**WEEK-7**

**React JS Handson -9**

• List the features of ES6

ES6 introduced modern features to JavaScript like:

1. let and const for better variable scoping
2. Arrow functions for simpler function syntax
3. Classes and inheritance
4. Template literals using backticks (`)
5. Default parameters in functions
6. Destructuring of arrays and objects
7. Spread and rest operators (...)
8. Promises for asynchronous code
9. Modules (import/export) for better structure
10. Set and Map objects

• Explain JavaScript let

The let keyword is used to declare variables that are block-scoped. This means the variable is only accessible within the block where it is declared, like inside a loop or function. It also prevents redeclaration within the same scope.

• Identify the differences between var and let

1. var is function-scoped, while let is block-scoped.
2. var allows re-declaration in the same scope, but let does not.
3. let helps prevent bugs caused by accidental overwriting of variables.

• Explain JavaScript const

const is used to declare constants – variables that cannot be reassigned once given a value. Like let, it is block-scoped. However, objects or arrays declared with const can still have their contents modified.

• Explain ES6 class fundamentals

ES6 introduced a cleaner way to create classes using the class keyword. A class is a template for creating objects with methods and properties.  
Example:

class Student {

constructor(name) {

this.name = name;

}

greet() {

return `Hello, ${this.name}`;

}

}

• Explain ES6 class inheritance

ES6 allows one class to inherit from another using the extends keyword. The child class uses super() to call the parent’s constructor.  
Example:

class Person {

constructor(name) {

this.name = name;

}

}

class Doctor extends Person {

constructor(name, dept) {

super(name);

this.dept = dept;

}

}

• Define ES6 arrow functions

Arrow functions (=>) are a shorter syntax for writing functions.  
Example:

const add = (a, b) => a + b;

They don't have their own this context, so making them useful inside callbacks.

• Identify set(), map()

* Set is a collection of unique values – it automatically removes duplicates.
* Map stores key-value pairs, and unlike objects, keys can be of any type (not just strings).
* **App.js**

import React from 'react';

import OddPlayers from './components/OddPlayers';

import EvenPlayers from './components/EvenPlayers';

import ListofPlayers from './components/ListofPlayers';

function App() {

  const flag = false;

  const players = [

    { name: 'Virat Kohli', score: 82 },

  { name: 'Rohit Sharma', score: 65 },

  { name: 'KL Rahul', score: 73 },

  { name: 'Hardik Pandya', score: 50 },

  { name: 'Jadeja', score: 74 },

  { name: 'Ashwin', score: 60 },

  { name: 'Shami', score: 90 },

  { name: 'Bumrah', score: 40 },

  { name: 'Surya Kumar', score: 88 },

  { name: 'Ishan Kishan', score: 67 },

  { name: 'Gill', score: 95 }

  ];

  const players70 = players.filter(player => player.score <= 70);

  const IndianTeam = [

    "Sachin",

    "Dhoni",

    "Virat",

    "Rohit",

    "Yuvraj",

    "Raina"

  ];

  const T20Players = ['Mr. First Player', 'Mr. Second Player', 'Mr. Third Player'];

  const RanjiTrophyPlayers = ['Mr. Fourth Player', 'Mr. Fifth Player', 'Mr. Sixth Player'];

  const IndianPlayers = [...T20Players, ...RanjiTrophyPlayers];

  if (flag === true) {

    return (

      <div>

        <h1>List of Players</h1>

        <ListofPlayers />

        <hr />

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

        <ul>

          {players70.map((item, index) => (

            <li key={index}>

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

            </li>

          ))}

        </ul>

      </div>

    );

  } else {

    return (

      <div>

        <h2>Odd Players</h2>

        <OddPlayers team={IndianTeam} />

        <hr />

        <h2>Even Players</h2>

        <EvenPlayers team={IndianTeam} />

        <hr />

        <h2>List of Indian Players Merged:</h2>

        <ul>

          {IndianPlayers.map((player, index) => (

            <li key={index}>{player}</li>

          ))}

        </ul>

      </div>

    );

  }

}

export default App;

* **EvenPlayers.js**

import React from 'react';

function EvenPlayers({ team }) {

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

  return (

    <ul>

      <li>Second : {second}</li>

      <li>Fourth : {fourth}</li>

      <li>Sixth : {sixth}</li>

    </ul>

  );

}

export default EvenPlayers;

* **OddPlayers.js**

import React from 'react';

function OddPlayers({ team }) {

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

  return (

    <ul>

      <li>First : {first}</li>

      <li>Third : {third}</li>

      <li>Fifth : {fifth}</li>

    </ul>

  );

