**WEEK 7\_REACTJS**

**Question : 01**

**Implementation of the required “cricketapp” using Create React App and the ES6 features (map, arrow functions, destructuring, spread, conditional rendering**

**Step 1: Create the React application**

Open a terminal and run:

npx create-react-app cricketapp

cd cricketapp

**Step 2: Replace src/App.js with the lab implementation**

Open src/App.js and replace its contents with:

// src/App.js

import React from "react";

// Component: ListofPlayers

const ListofPlayers = ({ players }) => {

return (

<div>

<h1>List of Players</h1>

<ul>

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

<li key={idx}>

Mr. {p.name} {p.score}

</li>

))}

</ul>

<hr />

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

<ul>

{players

.filter(p => p.score < 70) // arrow function + filter

.map((p, idx) => (

<li key={idx}>

Mr. {p.name} {p.score}

</li>

))}

</ul>

</div>

);

};

// Component: IndianPlayers

const IndianPlayers = ({

oddTeamPlayers,

evenTeamPlayers,

t20Players,

ranjiPlayers,

}) => {

// Merge using spread (ES6)

const merged = [...t20Players, ...ranjiPlayers];

return (

<div>

<h1>Indian Team</h1>

<div>

<h1>Odd Players</h1>

<ul>

{oddTeamPlayers.map((p, idx) => (

<li key={idx}>

{p.position} : {p.name}

</li>

))}

</ul>

</div>

<hr />

<div>

<h1>Even Players</h1>

<ul>

{evenTeamPlayers.map((p, idx) => (

<li key={idx}>

{p.position} : {p.name}

</li>

))}

</ul>

</div>

<hr />

<div>

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

<ul>

{merged.map((name, idx) => (

<li key={idx}>Mr. {name}</li>

))}

</ul>

</div>

</div>

);

};

// Root App

const App = () => {

const flag = true; // <-- Toggle this to false to show the IndianPlayers view

// Data for ListofPlayers

const players = [

{ name: "Jack", score: 50 },

{ name: "Michael", score: 70 },

{ name: "John", score: 40 },

{ name: "Ann", score: 61 },

{ name: "Elisabeth", 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 },

];

// IndianPlayers: destructure into odd/even by index (0-based)

const indianTeam = [

{ name: "Sachin", position: "First" }, // index 0 => odd (position 1)

{ name: "Dhoni", position: "Second" }, // index 1 => even

{ name: "Virat", position: "Third" },

{ name: "Rohit", position: "Fourth" },

{ name: "Yuvaraj", position: "Fifth" },

{ name: "Raina", position: "Sixth" },

];

const oddTeamPlayers = indianTeam.filter((\_, idx) => idx % 2 === 0);

const evenTeamPlayers = indianTeam.filter((\_, idx) => idx % 2 === 1);

// Two arrays to merge

const t20Players = ["First Player", "Second Player", "Third Player"];

const ranjiPlayers = ["Fourth Player", "Fifth Player", "Sixth Player"];

return (

<div style={{ padding: "16px" }}>

{flag === true ? (

<ListofPlayers players={players} />

) : (

<IndianPlayers

oddTeamPlayers={oddTeamPlayers}

evenTeamPlayers={evenTeamPlayers}

t20Players={t20Players}

ranjiPlayers={ranjiPlayers}

/>

)}

</div>

);

};

export default App;

**Step 3: Clean up default files**

You can delete or ignore:

* src/logo.svg
* Remove any leftover references to the default React logo in index.css or App.css.

**Step 4: Start the development server**

In your terminal inside the project directory:

npm start

This will open http://localhost:3000 in your browser and show the output.

