# WEEK -7

# HANDS ON 1:Create a React Application named “cricketapp”

# ES6 (ECMAScript 2015) — Features and Concepts

## 1. List the Features of ES6

* let and const for block-scoped variable declarations
* Arrow functions (=>)
* Classes and Inheritance
* Default function parameters
* Template literals (`Hello ${name}`)
* Destructuring (arrays & objects)
* Modules (import/export)
* Promises for async operations
* Map and Set collections
* Spread (...) and Rest (...args) operators
* Enhanced object literals
* for...of loop
* Symbol type
* Iterators and generators

## 2. Explain JavaScript let

* `let` is a block-scoped variable declaration introduced in ES6.
* It avoids the issues of hoisting and global scope pollution that come with `var`.
* Example:  
  let x = 10;  
  if (true) {  
   let x = 20;  
   console.log(x); // 20  
  }  
  console.log(x); // 10

## 3. Differences Between var and let

* | Feature | var | let |
* |----------------|----------------------------|---------------------------|
* | Scope | Function-scoped | Block-scoped |
* | Redeclaration | Allowed | Not allowed in same scope |
* | Hoisting | Hoisted (undefined) | Hoisted (not initialized) |
* | Global object | Adds to window | Does not add to window |

## 4. Explain JavaScript const

* `const` is used to declare constants whose value cannot be reassigned.
* It is block-scoped and must be initialized at declaration.
* Example:  
  const PI = 3.14;  
  PI = 3.1415; // Error: Assignment to constant variable
* Note: const prevents reassignment of reference, not mutation:  
  const arr = [1, 2];  
  arr.push(3); // Allowed

## 5. ES6 Class Fundamentals

* Syntax sugar over prototype-based inheritance.
* Includes constructor, methods, and uses `new`.
* Example:  
  class Person {  
   constructor(name) {  
   this.name = name;  
   }  
   greet() {  
   return `Hello, ${this.name}`;  
   }  
  }

## 6. ES6 Class Inheritance

* Use `extends` to inherit, and `super()` to call parent constructor.
* Example:  
  class Student extends Person {  
   constructor(name, roll) {  
   super(name);  
   this.roll = roll;  
   }  
  }

## 7. Define ES6 Arrow Functions

* Shorter syntax for writing functions.
* Does not bind its own `this`.
* Example:  
  const add = (a, b) => a + b;

## 8. Identify Set() and Map()

* Set:  
  - Collection of unique values  
  Example:  
  const ids = new Set();  
  ids.add(1);  
  ids.add(2);  
  ids.add(1); // ignored
* Map:  
  - Key-value pairs, can use objects as keys  
  Example:  
  const map = new Map();  
  map.set("name", "John");  
  map.set("age", 25);

# Code:

**IndianPlayers.js:**

import React from 'react';

const IndianPlayers = () => {

  const T20players = ["Rohit", "Virat", "Sky", "Pant"];

  const RanjiPlayers = ["Pujara", "Rahane", "Shaw"];

  // Merge both using spread operator

  const allPlayers = [...T20players, ...RanjiPlayers];

  // Destructure odd and even team

  const oddTeam = allPlayers.filter((\_, i) => i % 2 === 0);

  const evenTeam = allPlayers.filter((\_, i) => i % 2 !== 0);

  return (

    <div>

      <h2>All Players (Merged):</h2>

      <ul>

        {allPlayers.map((player, idx) => (

          <li key={idx}>{player}</li>

        ))}

      </ul>

      <h2>Odd Team Players</h2>

      <ul>

        {oddTeam.map((player, idx) => (

          <li key={idx}>{player}</li>

        ))}

      </ul>

      <h2>Even Team Players</h2>

      <ul>

        {evenTeam.map((player, idx) => (

          <li key={idx}>{player}</li>

        ))}

      </ul>

    </div>

  );

};

export default IndianPlayers;

**ListofPlayers.js:**

import React from 'react';

const ListofPlayers = () => {

  const players = [

    { name: "Rohit", score: 80 },

    { name: "Kohli", score: 67 },

    { name: "Gill", score: 75 },

    { name: "Surya", score: 45 },

    { name: "Hardik", score: 60 },

    { name: "Jadeja", score: 85 },

    { name: "Rahul", score: 58 },

    { name: "Bumrah", score: 72 },

    { name: "Shami", score: 35 },

    { name: "Kuldeep", score: 90 },

    { name: "Ashwin", score: 40 },

  ];

  const below70 = players.filter(p => p.score < 70);

  return (

    <div>

      <h2>All Players</h2>

      <ul>

        {players.map((player, idx) => (

          <li key={idx}>{player.name} - {player.score}</li>

        ))}

      </ul>

      <h2>Players with Score below 70</h2>

      <ul>

        {below70.map((player, idx) => (

          <li key={idx}>{player.name} - {player.score}</li>

        ))}

      </ul>

    </div>

  );

};

export default ListofPlayers;

**App.js:**

import React from 'react';

import ListofPlayers from './ListofPlayers';

import IndianPlayers from './IndianPlayers';

function App() {

  const flag = false; // Change to false to see IndianPlayers

  return (

    <div className="App">

      <h1>🏏 Cricket App</h1>

      {flag ? <ListofPlayers /> : <IndianPlayers />}

    </div>

  );

}

export default App;

**OUTPUT:**

**Flag=true**

****

**Flag=False:**



# HANDS ON 2:Create a React Application named “officespacerentalapp”

# ReactJS JSX and DOM Concepts

## 1. Define JSX

* JSX (JavaScript XML) is a syntax extension for JavaScript, commonly used with React to describe the UI.
* It allows writing HTML-like code directly in JavaScript, which gets transformed into React.createElement calls by Babel.
* Example:  
  const element = <h1>Hello, world!</h1>;

## 2. Explain about ECMA Script

* ECMAScript is a scripting-language specification standard maintained by ECMA International.
* JavaScript is based on ECMAScript and follows its rules and conventions.
* ES6 (ECMAScript 2015) introduced major improvements to JavaScript like `let`, `const`, arrow functions, classes, etc.

