## **Objectives**

* List the features of ES6

|  |  |
| --- | --- |
| Feature | Description |
| let | Declares block-scoped variables, prevents redeclaration |
| const | Declares block-scoped constants, must be initialized |
| Arrow Functions | Simplified function syntax, lexical 'this' binding |
| Classes | Introduces class-based OOP syntax |
| Class Inheritance | Extends functionality from parent class using `extends` and `super` |
| Template Literals | Allows multi-line strings and expression interpolation using backticks |
| Destructuring | Unpacks values from arrays or objects into variables |
| Modules | Supports import/export of variables, functions, classes from other files |
| Default Parameters | Allows setting default values for function parameters |
| Rest/Spread Operators | Handle variable arguments and array/object expansion |
| Promises | Handle asynchronous operations more cleanly than callbacks |
| Map & Set | New data structures for unique elements and key-value pairs |

* Explain JavaScript let

let is used to declare a variable in JavaScript. A variable is like a container that holds some value, such as a number or text.

**Features of let:**

* The value of a let variable can change.
* It only works inside the block where it is written (this is called block scope).
* You can’t declare the same let variable more than once in the same block.
* Identify the differences between var and let

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Keyword | Scope | Hoisting | Re-declaration | Usage |
| var | Function-scoped | Hoisted (undefined) | Allowed | Legacy, avoid in modern JS |
| let | Block-scoped | Hoisted (TDZ applies) | Not allowed in same scope | Preferred for mutable values |

* Explain JavaScript const

const is used to declare a variable that cannot be changed once it is assigned. It stands for constant.

* You must give it a value when you declare it.
* You cannot reassign it later.
* It also has block scope, just like let.
* Explain ES6 class fundamentals

Classes are templates for creating objects. They group together properties (variables) and methods (functions) in one structure.

**Example:**

class Car {

constructor(brand) {

this.brand = brand;

}

show() {

console.log("Car brand is " + this.brand);

}

}

let myCar = new Car("Toyota");

myCar.show(); // Output: Car brand is Toyota

Classes make it easier to create and manage multiple similar objects.

* Explain ES6 class inheritance

Inheritance means one class can get features from another class. The main class is called the parent (or base class), and the new class is the child (or derived class).

Use extends keyword and super() to inherit.

**Example:**

class Animal {

speak() {

console.log("Animal speaks");

}

}

class Dog extends Animal {

speak() {

super.speak(); // Calls parent class method

console.log("Dog barks");

}

}

let pet = new Dog();

pet.speak();

// Output:

// Animal speaks

// Dog barks

* Define ES6 arrow functions

Arrow functions are a **shorter way** to write functions in JavaScript.

**Syntax:**

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

console.log(add(2, 3)); // Output: 5

* Arrow functions don’t have their own this.
* They are mostly used in short and simple functions.
* Identify set(), map()

A Set is a collection of unique values (no duplicates allowed). It automatically removes duplicates.

**Example:**

let mySet = new Set();

mySet.add(1);

mySet.add(2);

mySet.add(2); // Duplicate, will not be added

console.log(mySet); // Output: Set {1, 2}

Use Set when you want to store unique values only.

**Map**

A Map stores key-value pairs. Unlike objects, the keys in a Map can be any type (not just strings).

**Example:**

let myMap = new Map();

myMap.set("name", "Sree");

myMap.set(1, "One");

console.log(myMap.get("name")); // Output: Sree

Use Map when you need key-value pairs with flexible key types.

## **Notes**

Estimated time to complete this lab: **60 minutes.**

Create a React Application named “cricketapp” with the following components:

1. ListofPlayers

* Declare an array with 11 players and store details of their names and scores using the map feature of ES6



* Filter the players with scores below 70 using arrow functions of ES6.



