**ReactJS – HOL 9**

**Questions & Answers –**

1. **List the features of ES6**

**Ans -** ES6 introduced many new features to JavaScript to make the language more powerful and readable. Key features include:

* let and const for block-scoped variable declarations
* Arrow functions (=>) for shorter function syntax
* Classes for object-oriented programming
* Default parameters in functions
* Template literals using backticks (`)
* Destructuring assignment for arrays and objects
* Spread (...) and rest operators
* Promises for handling asynchronous code
* Modules (import and export)
* Enhanced object literals
* New data structures: Map and Set
* For-of loop for iterating over iterable objects

1. **Explain JavaScript let**

**Ans -** *let* is a keyword introduced in ES6 to declare variables. It is block-scoped, meaning it is only accessible within the block in which it is defined. Unlike var, it doesn't allow redeclaration in the same scope.

1. **Identify the differences between var and let**

**Ans –**

| **Feature** | **var** | **let** |
| --- | --- | --- |
| **Scope** | Function-scoped | Block-scoped |
| **Hoisting** | Hoisted (initialized as undefined) | Hoisted but not initialized |
| **Redeclaration** | Allowed within the same scope | Not allowed in the same scope |
| **Use Case** | Legacy JavaScript | Recommended for modern usage |

1. **Explain JavaScript const**

**Ans -** *const* is used to declare constants—variables whose values cannot be reassigned. Like *let*, *const* is block-scoped. However, for objects and arrays, *const* prevents reassignment but not modification of contents.

1. **Explain ES6 class fundamentals**

**Ans -** ES6 introduced the class keyword for defining classes in a cleaner, more OOP-like syntax. A class can contain a constructor and methods.

**Example –**

class Person {

constructor(name) {

this.name = name;

}

greet() {

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

}

}

1. **Explain ES6 class inheritance**

**Ans -** ES6 allows classes to inherit from other classes using the extends keyword. The super() method is used to call the parent class's constructor.

**Example –**

class Animal {

constructor(name) {

this.name = name;

}

}

class Dog extends Animal {

bark() {

return `${this.name} says woof!`;

}

}

1. **Define ES6 arrow functions**

**Ans -** Arrow functions provide a shorter syntax for writing functions. They do not have their own *this*, making them useful for callbacks and functional programming.

**Example –**

// Traditional function

function add(a, b) {

return a + b;

}

// Arrow function

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

1. **Identify set(), map()**

**Ans –**

**Set():**

* A collection of **unique values.**
* Does not allow duplicates.
* Useful for storing non-repeating items.

**Example –**

const set = new Set([1, 2, 2, 3]);

console.log(set); // Set(3) {1, 2, 3}

**Map()**:

* A collection of **key-value pairs**.
* Keys can be of **any type** (not just strings).
* Maintains the **insertion order**.

**Example –**

const map = new Map();

map.set('name', 'Alice');

map.set(1, 'one');

console.log(map.get('name')); // Alice

**Hands-On – (Code)**

*ListOfPlayers.js –*

const ListofPlayers = () => {

  const players = [

    { name: "Virat Kohli", score: 87 },

    { name: "Rohit Sharma", score: 76 },

    { name: "Shubman Gill", score: 90 },

    { name: "KL Rahul", score: 101 },

    { name: "Suryakumar Yadav", score: 64 },

    { name: "Hardik Pandya", score: 39 },

    { name: "Ravindra Jadeja", score: 25 },

    { name: "Kuldeep Yadav", score: 10 },

    { name: "Mohammed Shami", score: 5 },

    { name: "Jasprit Bumrah", score: 8 },

    { name: "Mohammed Siraj", score: 3 }

  ];

const oddTeam = players

  .map((player, index) => ({ ...player, originalIndex: index }))

  .filter((\_, index) => index % 2 === 0);

const evenTeam = players

  .map((player, index) => ({ ...player, originalIndex: index }))

  .filter((\_, index) => index % 2 !== 0);

  const ordinals = [

  "First", "Second", "Third", "Fourth", "Fifth",

  "Sixth", "Seventh", "Eighth", "Ninth", "Tenth", "Eleventh"

];

  var flag = false;

  if(flag === true){

        return(

            <div>

                <h2>List of players</h2>

                <ul>

                    {players.map((item) => (

                        <li>

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

                        </li>

                    ))}

                </ul>

                <h2>List of players having score less than 70</h2>

                <ul>

                    {players.filter(player => player.score < 70)

                    .map((item) => (

                        <li>

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

                        </li>

                    ))}

                </ul>

            </div>

        );