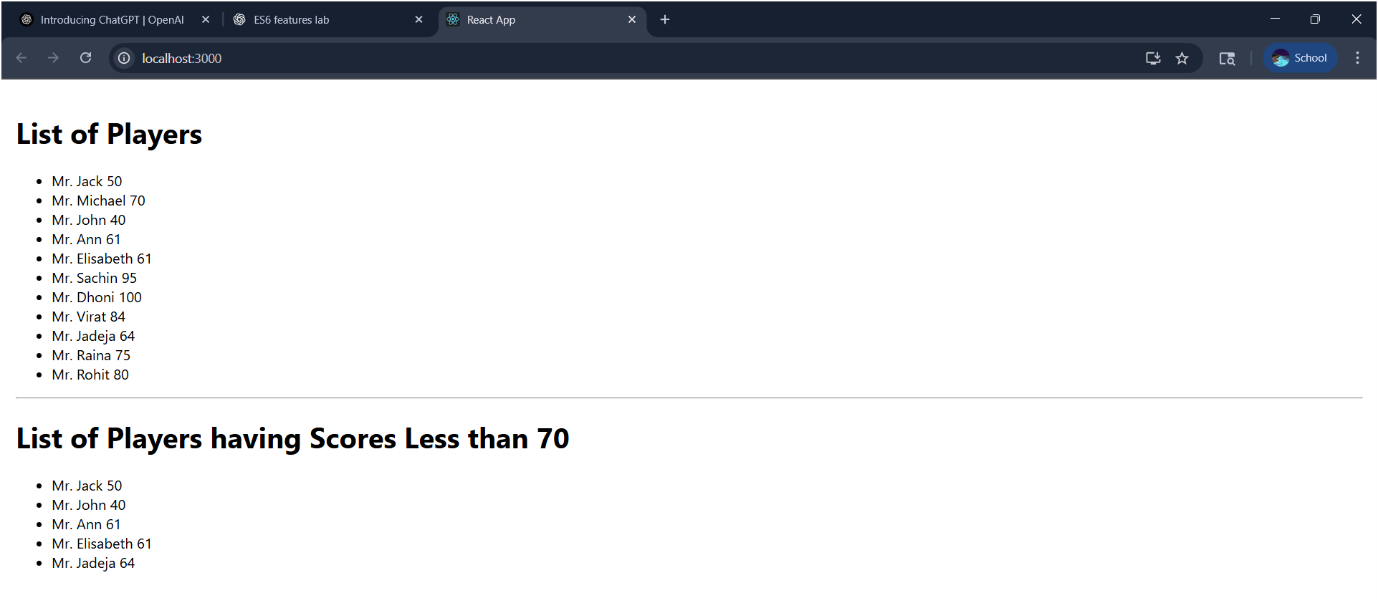
**Step 5: Toggle views**

const flag = false;

**Step 6: Explanation of ES6 usage in your app**

1. **map()**: Used to iterate over arrays to render <li> elements from players, oddTeamPlayers, evenTeamPlayers, and merged arrays.
2. **Arrow functions**:
   * players.filter(p => p.score < 70)
   * players.map((p, idx) => ...)
   * Used in .filter and .map callbacks for concise syntax.
3. **Destructuring**:
   * Component props: const ListofPlayers = ({ players }) => { ... }
   * Pulling odd/even players from indianTeam using array index logic.
4. **Spread operator**: [...t20Players, ...ranjiPlayers] merges two arrays.
5. **Conditional rendering**: flag === true ? <ListofPlayers .../> : <IndianPlayers .../>
6. **const and let**: All static data uses const to reflect immutability of bindings.

<http://localhost:3000/>



**Question : 02**

**Implementation of the “officespacerentalapp” lab with JSX, inline CSS, conditional coloring of rent, and looping through a list of office objects.**

**Step1:**

npx create-react-app officespacerentalapp

cd officespacerentalapp

**Step2:**

code .

**Step 3: Replace src/App.js with the following code**

// src/App.js

import React from "react";

// Single OfficeCard component

const OfficeCard = ({ office }) => {

// Conditional inline style for Rent

const rentStyle = {

color: office.Rent <= 60000 ? "red" : "green",

};

return (

<div style={{ border: "1px solid #ccc", padding: "12px", margin: "12px 0" }}>

{/\* JSX element for heading part of this card \*/}

<h2>{office.title}, at Affordable Range</h2>

{/\* Image attribute with JSX \*/}

<div>

<img

src={office.image}

width="200"

height="200"

alt={office.Name}

style={{ display: "block", marginBottom: "8px" }}

/>

</div>

{/\* Details object rendered \*/}

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

<h3 style={rentStyle}>Rent: Rs. {office.Rent}</h3>

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

</div>

);

};

const App = () => {

// JSX expression for page title

const pageTitle = "Office Space";

// List of office objects

const offices = [

{

title: pageTitle,

Name: "DBS",

Rent: 50000,

Address: "Chennai",

Image:”carimage.jpeg”