## 3. Explain React.createElement()

* React.createElement() is a core function that creates a virtual DOM representation.
* It takes three arguments: element type, props, and children.
* Example:  
  React.createElement('h1', { className: 'title' }, 'Hello World')

## 4. Explain how to create React nodes with JSX

* React nodes can be created using JSX syntax which is syntactic sugar over React.createElement().
* Example:  
  const node = <div>Hello JSX</div>;

## 5. Define how to render JSX to DOM

* JSX can be rendered to the actual DOM using `ReactDOM.render()` method.
* Example:  
  ReactDOM.render(<App />, document.getElementById('root'));

## 6. Explain how to use JavaScript expressions in JSX

* JavaScript expressions can be embedded inside JSX using curly braces `{}`.
* Only expressions (not statements) are allowed.
* Example:  
  const name = 'John';  
  const element = <h1>Hello, {name}</h1>;

## 7. Explain how to use inline CSS in JSX

* Inline styles in JSX are written as JavaScript objects using camelCase property names.
* Example:  
  const divStyle = { color: 'blue', backgroundColor: 'lightgray' };
* Usage:  
  <div style={divStyle}>Styled Text</div>

**CODE:**

**App.js:**

import React from 'react';

function App() {

  const offices = [

    {

      name: "DBS",

      rent: 50000,

      address: "Chennai",

      image: "https://cdn.sanity.io/images/uqxwe2qj/production/62db3c671745e98cb27690dff96f8033d2bb7f35-2048x1010.jpg?q=80&auto=format&fit=clip&w=1440"

    },

    {

      name: "WeWork",

      rent: 65000,

      address: "Bangalore",

      image: "https://thumbs.dreamstime.com/b/office-work-place-5879959.jpg?w=768"

    }

  ];

  const headingStyle = {

    fontWeight: 'bold',

    fontSize: '28px',

    marginBottom: '30px'

  };

  return (

    <div style={{ padding: '40px', fontFamily: 'Arial, sans-serif' }}>

      {/\* Page Heading \*/}

      <div style={headingStyle}>Office Space , at Affordable Range</div>

      {/\* List of Offices \*/}

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

        const rentStyle = {

          color: office.rent < 60000 ? 'red' : 'green',

          fontWeight: 'bold'

        };

        return (

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

            <img

              src={office.image}

              alt={office.name}

              width="300"

              height="200"

              style={{ marginBottom: '20px' }}

            />

            <div style={{ fontSize: '20px', marginBottom: '10px' }}>

              <strong>Name:</strong> {office.name}

            </div>

            <div style={{ ...rentStyle, fontSize: '18px', marginBottom: '10px' }}>

              Rent: Rs. {office.rent}

            </div>

            <div style={{ fontSize: '18px' }}>

              <strong>Address:</strong> {office.address}

            </div>

          </div>

        );

      })}

    </div>

  );

}

export default App;

**OUTPUT:**



# HANDS ON 3:Create a React Application named “eventexamplesapp”

# React Events and Event Handling

## 1. Explain React events

* React events are JavaScript events that are handled within a React component.
* They are similar to native DOM events but follow a React-specific approach using synthetic events.
* React wraps native events into SyntheticEvent objects to provide cross-browser compatibility.

## 2. Explain about event handlers

* Event handlers in React are functions triggered by user actions like clicks, input, form submission, etc.
* Handlers are passed as props to elements using camelCase syntax.
* Example:  
  <button onClick={handleClick}>Click Me</button>

## 3. Define Synthetic event

* A SyntheticEvent is a cross-browser wrapper around the browser’s native event.
* It normalizes event properties and behavior across different browsers.
* It wraps the native event with additional functionality and is reused by React’s event system.

## 4. Identify React event naming convention

* React uses camelCase for event names instead of lowercase.
* Example:  
  - onClick (not onclick)  
  - onChange (not onchange)  
  - onSubmit (not onsubmit)
* The handler function should be passed as a value, not as a string.