1. IndianPlayers
   1. Display the Odd Team Player and Even Team players using the Destructuring features of ES6



* 1. Declare two arrays T20players and RanjiTrophy players and merge the two arrays and display them using the Merge feature of ES6



Display these two components in the same home page using a simple if else in the flag variable.

**Output:**

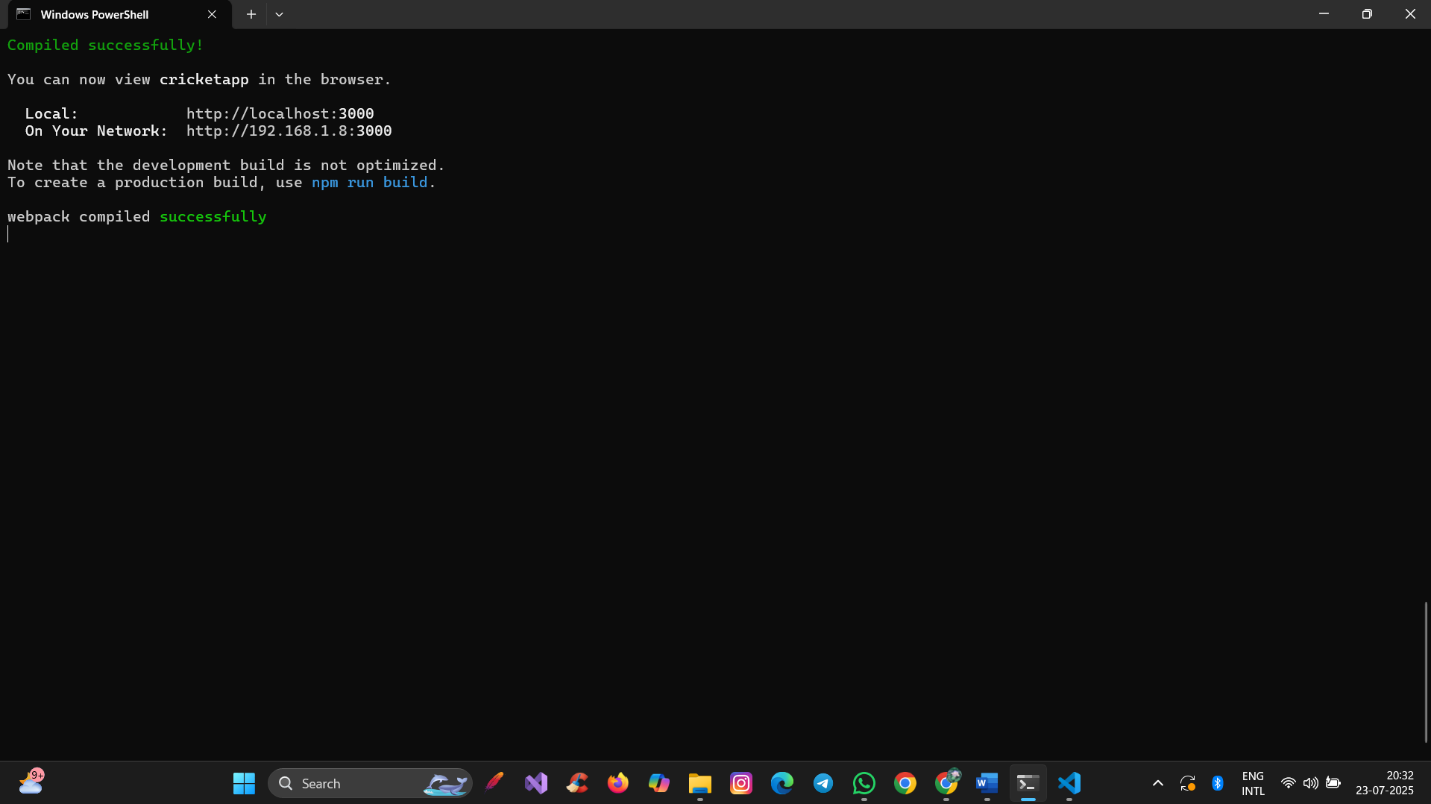
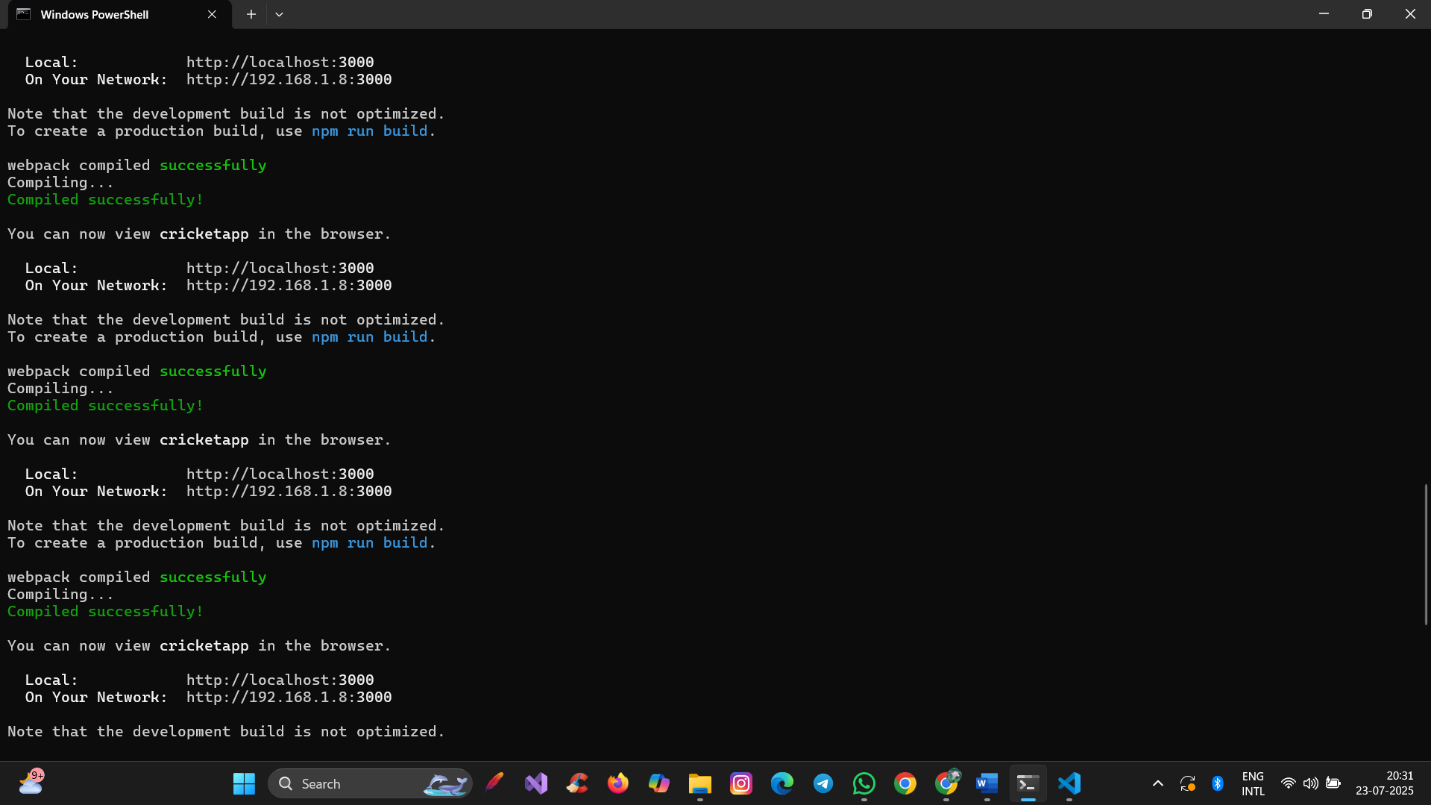
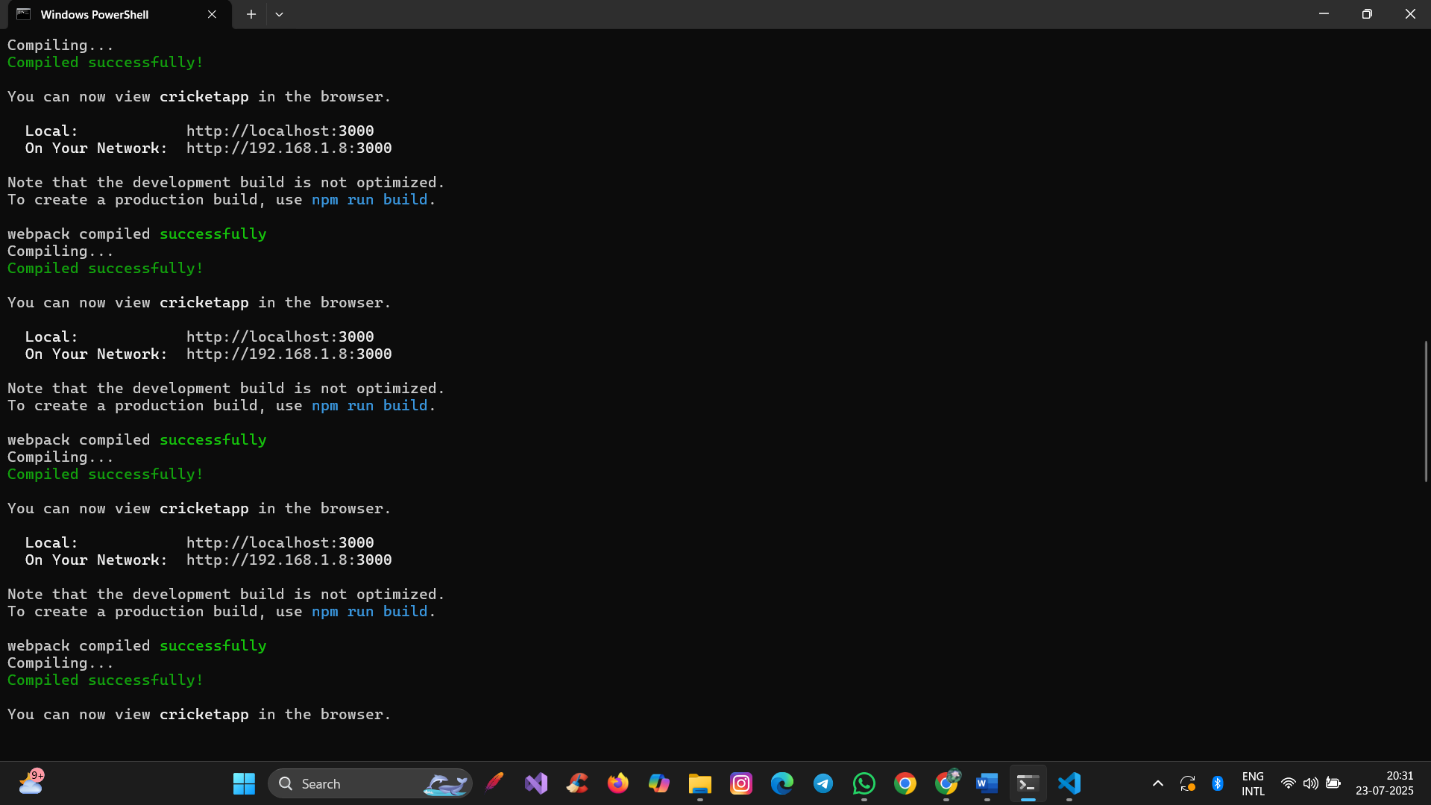