},

{

title: pageTitle,

Name: "TechHub",

Rent: 75000,

Address: "Bengaluru",

image: "carimage2.png",

},

{

title: pageTitle,

Name: "WorkNest",

Rent: 60000,

Address: "Hyderabad",

image: "carimage3.jpeg”

},

];

return (

<div style={{ padding: "24px", fontFamily: "Arial, sans-serif" }}>

{/\* JSX heading \*/}

<h1 style={{ fontSize: "36px", fontWeight: "bold" }}>

{pageTitle} , at Affordable Range

</h1>

{/\* Loop through list of office objects \*/}

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

<OfficeCard key={idx} office={office} />

))}

</div>

);

};

export default App;

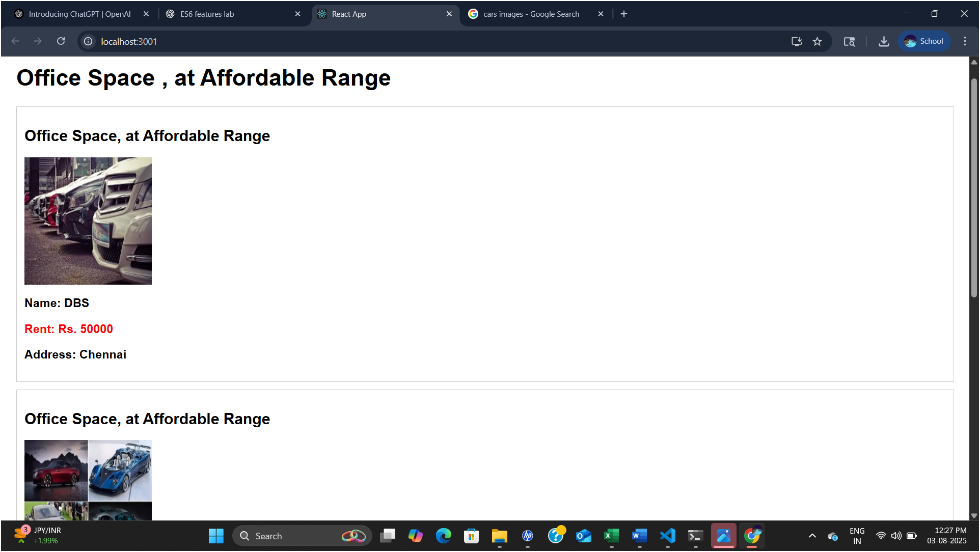
**Step 4: Explanation of requirements**

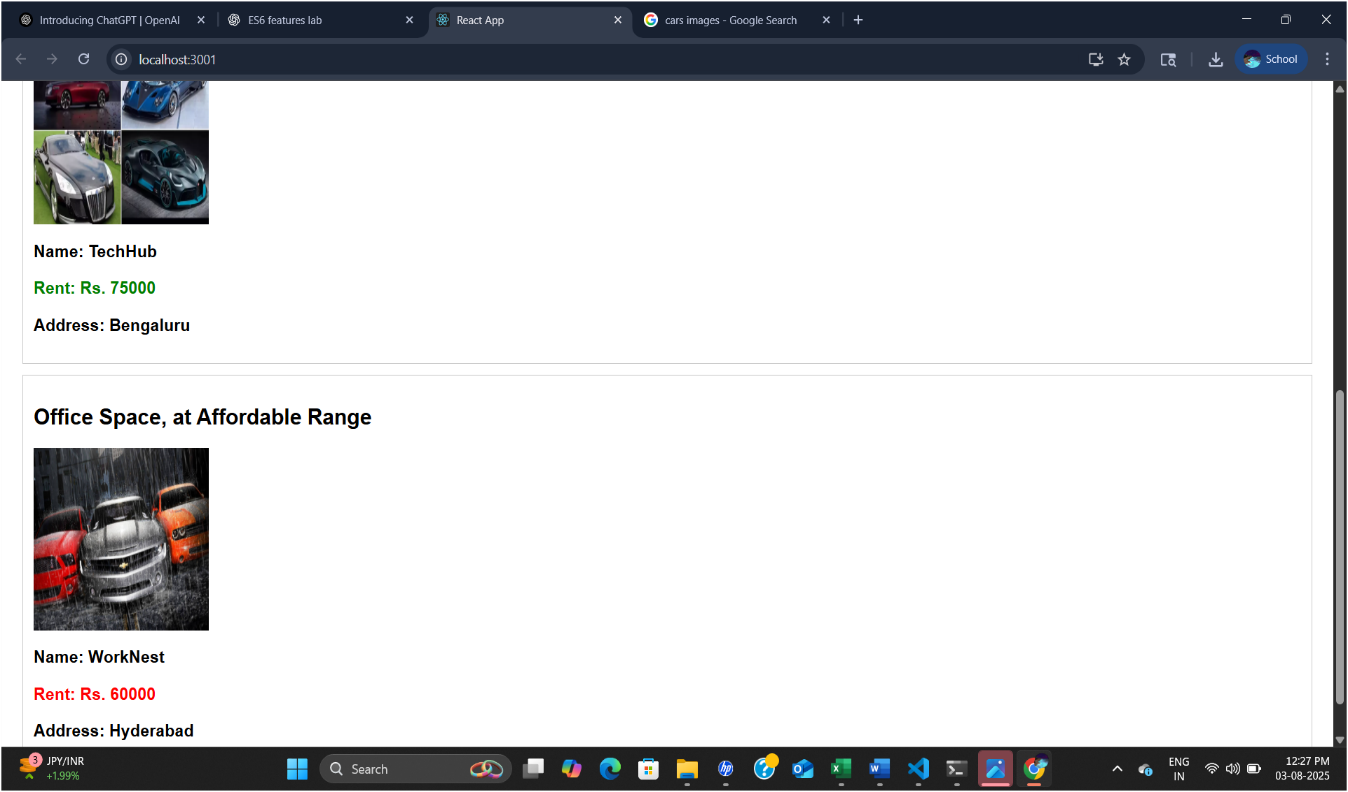
1. **JSX definition / usage**: The code uses JSX to define elements like <h1>, <img>, `<of<div>, and embeds JavaScript expressions inside {}(eg, `{pageTi{pageTitle}, {office.Name}).
2. **React.createElement**: JSX is syntactic sugar for React.createElement(...) under the hood.
3. **Creating React nodes with JSX**:OfficeCard, App) return JSX nodes.
4. **Rendering**: create-react-app'sesrc/index.jsalready<App /> into the DOM.
5. **JavaScript expressions in JSX**: Examples: {office.Rent <= 60000 ? "red" : "green"}(in{offices.map(...)} for iteration, and interpolation like {office.Address}.

**Step 5: Start the app**

npm start

<http://localhost:3001/>





**Question : 03**

**Eventexamplesapp**

Build a React app with:

* Counter with Increment (does two things), Decrement
* “Say welcome” button with argument
* Button showing “I was clicked” (synthetic event)
* Currency converter INR → Euro

**Step 1. Create the project**

Open terminal and run:

npx create-react-app eventexamplesapp

cd eventexamplesapp

**Step 2. Open in VS Code**

code .

**Step 3. src/App.js**

// src/App.js

