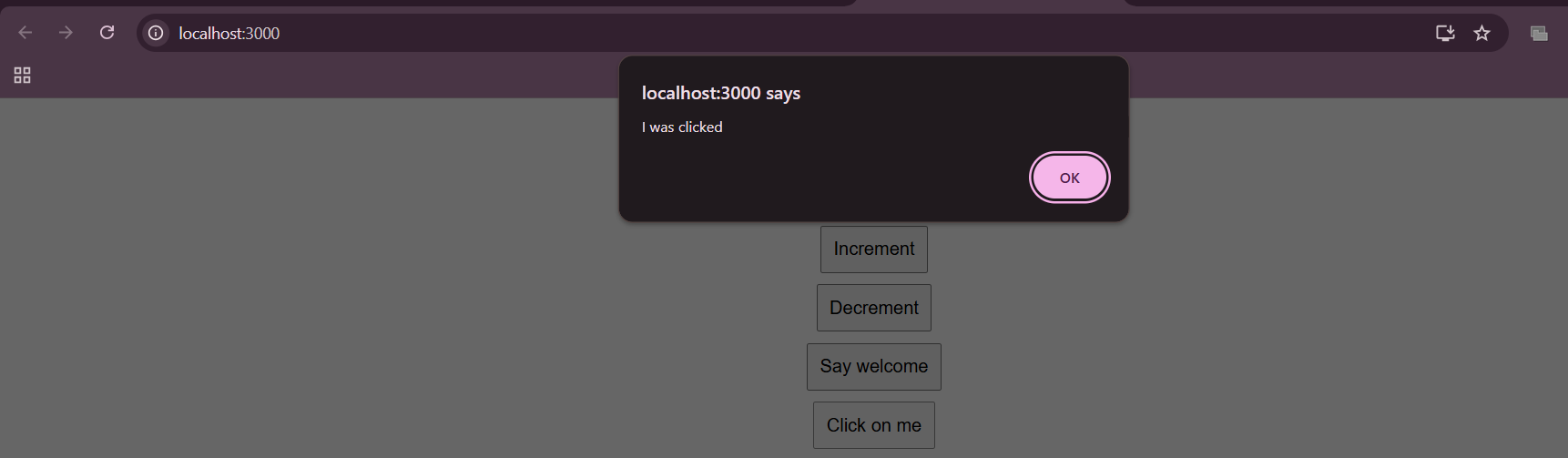
1. **Decrement :**

****

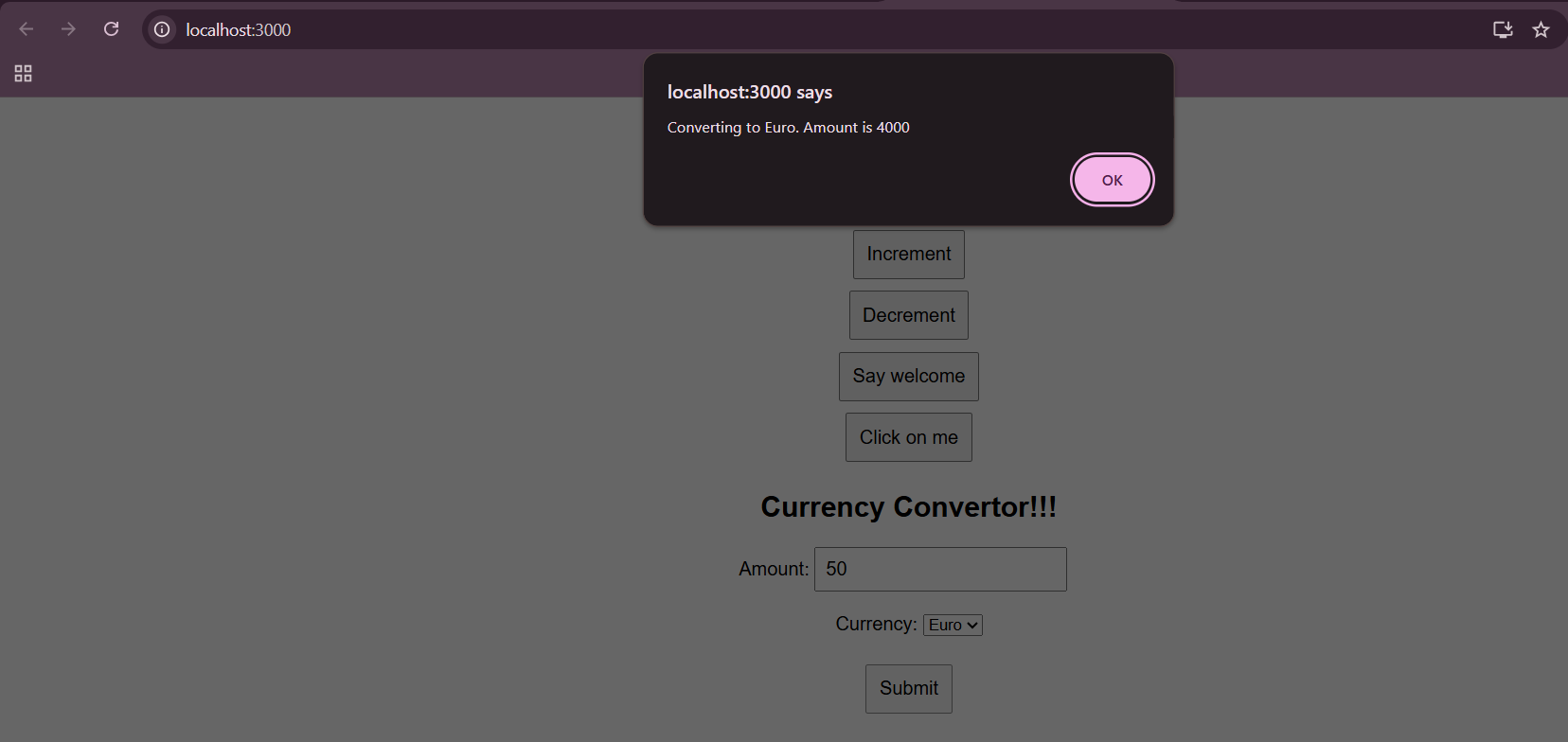
1. **Welcome :**

****

1. **Click:**

****

1. **Currency convertor :**

****

**--------------------------------**

**Hands On (filename:- 12. reactJS – HOL) :**

**Explain Conditional Rendering in React**

Conditional rendering means showing different UI elements based on certain conditions. In React, we often use if statements, ternary operators (condition ? A : B), or logical AND (&&) inside JSX to control what we display depending on the app’s state or props.

**Define Element Variables**

Element variables are regular variables that store JSX elements. We can use them to conditionally assign and render content.

**Explain How to Prevent Components from Rendering**

To prevent a component from rendering, we can return null from its render method or function. This tells React not to render anything at all for that component

**Create a React Application named “ticketbookingapp” where the guest user can browse the page where the flight details are displayed whereas the logged in user only can book tickets.**

**The Login and Logout buttons should accordingly display different pages. Once the user is logged in the User page should be displayed. When the user clicks on Logout, the Guest page should be displayed.**

**=**

**App.js :**

import React, { useState } from 'react';

function GuestPage({ onLogin }) {

  return (

    <div style={{ textAlign: 'center', marginTop: '50px' }}>

      <h2>Please sign up.</h2>

      <button onClick={onLogin}>Login</button>

    </div>

  );

}

function UserPage({ onLogout }) {

  return (

    <div style={{ textAlign: 'center', marginTop: '50px' }}>

      <h2>Welcome back</h2>

      <button onClick={onLogout}>Logout</button>

    </div>

  );