}

export default OddPlayers;

* **ListofPlayers.js**

import React from 'react';

const players = [

  { name: 'Virat Kohli', score: 82 },

  { name: 'Rohit Sharma', score: 65 },

  { name: 'KL Rahul', score: 73 },

  { name: 'Hardik Pandya', score: 50 },

  { name: 'Jadeja', score: 74 },

  { name: 'Ashwin', score: 60 },

  { name: 'Shami', score: 90 },

  { name: 'Bumrah', score: 40 },

  { name: 'Surya Kumar', score: 88 },

  { name: 'Ishan Kishan', score: 67 },

  { name: 'Gill', score: 95 }

];

const ListofPlayers = () => {

  return (

    <ul>

      {players.map((item, index) => (

        <li key={index}>

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

        </li>

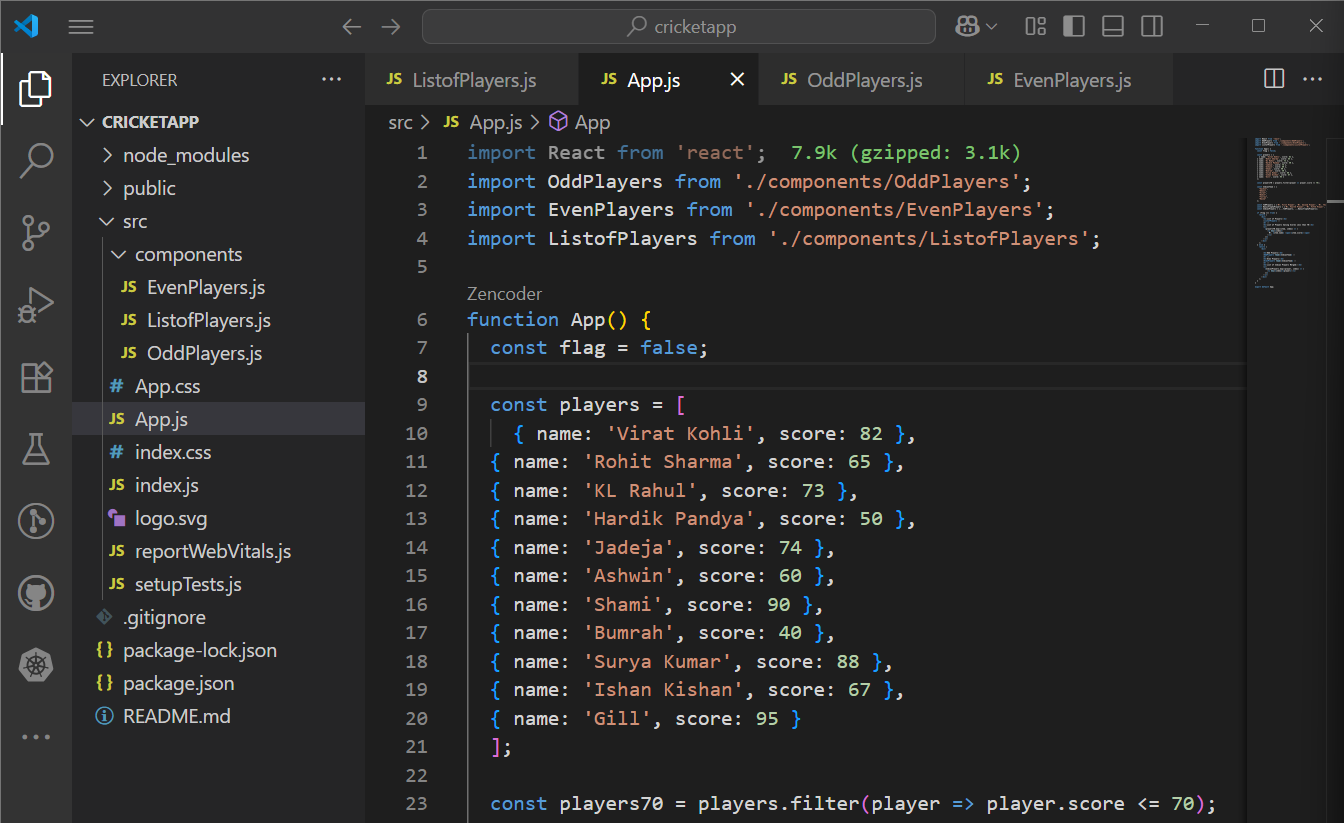
      ))}

    </ul>

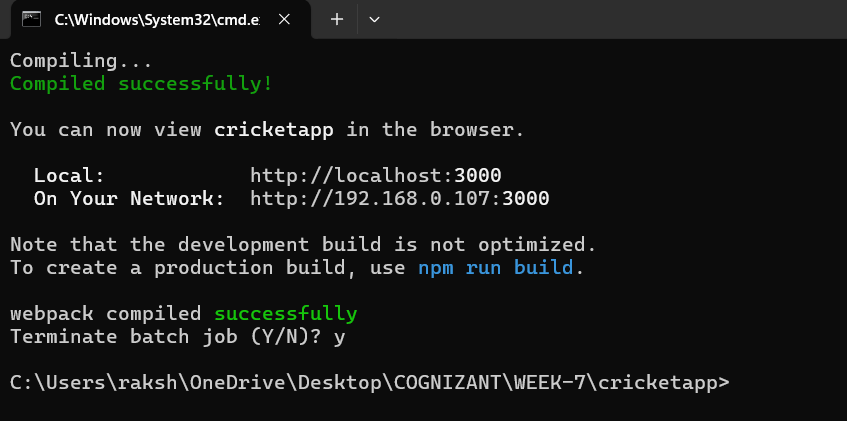
  );

};

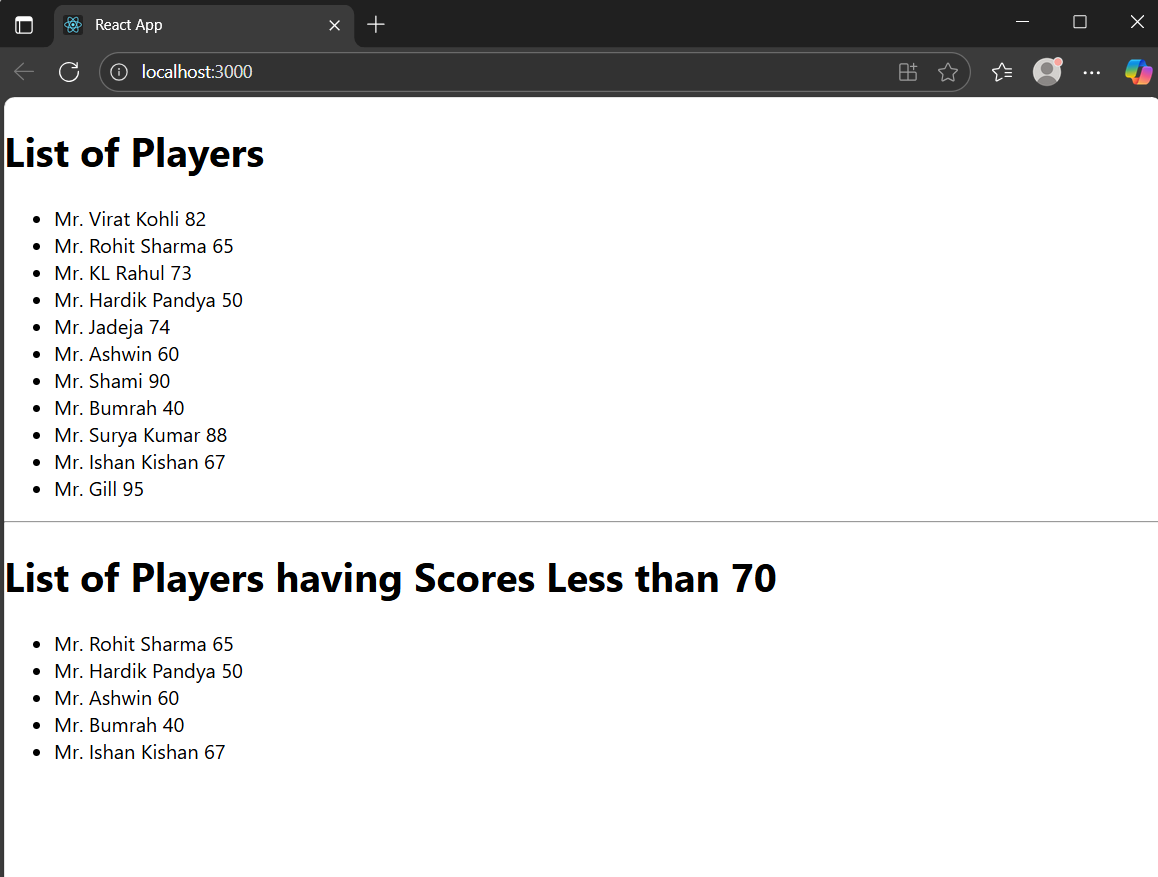
export default ListofPlayers;