import React, { useState } from "react";

// Counter block with multiple handlers

const Counter = () => {

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

const increment = () => setCount(c => c + 1);

const sayHello = () => alert("Hello! This is a static message.");

const handleIncrement = () => {

increment();

sayHello();

};

const handleDecrement = () => setCount(c => c - 1);

const welcome = message => alert(message);

return (

<div style={{ marginBottom: "20px" }}>

<div style={{ fontSize: "22px" }}>{count}</div>

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

<br />

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

<br />

<button onClick={() => welcome("welcome")}>Say welcome</button>

<br />

<button

onClick={e => {

// synthetic event

alert("I was clicked");

}}

>

Click on me

</button>

</div>

);

};

// Currency converter INR to Euro

const CurrencyConverter = () => {

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

const [converted, setConverted] = useState(null);

const rate = 90; // 1 Euro = 90 INR

const handleSubmit = e => {

e.preventDefault(); // synthetic event usage

const inr = parseFloat(amount);

if (isNaN(inr)) {

alert("Please enter a valid number");

return;

}

const euro = inr / rate;

setConverted(euro.toFixed(2) + " €");

};

return (

<div>

<h2 style={{ color: "green" }}>Currency Convertor!!!</h2>

<form onSubmit={handleSubmit}>

<div>

<label>

Amount (INR):{" "}

<input

type="text"

value={amount}

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

/>

</label>

</div>

<div>

<label>

Currency: <input type="text" value="Euro" readOnly />

</label>

</div>

<div>

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

</div>

</form>

{converted && (

<div style={{ marginTop: "8px" }}>

<strong>Converted:</strong> {converted}

</div>

)}

</div>

);