    } else{

        return(

            <div>

                <h2>Odd Players</h2>

                <ul>

                    {oddTeam.map(({name, score, originalIndex}, index) => (

                        <li key={index}>

                            <span>{ordinals[originalIndex]}: </span>Mr. <span>{name}</span> - <span>{score}</span>

                        </li>

                    ))}

                </ul>

                <h2>Even Players</h2>

                <ul>

                    {evenTeam.map(({name, score, originalIndex}, index) =>  (

                        <li key={index}>

                            <span>{ordinals[originalIndex]}: </span>Mr. <span>{name}</span> - <span>{score}</span>

                        </li>

                    ))}

                </ul>

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

                <ul>

                    {players.map(({name}, index) => (

                        <li key={index}>

                            <span>{ordinals[index]}: </span>Mr. <span>{name}</span>

                        </li>

                    ))}

                </ul>

            </div>

        );

    }

};

export default ListofPlayers;

*App.js –*

import './App.css';

import ListofPlayers from './components/ListOfPlayers';

function App() {

  return (

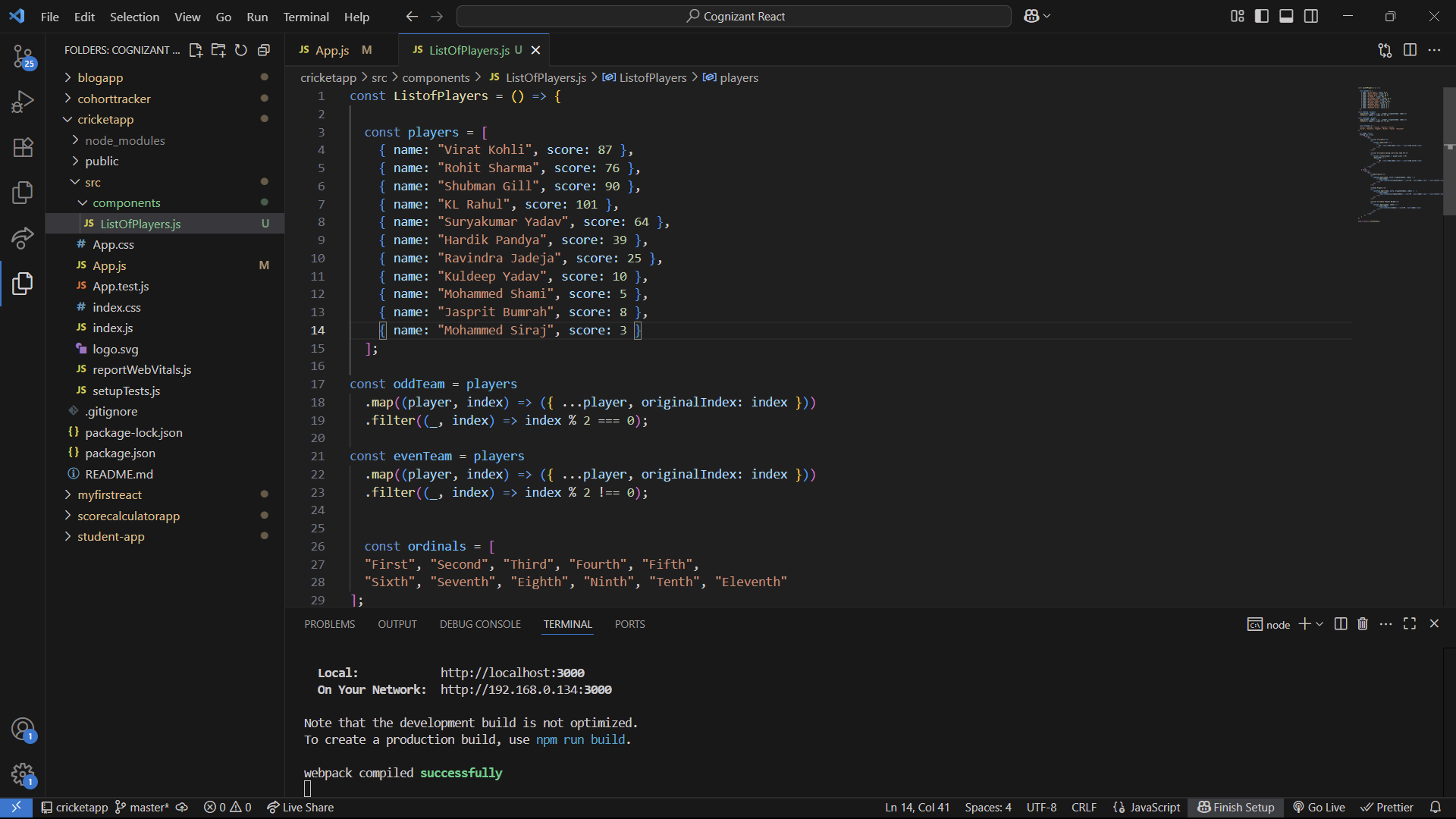
    <ListofPlayers />

  );