}

function App() {

  const [isLoggedIn, setIsLoggedIn] = useState(false);

  return (

    <div>

      {isLoggedIn ? (

        <UserPage onLogout={() => setIsLoggedIn(false)} />

      ) : (

        <GuestPage onLogin={() => setIsLoggedIn(true)} />

      )}

    </div>

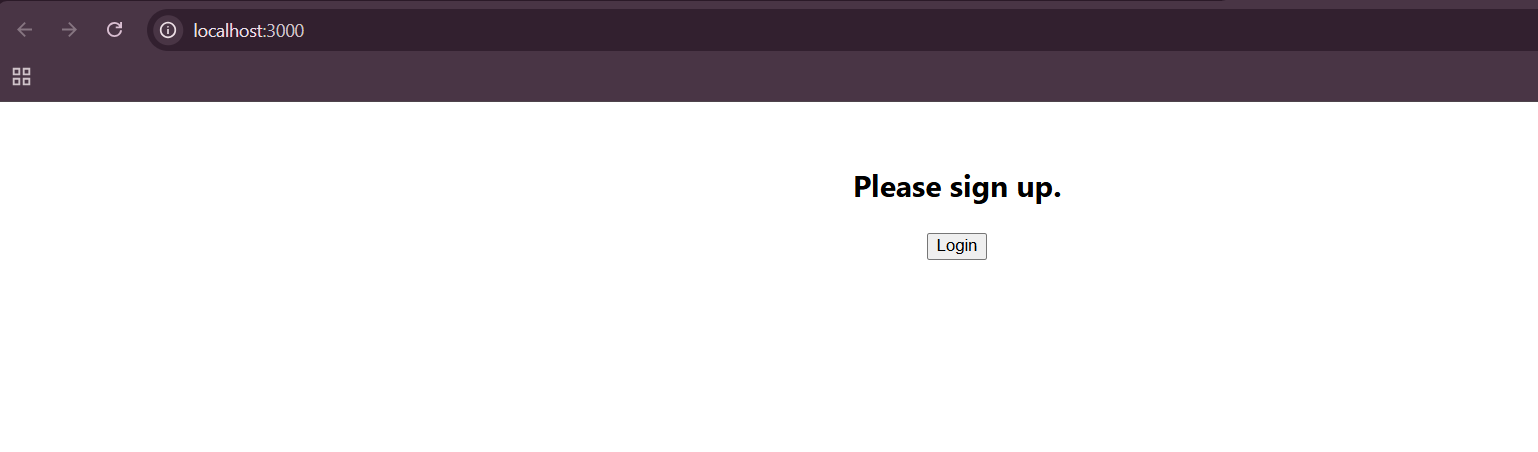
  );

}

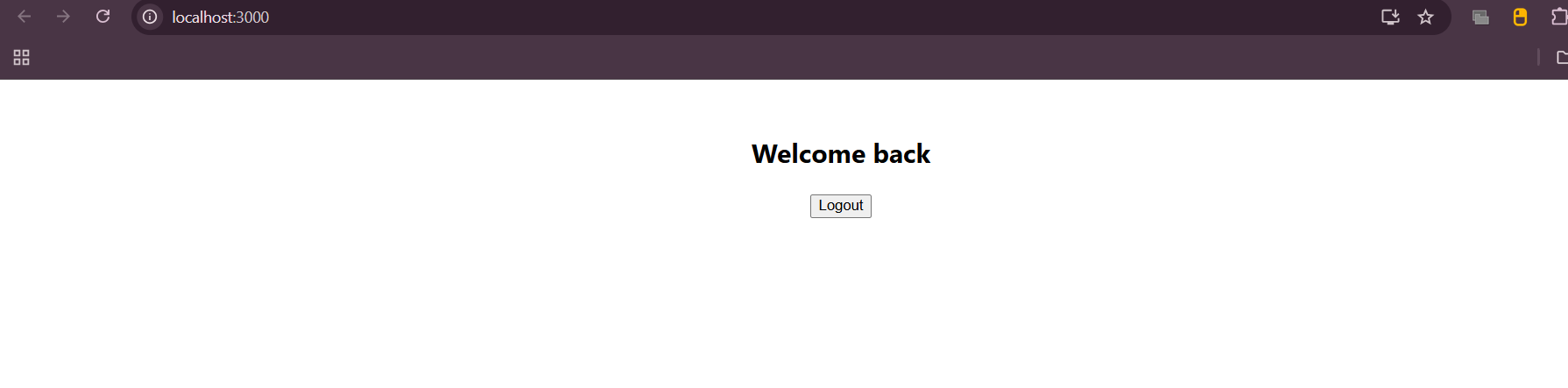
export default App;

**OUTPUT :**

WHEN NOT LOGGED IN



WHEN LOGGED IN

****

**Hands On ( filename:- 13. reactJS –HOL) :**

**Explain Various Ways of Conditional Rendering**

In React, we can conditionally render elements using if statements, ternary operators, logical && , or element variables. Each approach helps to decide what should be shown on the screen based on a condition like user status or data availability.