};

// Root app

const App = () => (

<div style={{ padding: "20px", fontFamily: "sans-serif" }}>

<Counter />

<CurrencyConverter />

</div>

);

export default App;

**Step 4. Start the app**

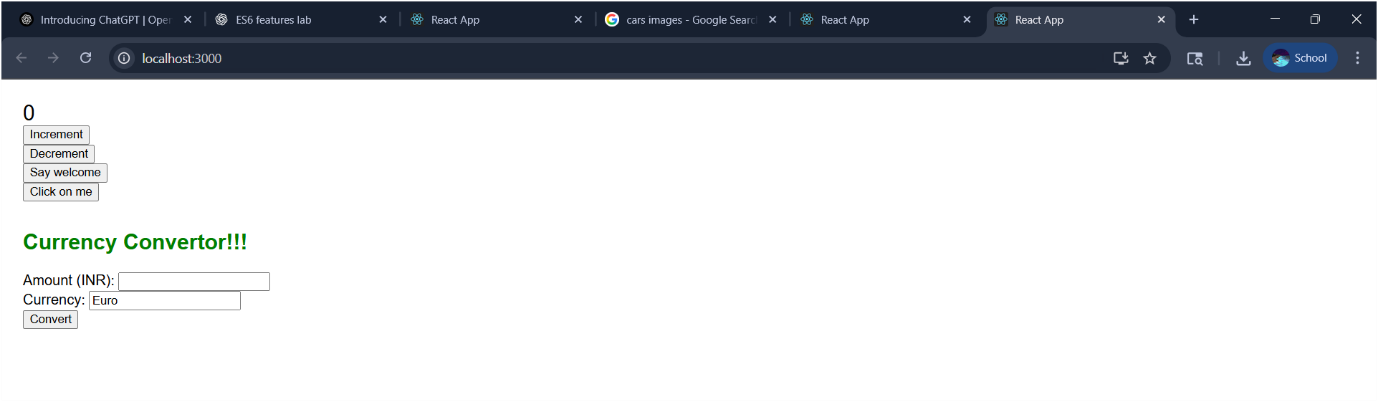
In terminal:

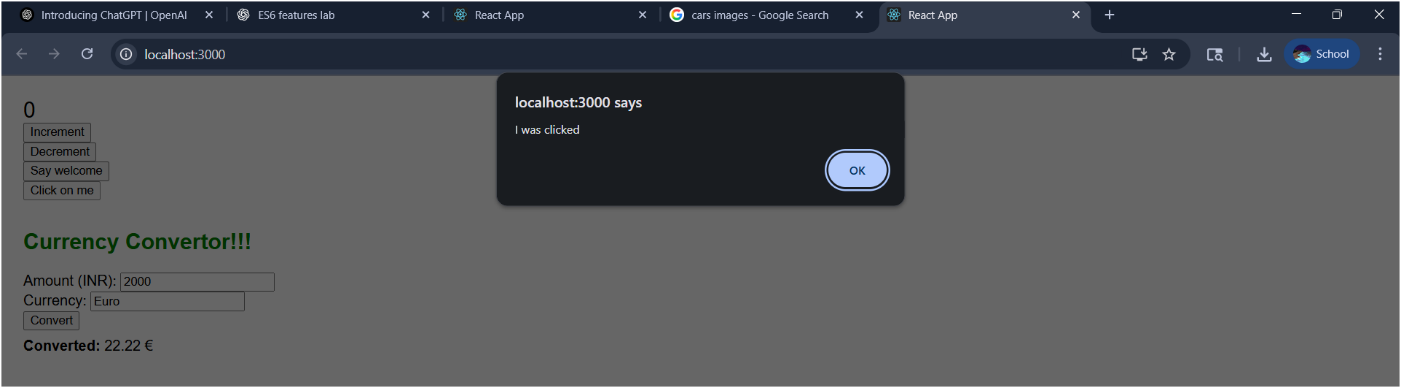
npm start

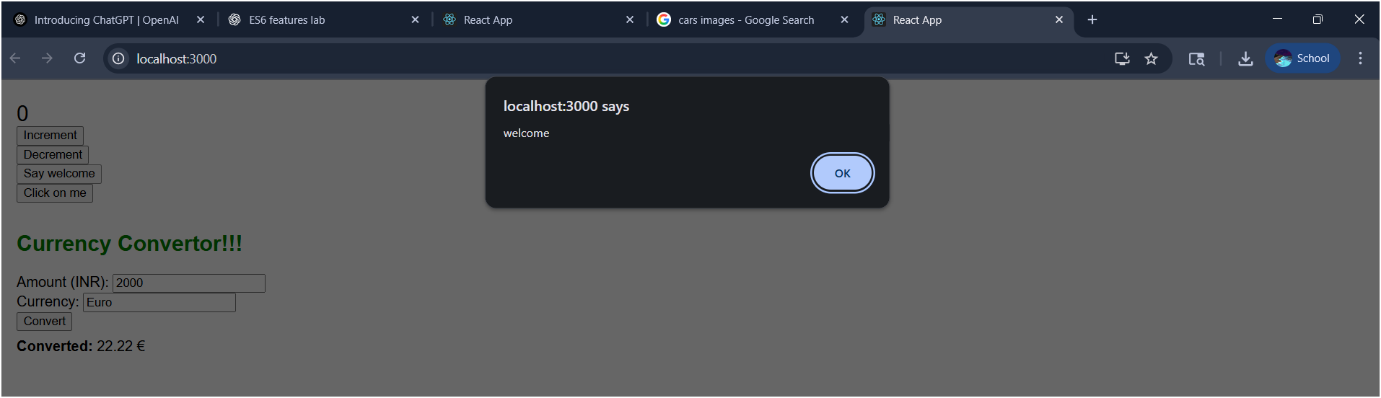
It opens [http://localhost:3000](http://localhost:3000%20) automatically.

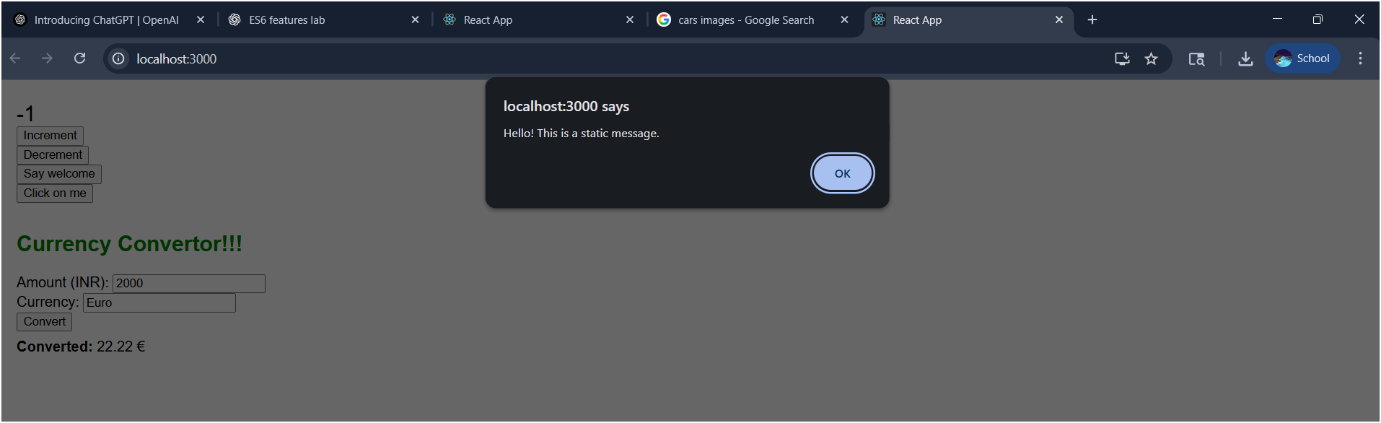
**Step 5. Test behavior**

* Click **Increment**: counter increases and shows “Hello!” alert.
* Click **Decrement**: counter decreases.
* Click **Say welcome**: alert shows “welcome”.
* Click **Click on me**: alert shows “I was clicked”.
* In **Currency Convertor**, enter INR and press Convert → shows Euro equivalent.