**CODE:**

**CurrencyConverter.js:**

import React, { Component } from 'react';

class CurrencyConvertor extends Component {

  constructor(props) {

    super(props);

    this.state = {

      rupees: '',

      euros: ''

    };

  }

  handleChange = (e) => {

    this.setState({ rupees: e.target.value });

  };

  handleSubmit = (e) => {

    e.preventDefault();

    const euroRate = 0.011; // Example rate

    const euros = (parseFloat(this.state.rupees) \* euroRate).toFixed(2);

    this.setState({ euros });

  };

  render() {

    return (

      <div style={{ marginTop: '30px' }}>

        <h2>Currency Convertor (INR to Euro)</h2>

        <form onSubmit={this.handleSubmit}>

          <label>Enter Amount in INR: </label>

          <input

            type="number"

            value={this.state.rupees}

            onChange={this.handleChange}

            required

          />

          <button type="submit" style={{ marginLeft: '10px' }}>

            Convert

          </button>

        </form>

        {this.state.euros && (

          <h3>Equivalent in Euro: €{this.state.euros}</h3>

        )}

      </div>

    );

  }

}

export default CurrencyConvertor;

**App.js:**

import React, { Component } from 'react';

import CurrencyConvertor from './CurrencyConvertor';

class App extends Component {

  constructor(props) {

    super(props);

    this.state = {

      count: 0,

    };

  }

  // 🔼 Increment Counter

  increment = () => {

    this.setState(prev => ({ count: prev.count + 1 }));

    this.sayHello();

  };

  // 🔽 Decrement Counter

  decrement = () => {

    this.setState(prev => ({ count: prev.count - 1 }));

  };

  // 👋 Say Hello

  sayHello = () => {

    alert("Hello! Counter Increased!");

  };

  // 🙋 Say Welcome

  sayWelcome = (message) => {

    alert("Message: " + message);

  };

  // 🎯 Synthetic Event Handler

  handleClick = (e) => {

    alert("I was clicked");

    console.log("Synthetic Event:", e);

  };

  render() {

    return (

      <div style={{ padding: '40px', fontFamily: 'Arial' }}>

        <h1>React Event Handling Examples</h1>

        <h2>Counter: {this.state.count}</h2>

        <button onClick={this.increment}>Increment</button>

        <button onClick={this.decrement} style={{ marginLeft: '10px' }}>

          Decrement

        </button>

        <br /><br />

        <button onClick={() => this.sayWelcome("Welcome to React Events!")}>

          Say Welcome

        </button>

        <br /><br />

        <button onClick={this.handleClick}>Synthetic OnPress</button>

        <br /><br />

        <CurrencyConvertor />

      </div>

    );