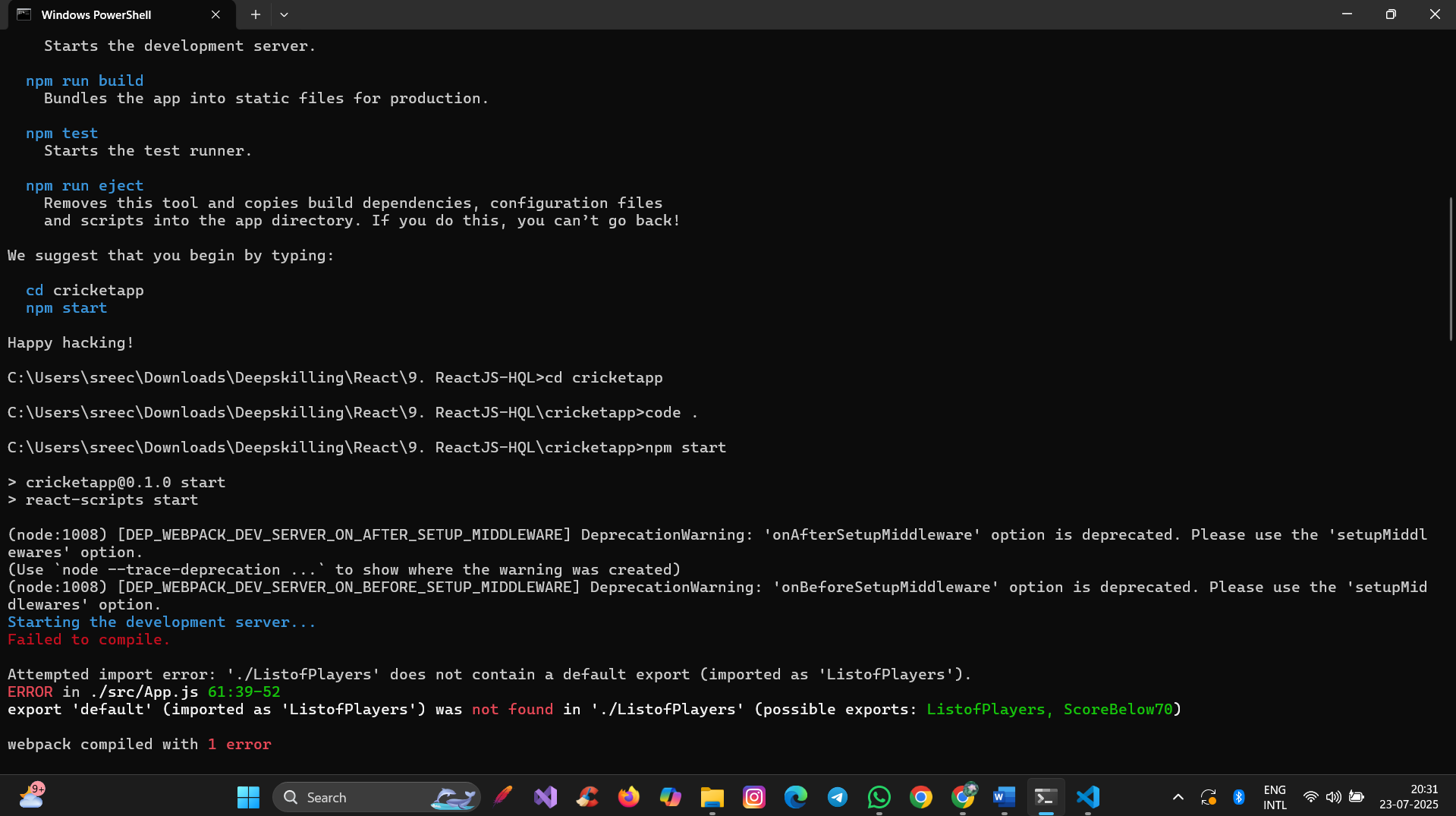
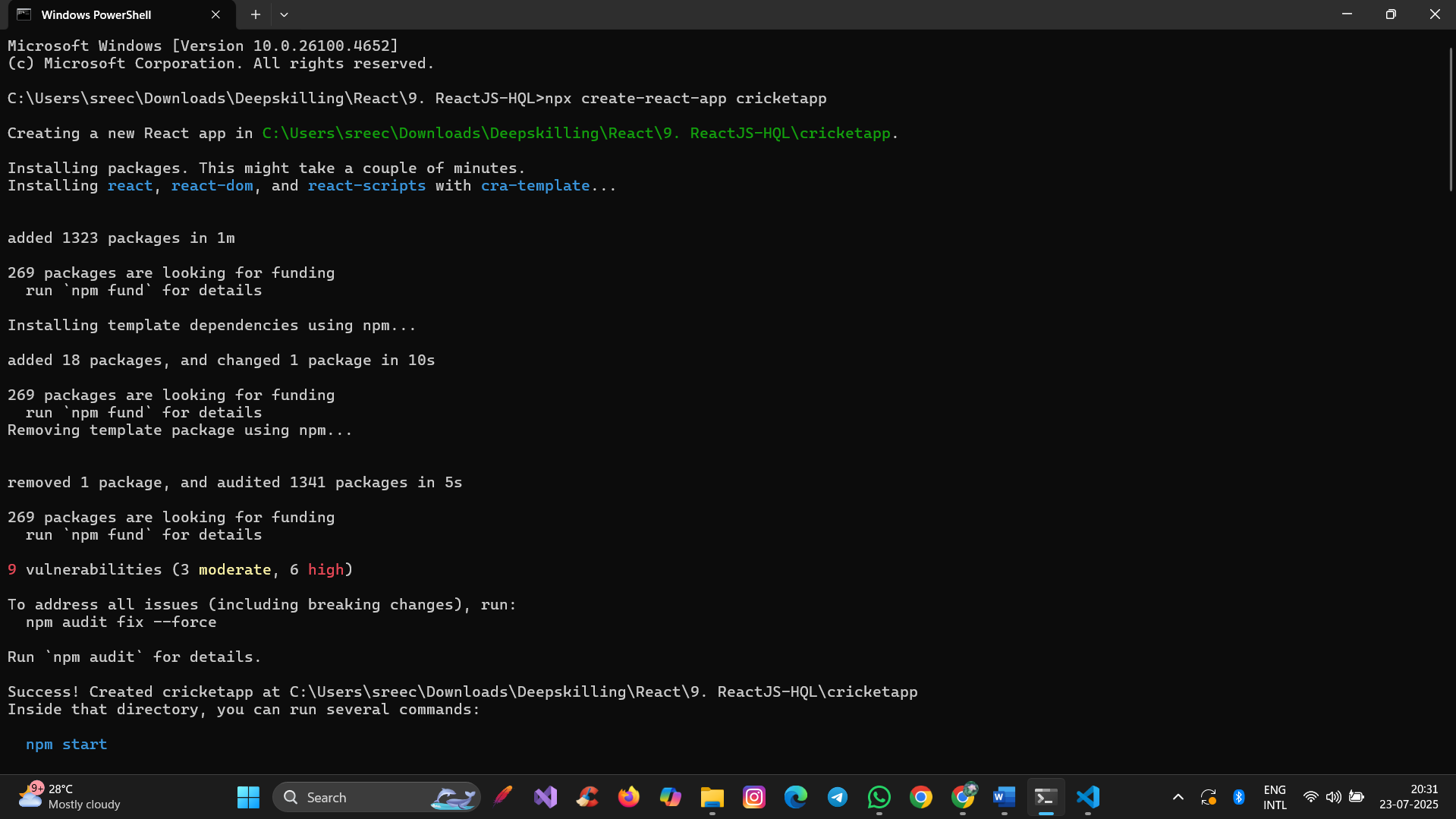
When Flag=true