****

**Definitions**

1. **React event naming**: camelCase (onClick, onChange).
2. **Event handler**: function assigned to an event prop (e.g., onClick={handleIncrement}).
3. **SyntheticEvent**: React’s wrapper around native events (e.preventDefault() used in form submission).
4. **Passing arguments**: use arrow function: () => welcome("welcome").
5. **Multiple actions in one handler**: handleIncrement calls increment() and

**Question :04**

React app using **conditional rendering** to show either a **guest view** or a **user view** based on login status **“ticketbookingapp”**

**Conditional Rendering in React**

React allows you to conditionally render components using JavaScript logic:

{isLoggedIn ? <UserPage /> : <GuestPage />}

In your example, this is done in the Greeting component:

function Greeting(props) {

const isLoggedIn = props.isLoggedIn;

if (isLoggedIn) {

return <UserGreeting />;

}

return <GuestGreeting />;

}

**Element Variables**

Element variables can be used to store JSX and render dynamically:

let button;

if (isLoggedIn) {

button = <LogoutButton onClick={handleLogoutClick} />;

} else {

button = <LoginButton onClick={handleLoginClick} />;

}

**Prevent Rendering**

Return null from a component if you don’t want it to render:

function WarningBanner(props) {

if (!props.warn) {

return null;

}

return <div className="warning">Warning!</div>;

}

**Ticketbookingapp**

**1. App Setup**

In terminal:

npx create-react-app ticketbookingapp

cd ticketbookingapp

**2. File Structure**

Create components:

* LoginButton.js
* LogoutButton.js
* Greeting.js
* UserGreeting.js (says "Welcome back")
* GuestGreeting.js (says "Please sign up")
* FlightDetails.js (visible to everyone)
* TicketBooking.js (only visible when logged in)

**3. App Component Code**

App.js

import React, { useState } from 'react';

import Greeting from './Greeting';

import LoginButton from './LoginButton';

import LogoutButton from './LogoutButton';

import FlightDetails from './FlightDetails';

import TicketBooking from './TicketBooking';

function App() {

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

const handleLoginClick = () => setIsLoggedIn(true);

const handleLogoutClick = () => setIsLoggedIn(false);

let button;

if (isLoggedIn) {

button = <LogoutButton onClick={handleLogoutClick} />;

} else {

button = <LoginButton onClick={handleLoginClick} />;

}

return (

<div>

<Greeting isLoggedIn={isLoggedIn} />

{button}

<FlightDetails />

{isLoggedIn && <TicketBooking />}

</div>

);

}

export default App;

**4. Components**

**LoginButton.js**

function LoginButton(props) {

return (

<button onClick={props.onClick}>

Login

</button>

);

}

export default LoginButton;

**LogoutButton.js**

function LogoutButton(props) {

return (

<button onClick={props.onClick}>

Logout

</button>

);

}

export default LogoutButton;

**Greeting.js**

import UserGreeting from './UserGreeting';

import GuestGreeting from './GuestGreeting';

function Greeting(props) {

const isLoggedIn = props.isLoggedIn;

return isLoggedIn ? <UserGreeting /> : <GuestGreeting />;

}

export default Greeting;

**UserGreeting.js**

function UserGreeting() {

return <h1>Welcome back</h1>;

}

export default UserGreeting;

**GuestGreeting.js**

function GuestGreeting() {

return <h1>Please sign up.</h1>;

}

export default GuestGreeting;

**FlightDetails.js**

function FlightDetails() {

return <p>Flight info: Delhi to Mumbai, 6:00 PM, Rs. 4500</p>;

}

export default FlightDetails;