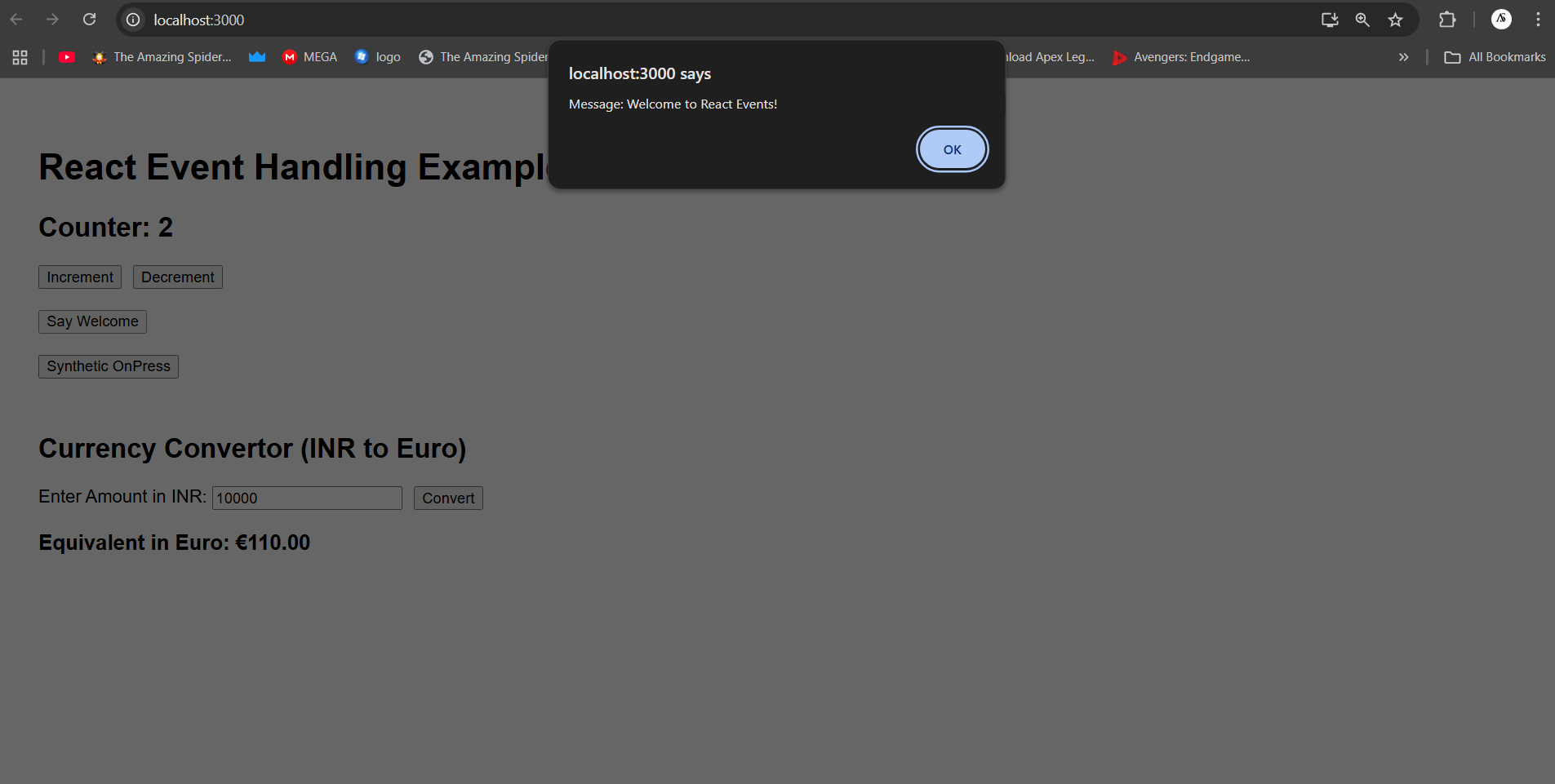
  }

}

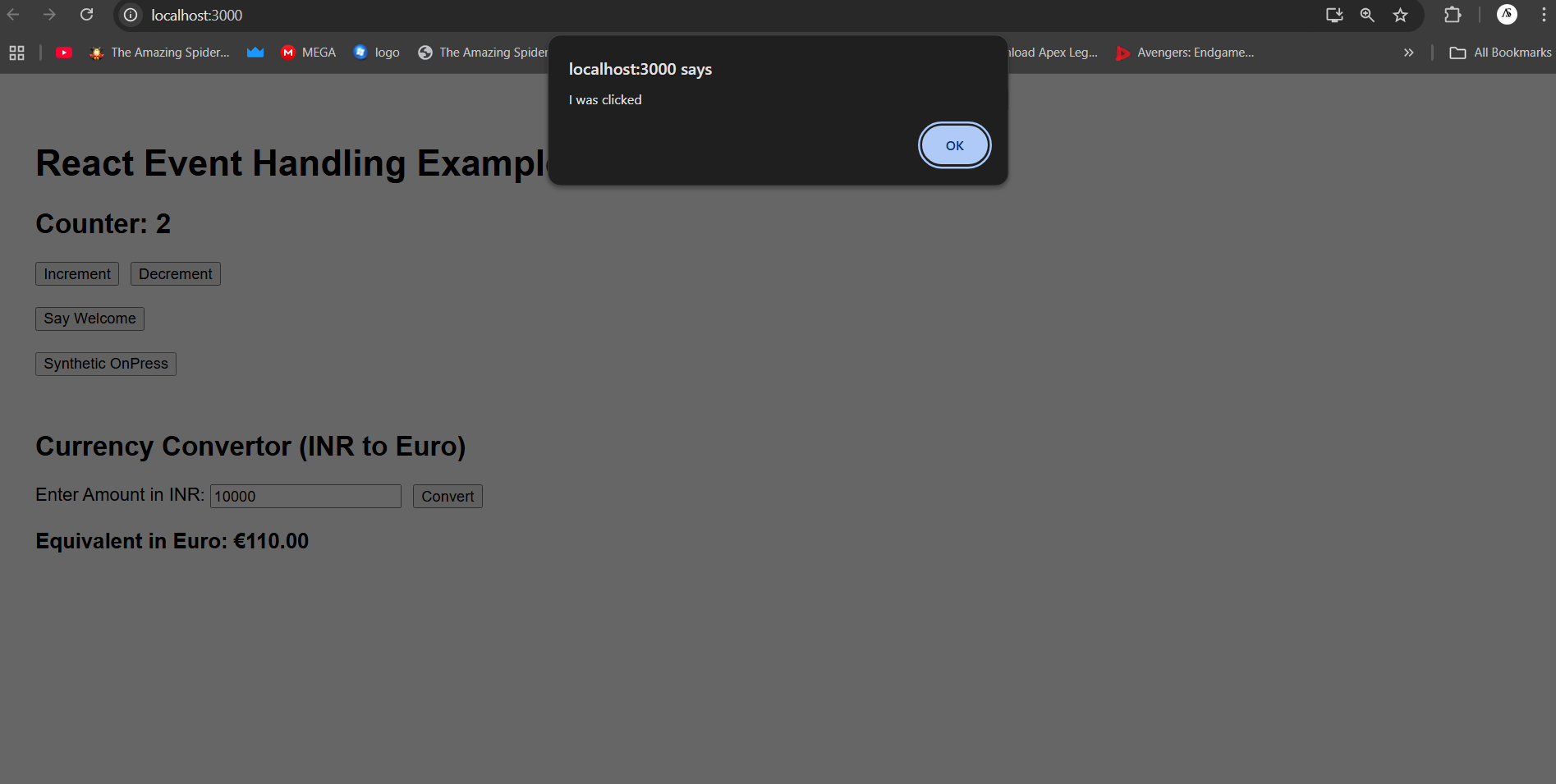
export default App;