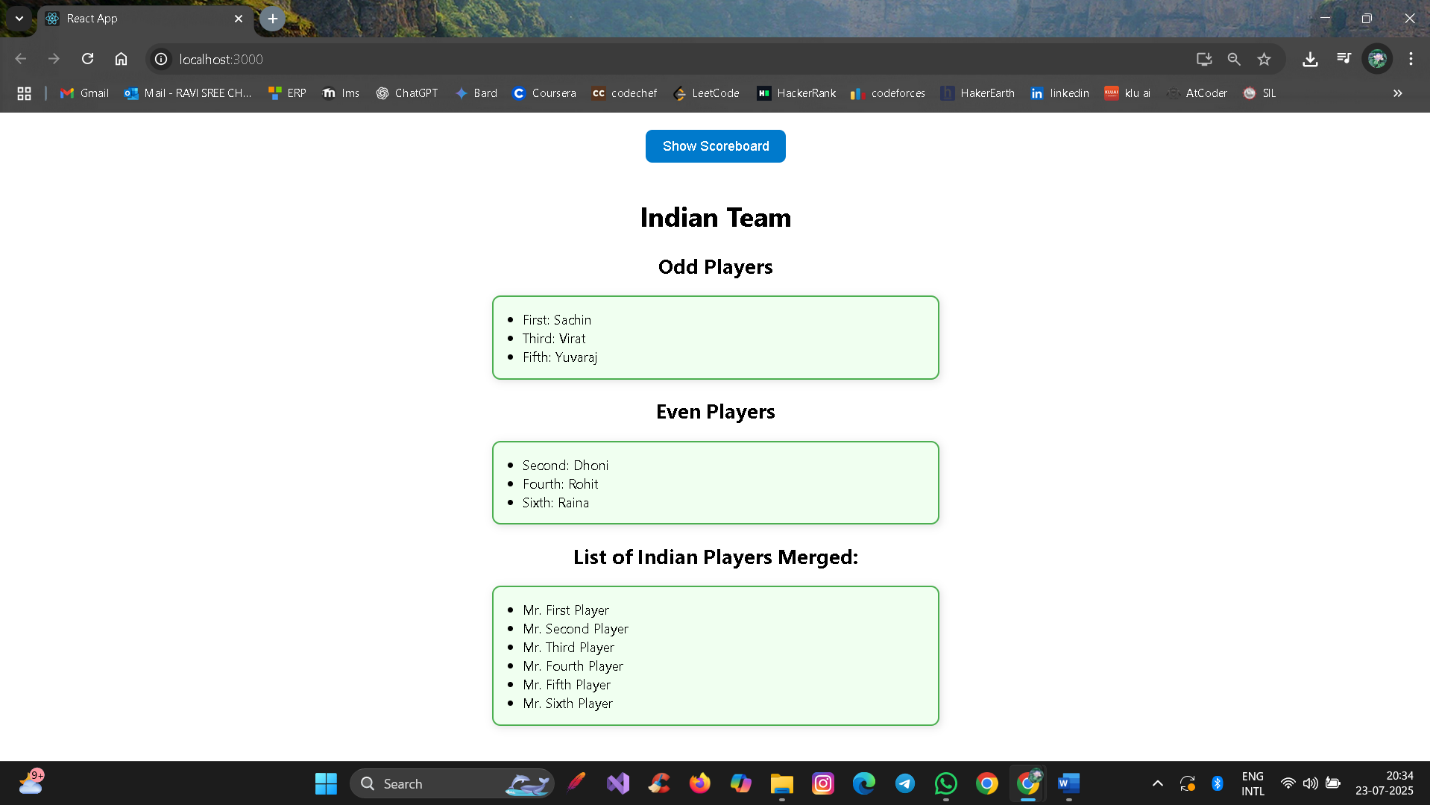