**Explain How to Render Multiple Components**

We can render multiple components together by placing them side by side inside a parent element like a <div>, or using React fragments (<> </>) to group them without adding extra DOM nodes.

**Define List Component**

A list component in React is one that takes a list of data and displays it by looping through each item. It usually returns multiple repeated components , making the UI dynamic and scalable.

**Explain About Keys in React Applications**

Keys are unique identifiers used by React to track elements in a list. They help React know which items have changed, been added, or removed, making updates more efficient. A good key is one that stays stable between renders.

**Explain How to Extract Components with Keys**

When building lists, we often move the repeated JSX into its own component. We pass the unique key prop to each extracted component so that React can track themproperly

**Explain React Map, map() Function**

map() function is commonly used in React to loop through an array and return a list of JSX elements. Rhis transforms each item into a component or element and is a clean way to generate dynamic lists in the UI.

**Create a React App named “bloggerapp” in with 3 components.**

**1. Book Details**

**2. Blog Details**

**3. Course Details**

**Implement this with as many ways possible of Conditional Rendering.**

**=**

**App.js :**

import React from 'react';

import './App.css';

import BookDetails from './BookDetails';

import BlogDetails from './BlogDetails';

import CourseDetails from './CourseDetails';

function App() {

const showBooks = true;

const showCourse = true;

const showBlog = true;

return (

<div className="container">

*{/\* Here Conditional rendering using && \*/}*

{showCourse && (

<div className="column">

<h1>Course Details</h1>

<CourseDetails />

</div>

)}

*{/\* Conditional rendering using ternary here \*/}*

{showBooks ? (

<div className="column">

<h1>Book Details</h1>

<BookDetails />

</div>

) : (

<p>No book details available</p>

)}

{/\* Blog section shown if true \*/}

{showBlog && (

<div className="column">

<h1>Blog Details</h1>

<BlogDetails />

</div>

)}

</div>

);

}

export default App;

**BookDetails.js :**

import React from 'react';

const books = [

{ id: 1, bname: 'Master React', price: 670 },

{ id: 2, bname: 'Deep Dive into Angular 11', price: 800 },

{ id: 3, bname: 'Mongo Essentials', price: 450 }

];

function BookDetails() {

const bookdet = (

<ul>

{books.map((book) => (

<div key={book.id}>

<h3>{book.bname}</h3>

<h4>{book.price}</h4>

</div>

))}

</ul>

);

return <>{bookdet}</>;

}

export default BookDetails;

**CourseDetails.js :**

import React from 'react';

function CourseDetails() {

const courses = [

{ name: 'Angular', date: '4/5/2021' },

{ name: 'React', date: '6/3/2021' }

];

return (

<ul>

{courses.map((c, index) => (

<div key={index}>

<h3>{c.name}</h3>

<p>{c.date}</p>

</div>

))}

</ul>

);

}

export default CourseDetails;

**BlogDetails.js :**

import React from 'react';

function BlogDetails() {

const showFirstBlog = true;

const showSecondBlog = true;

const selectedBlog = 2;

// if condition is being used here

let firstPost;

if (showFirstBlog) {

firstPost = (

<div>

<h3>React Learning</h3>

<p>Stephen Biz</p>

<p>Welcome to learning React!</p>

</div>

);

}

// if-else condition is what we are using…now

let secondPost;

if (showSecondBlog) {

secondPost = (

<div>

<h3>Installation</h3>

<p>Schwezdenier</p>

<p>You can install React from npm.</p>

</div>

);

} else {

secondPost = <p>No installation blog available</p>;

}

// switch-case rendering is what we have opted for now here downwards

let switchPost;

switch (selectedBlog) {

case 1:

switchPost = <p>it will be there</p>;

break;

case 2:

switchPost = <p>Selected: Installation Blog</p>;

break;

default:

switchPost = <p>Nothing</p>;