**OUTPUT:**

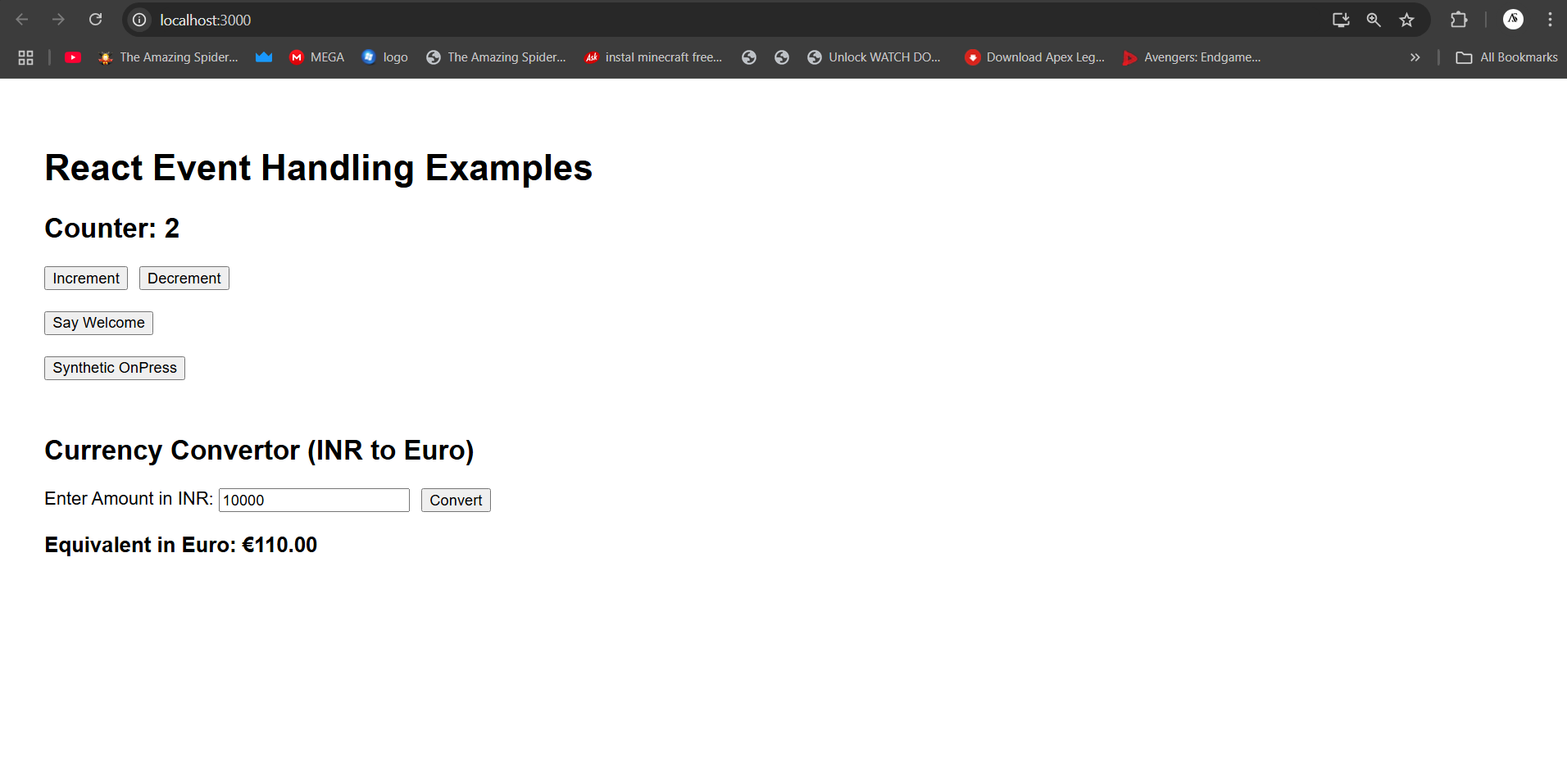
**Welcome:**



**Clicked:**

****

**Counter and Converter:**

****

# HANDS ON 4:Create a React Application named **“ticketbookingapp”**

# React Conditional Rendering Concepts

## 1. Explain about conditional rendering in React

* Conditional rendering in React allows components to render differently based on a condition.
* It is similar to how conditions work in JavaScript using `if`, `ternary`, or logical operators.
* Examples:
* Using if-statement:
* if (isLoggedIn) { return <Dashboard />; } else { return <Login />; }
* Using ternary operator:
* return isLoggedIn ? <Dashboard /> : <Login />;

## 2. Define element variables

* Element variables can hold JSX elements and help conditionally render parts of the component.
* They are useful to clean up large render logic.
* Example:
* let content;
* if (isLoggedIn) { content = <Dashboard />; } else { content = <Login />; }
* return <div>{content}</div>;

## 3. Explain how to prevent components from rendering

* React components can be prevented from rendering by returning `null` in the render method.
* This is useful for hiding components without unmounting them.
* Example:
* if (!this.props.visible) { return null; }
* return <div>This is visible</div>;

**Code:**

**GuestPage.js:**

import React from 'react';

function GuestPage() {

  return (

    <div>

      <h2>Welcome Guest</h2>

      <p>Flight Details:</p>

      <ul>

        <li>✈️ Flight: AI-202</li>

        <li>From: Delhi</li>

        <li>To: Mumbai</li>

        <li>Departure: 10:00 AM</li>

        <li>Arrival: 12:00 PM</li>

      </ul>

      <p>Please login to book your tickets.</p>

    </div>

  );

}

export default GuestPage;

**UserPage.js:**

import React from 'react';

function UserPage() {

  return (

    <div>

      <h2>Welcome User</h2>

      <p>You can now book your flight tickets.</p>

      <button>Book Ticket</button>

    </div>

  );

}

export default UserPage;

**App.js:**

import React, { Component } from 'react';

import GuestPage from './GuestPage';

import UserPage from './UserPage';

class App extends Component {

  constructor(props) {

    super(props);

    this.state = {

      isLoggedIn: false

    };

  }

  handleLogin = () => {

    this.setState({ isLoggedIn: true });