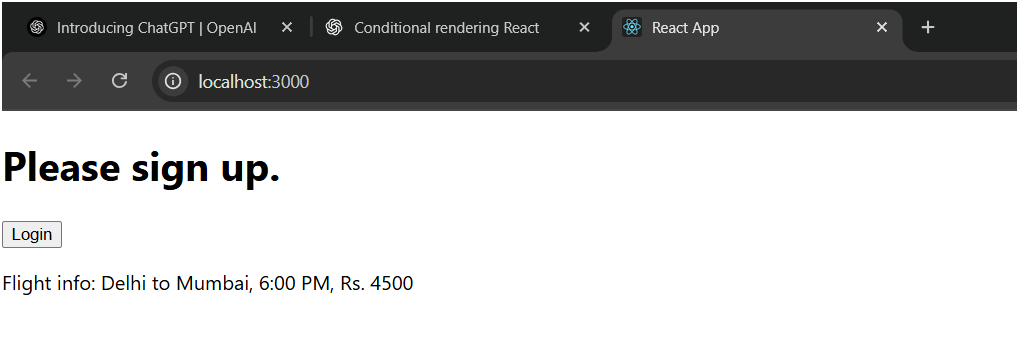
**TicketBooking.js**

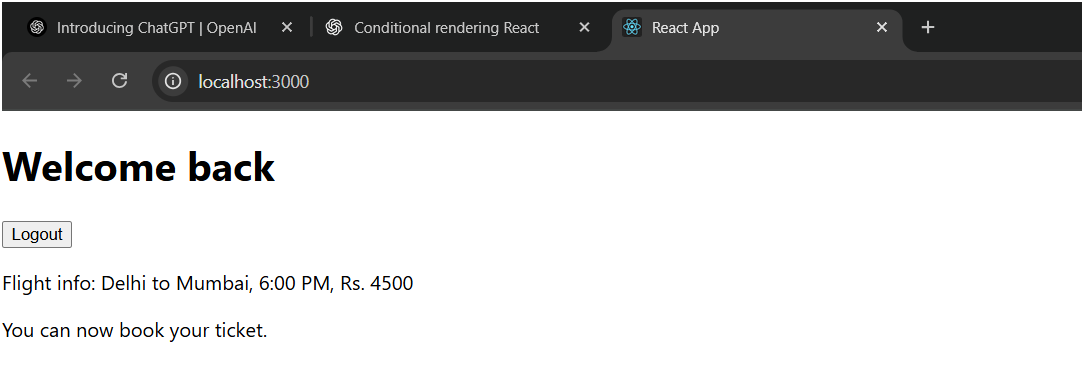
function TicketBooking() {

return <p>You can now book your ticket.</p>;

}

export default TicketBooking;





**Question : 05**

**"Bloggerapp"** React project

**Project: bloggerapp**

This app has **3 components**:

1. Course Details
2. Book Details
3. Blog Details

Each displays a list with conditional rendering wherever applicable.

**1. App.js**

import React from 'react';

import BookDetails from './BookDetails';

import BlogDetails from './BlogDetails';

import CourseDetails from './CourseDetails';

function App() {

const showCourses = true;

const showBooks = true;

const showBlogs = true;

return (

<div style={{ display: 'flex', justifyContent: 'space-around', marginTop: '40px' }}>

{showCourses && <CourseDetails />}

{showBooks ? <BookDetails /> : <p>No Books Found</p>}

{showBlogs && <BlogDetails />}

</div>

);

export default App;

**2. CourseDetails.js**

function CourseDetails() {

const courses = [

{ id: 1, name: 'Angular', date: '4/5/2021' },

{ id: 2, name: 'React', date: '6/3/2021' },

];

return (

<div style={{ borderLeft: '4px solid green', padding: '0 20px' }}>

<h2>Course Details</h2>

{courses.map(course => (

<div key={course.id}>

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

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

</div>

))}

</div>

);

}

export default CourseDetails;

**3. BookDetails.js**

function BookDetails() {

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 },

];

const bookdet = (

<ul>

{books.map(book => (

<div key={book.id}>

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

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

</div>

))}

</ul>

);

return (

<div style={{ borderLeft: '4px solid green', padding: '0 20px' }}>

<h2>Book Details</h2>

{bookdet}

</div>

);

}

export default BookDetails;

**4. BlogDetails.js**

function BlogDetails() {

const blogs = [

{

id: 1,

title: 'React Learning',

author: 'Stephen Biz',

content: 'Welcome to learning React!'

},

{

id: 2,

title: 'Installation',

author: 'Schwezdenier',

content: 'You can install React from npm.'

},

];

return (

<div style={{ borderLeft: '4px solid green', padding: '0 20px' }}>

<h2>Blog Details</h2>

{blogs.map(blog => (

<div key={blog.id}>

<h3>{blog.title}</h3>

<h5><strong>{blog.author}</strong></h5>

<p>{blog.content}</p>

</div>

))}

</div>

);

}

export default BlogDetails;

**Conditional Rendering Used:**

* && short-circuit ({showCourses && <CourseDetails />})
* Ternary operator ({showBooks ? <BookDetails /> : <p>No Books Found</p>})
* Separate constant JSX (bookdet)