}

return (

<div>

{firstPost}

{secondPost}

<hr />

{switchPost}

</div>

);

}

export default BlogDetails;

**//We previously selected 2 so tht the output comes just like the output attatched in the hands on. So I have tried to maintain the same output as of the given handson**

**App.css :**

.container {

display: flex;

justify-content: space-around;

padding: 30px;

font-family: Arial, sans-serif;

}

.column {

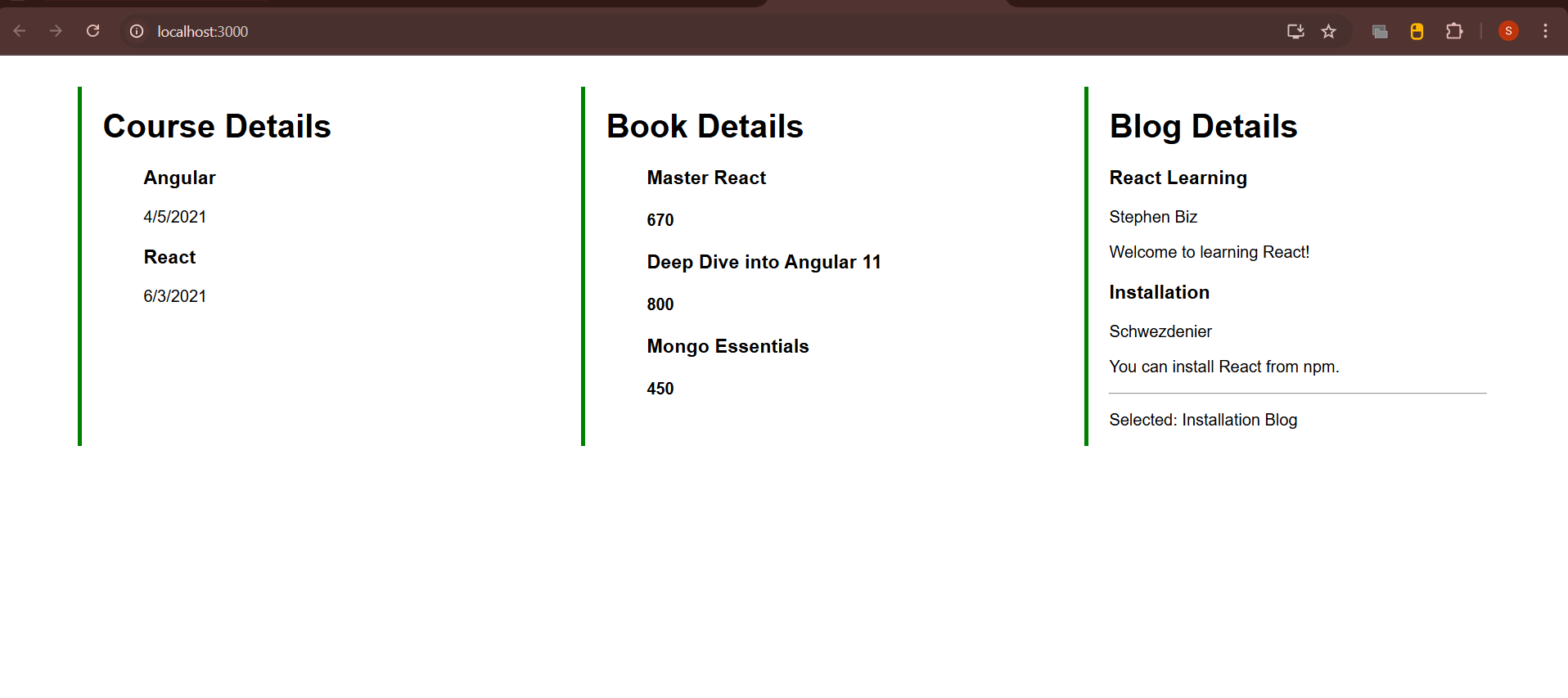
border-left: 4px solid green;

padding-left: 20px;

width: 25%;

}

**OUTPUT : (tried to make similar names s of the given handson output in the file)**

****

**Assignment By –**

**Name : Swarnadri Sekhar Mukherjee**