  };

  handleLogout = () => {

    this.setState({ isLoggedIn: false });

  };

  render() {

    let pageContent;

    let button;

    if (this.state.isLoggedIn) {

      pageContent = <UserPage />;

      button = <button onClick={this.handleLogout}>Logout</button>;

    } else {

      pageContent = <GuestPage />;

      button = <button onClick={this.handleLogin}>Login</button>;

    }

    return (

      <div style={{ padding: '40px', fontFamily: 'Arial, sans-serif' }}>

        <h1>Ticket Booking App</h1>

        {button}

        <hr />

        {pageContent}

      </div>

    );

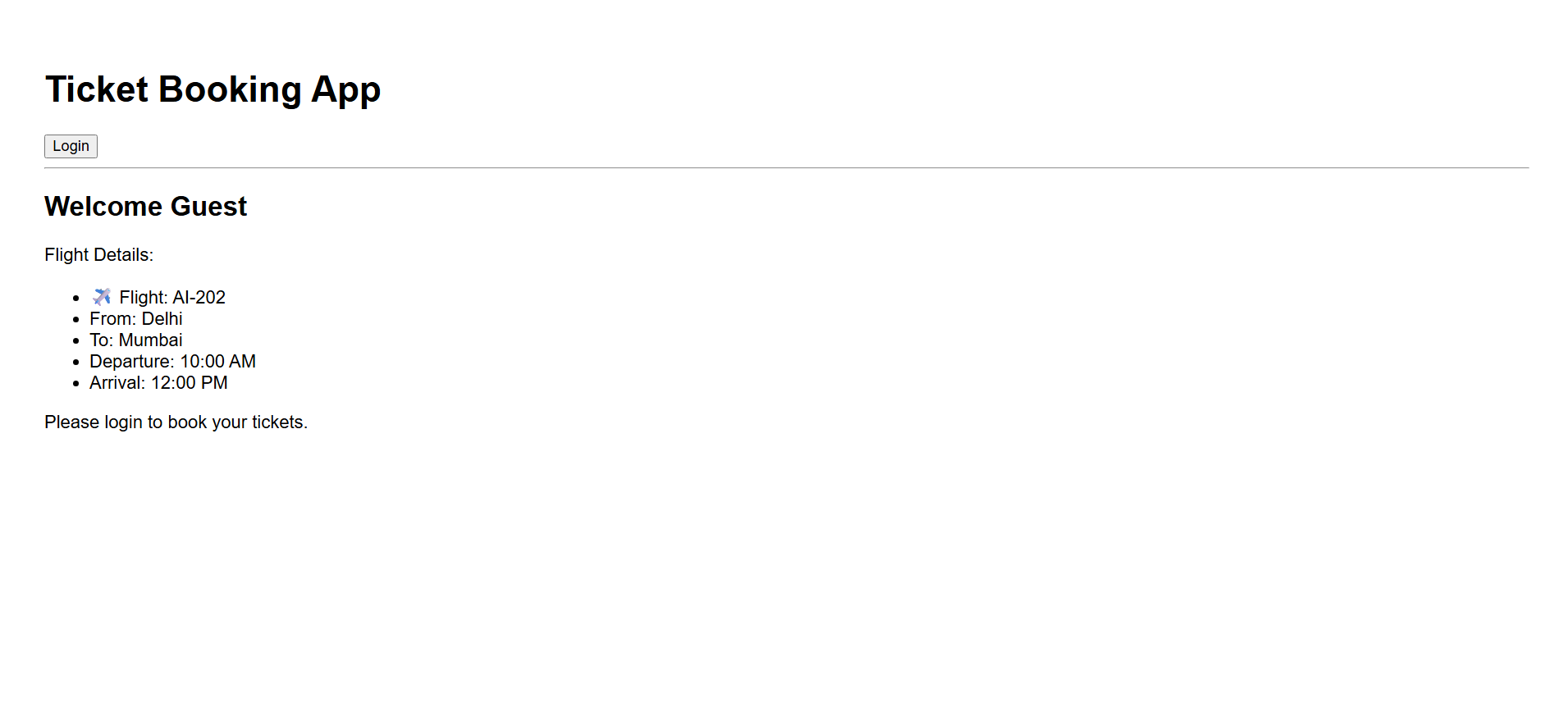
  }

}

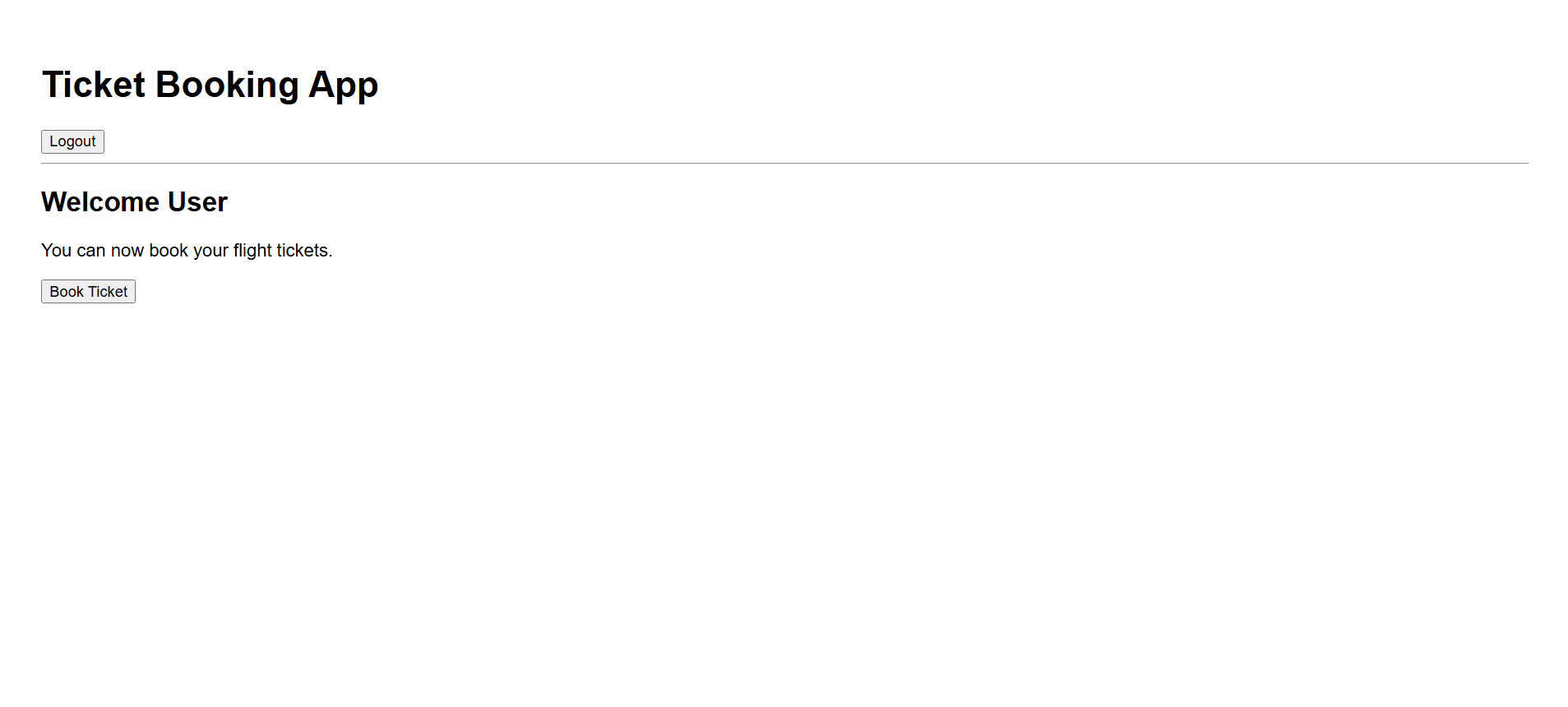
export default App;

**OUTPUT:**

**Guestpage:**

****

**Loginpage:**

****

# HANDS ON 5:Create a React Application named “bloggerapp”

# React Lists, Keys, and Conditional Rendering

## 1. Explain various ways of conditional rendering

* React supports multiple ways to conditionally render components or elements:
* 1. if-else statement:
* if (isLoggedIn) { return <Dashboard />; } else { return <Login />; }
* 2. Ternary operator:
* return isLoggedIn ? <Dashboard /> : <Login />;
* 3. Logical AND (&&) operator:
* return isLoggedIn && <Dashboard />;
* 4. Element variables:
* let content; if (isLoggedIn) { content = <Dashboard />; } else { content = <Login />; }

## 2. Explain how to render multiple components

* You can render multiple components together by including them inside a parent component.
* They can be wrapped using a single parent `<div>` or a React Fragment (`<>...</>`).
* Example:  
  return (  