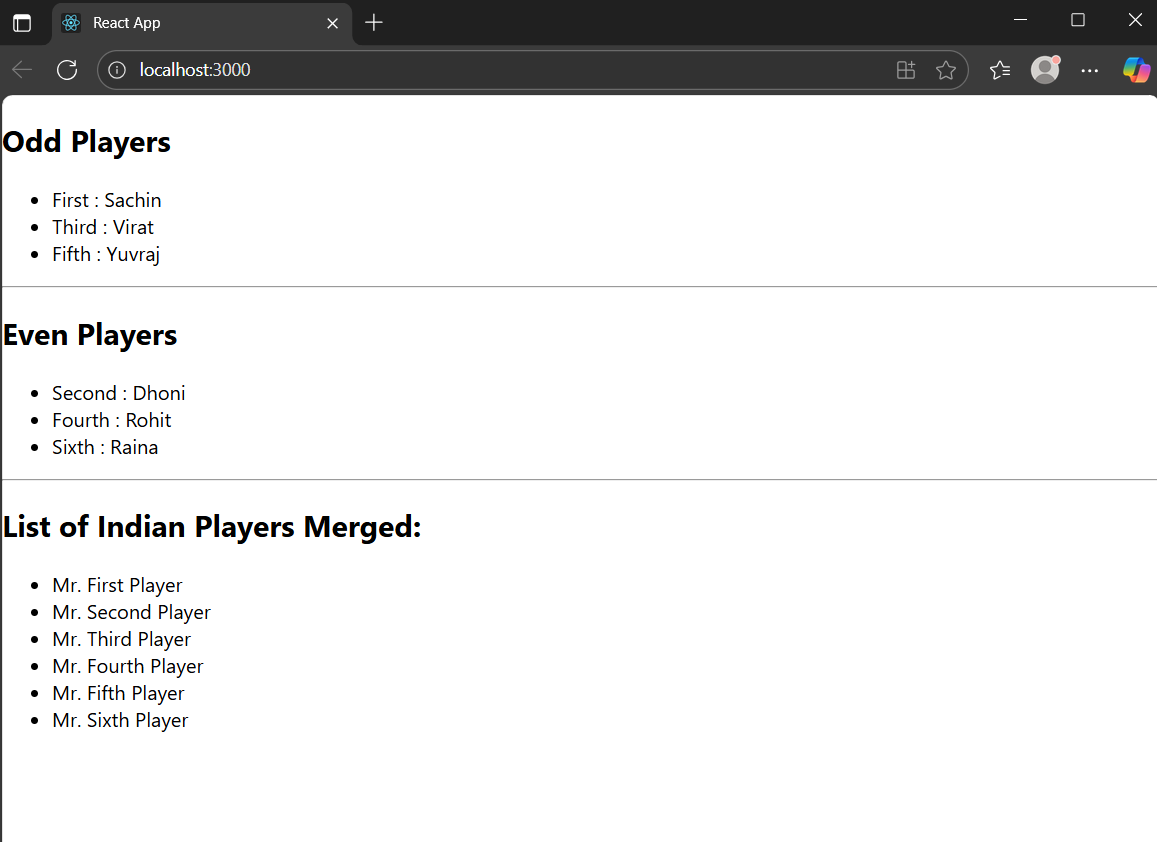
**Output:**



If flag=true



If flag=false



**Explanation:**

1. Created Multiple Components:  
   We created three separate React components — ListofPlayers, OddPlayers, and EvenPlayers — to display player data in different ways using reusable and modular code.
2. Used ES6 Features:  
   We used ES6 features like map() to iterate over arrays, filter() (or conditional logic) to extract players with scores less than 70, array destructuring to split odd/even players, and the spread operator (...) to merge arrays.
3. Conditional Rendering with Flag:  
   A flag variable was used to conditionally render either the list of players or the Indian team split into odd/even, depending on its value. This helped demonstrate simple conditional logic in JSX.
4. Props for Data Passing:  
   Data like player names and scores were passed to components using props, allowing each component to access and render dynamic information.
5. Final Output Rendered in Browser:  
   Based on the flag value, the app displays either the full player list and scores below 70, or the odd/even players and a merged list of Indian players — showcasing various ways to handle and display data in React.