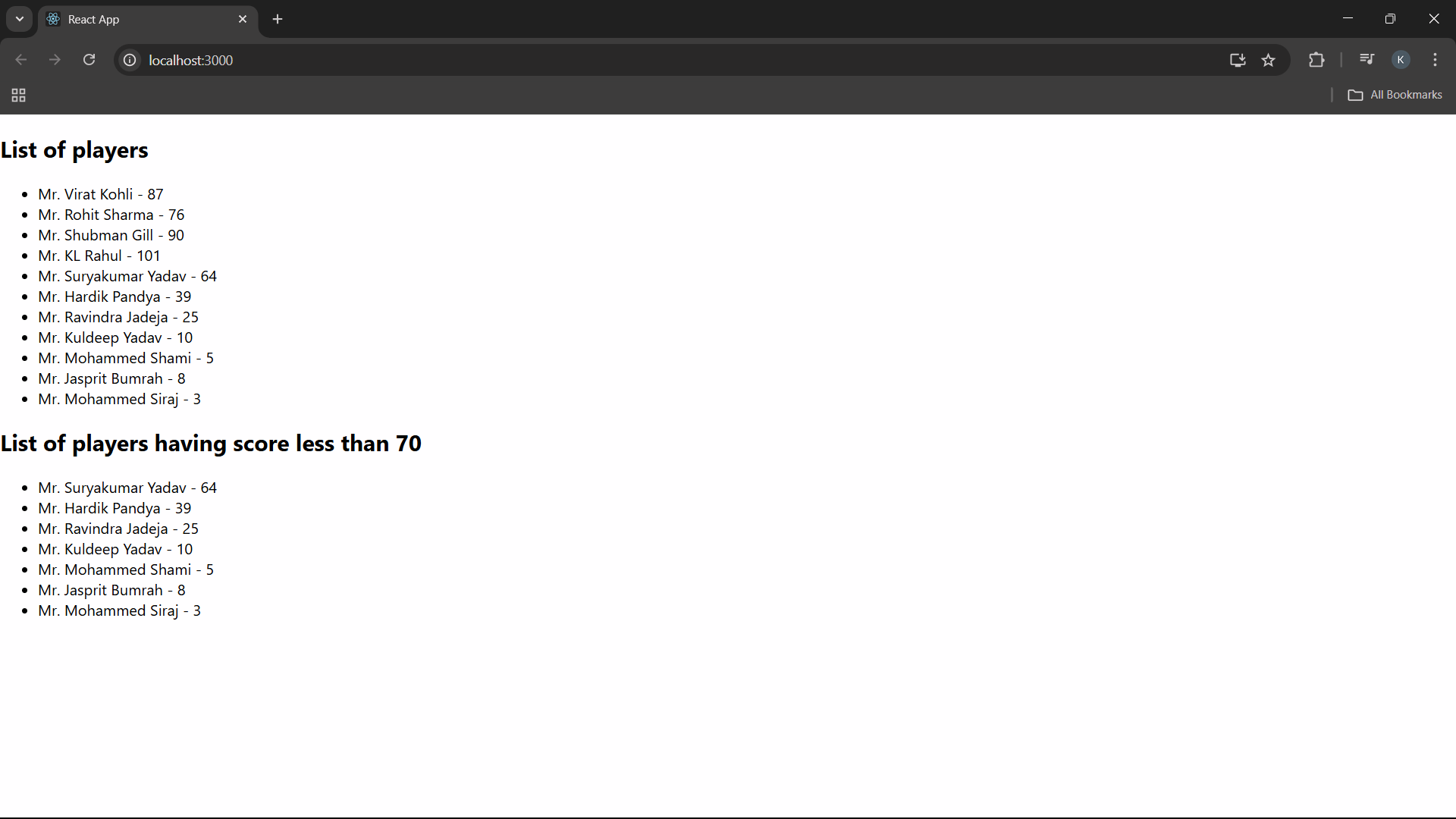
}

export default App;

**Output –**



**When flag is true –**



**When flag is false –**