   <>  
   <Header />  
   <Content />  
   <Footer />  
   </>  
  );

## 3. Define list component

* A list component in React is used to render a list of data dynamically using JavaScript's `map()` function.
* It accepts an array and renders individual items using child components or JSX.
* Example:  
  const items = ['One', 'Two'];  
  const listItems = items.map(item => <li>{item}</li>);

## 4. Explain about keys in React applications

* Keys help React identify which items have changed, are added, or are removed.
* They must be unique and stable (not random or index if possible).
* Example:  
  items.map(item => <li key={item.id}>{item.name}</li>)

## 5. Explain how to extract components with keys

* You can break down a list into smaller components by passing item data and assigning a `key` prop.
* Example:  
  function ListItem(props) { return <li>{props.value}</li>; }
* const listItems = numbers.map((number) => <ListItem key={number.toString()} value={number} />);

## 6. Explain React Map, map() function

* `map()` is a JavaScript array method used to transform arrays.
* In React, it is commonly used to render lists of components.
* Example:  
  const names = ['Alice', 'Bob'];  
  const items = names.map(name => <li>{name}</li>);

**CODE:**

**BlogDetails.js:**

import React from 'react';

function BlogDetails() {

  const blogs = [

    {

      title: "React Learning",

      author: "Stephen Biz",

      content: "Welcome to learning React!"

    },

    {

      title: "Installation",

      author: "Schwezdenier",

      content: "You can install React from npm."

    }

  ];

  return (

    <div style={{ padding: '20px' }}>

      <h2 style={{ fontWeight: 'bold' }}>Blog Details</h2>

      {blogs.map((blog, index) => (

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

          <div style={{ fontWeight: 'bold', fontSize: '18px' }}>{blog.title}</div>

          <div style={{ fontWeight: 'bold' }}>{blog.author}</div>

          <div>{blog.content}</div>

        </div>

      ))}

    </div>

  );

}

export default BlogDetails;

**BookDetails.js:**

import React from 'react';

function BookDetails() {

  const books = [

    { title: "Master React", price: "670" },

    { title: "Deep Dive into Angular 11", price: "800" },

    { title: "Mongo Essentials", price: "450" }

  ];

  return (

    <div style={{ padding: '20px', borderRight: '4px solid green' }}>

      <h2 style={{ fontWeight: 'bold' }}>Book Details</h2>

      {books.map((book, index) => (

        <div key={index} style={{ marginBottom: '15px' }}>

          <div style={{ fontWeight: 'bold', fontSize: '18px' }}>{book.title}</div>

          <div>{book.price}</div>

        </div>

      ))}

    </div>

  );

}

export default BookDetails;

**CourseDetails.js:**

import React from 'react';

function CourseDetails() {

  const courses = [

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

    { name: "React", date: "6/3/20201" }

  ];

  return (

    <div style={{ padding: '20px', borderRight: '4px solid green' }}>

      <h2 style={{ fontWeight: 'bold' }}>Course Details</h2>

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

        <div key={index} style={{ marginBottom: '15px' }}>

          <div style={{ fontWeight: 'bold', fontSize: '18px' }}>{course.name}</div>

          <div>{course.date}</div>

        </div>

      ))}

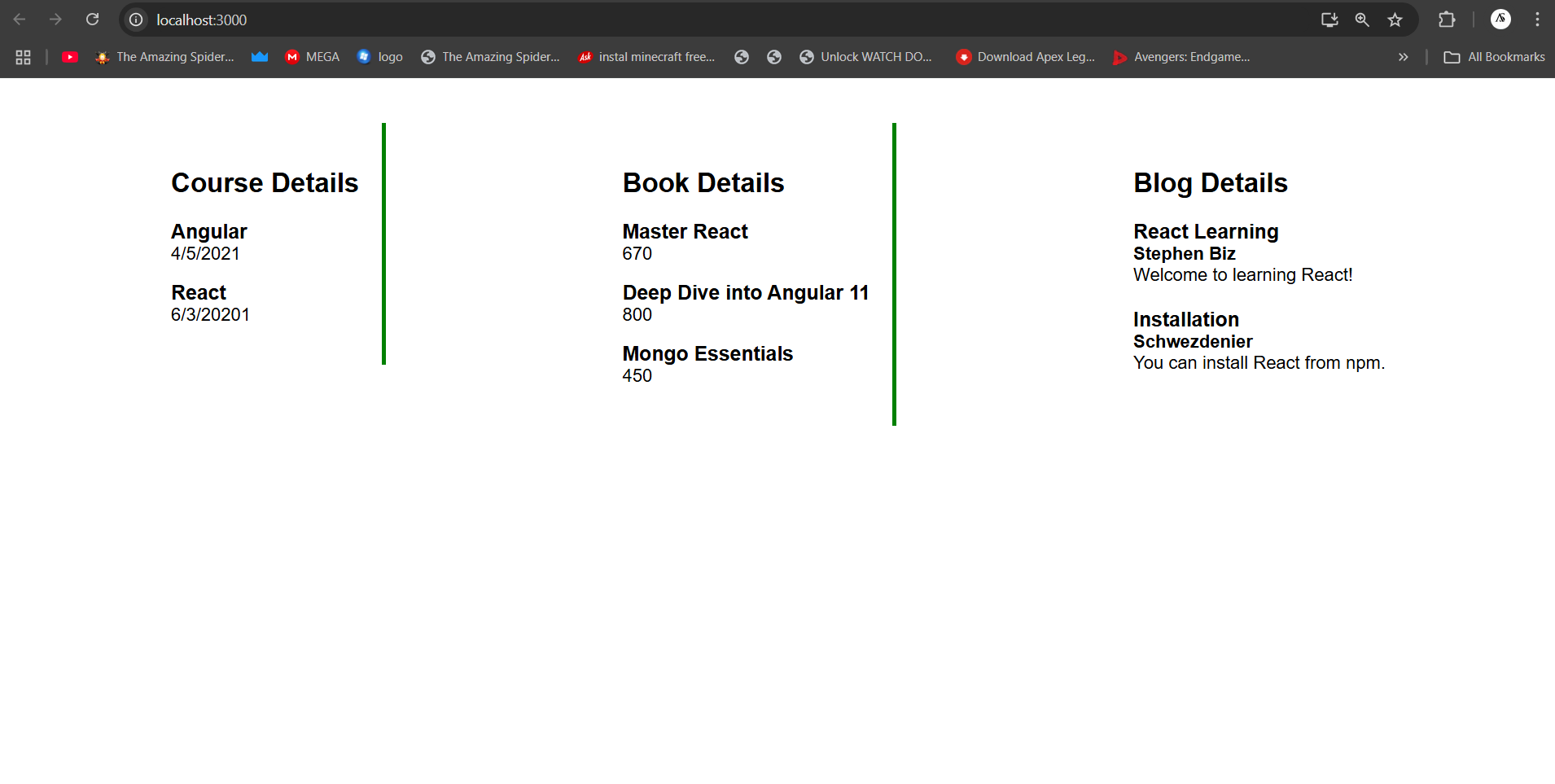
    </div>

  );

}

export default CourseDetails;

**OUTPUT:**

****