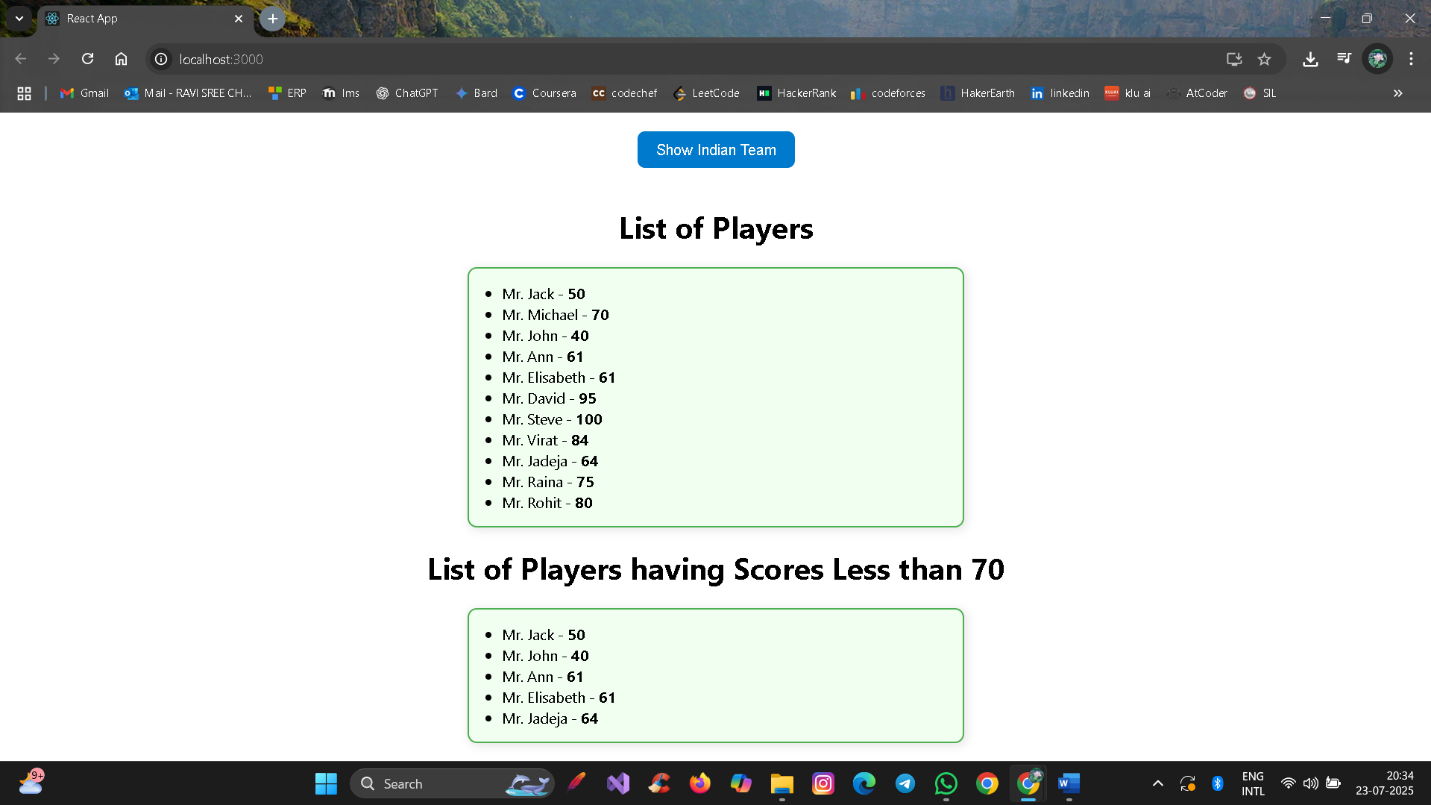
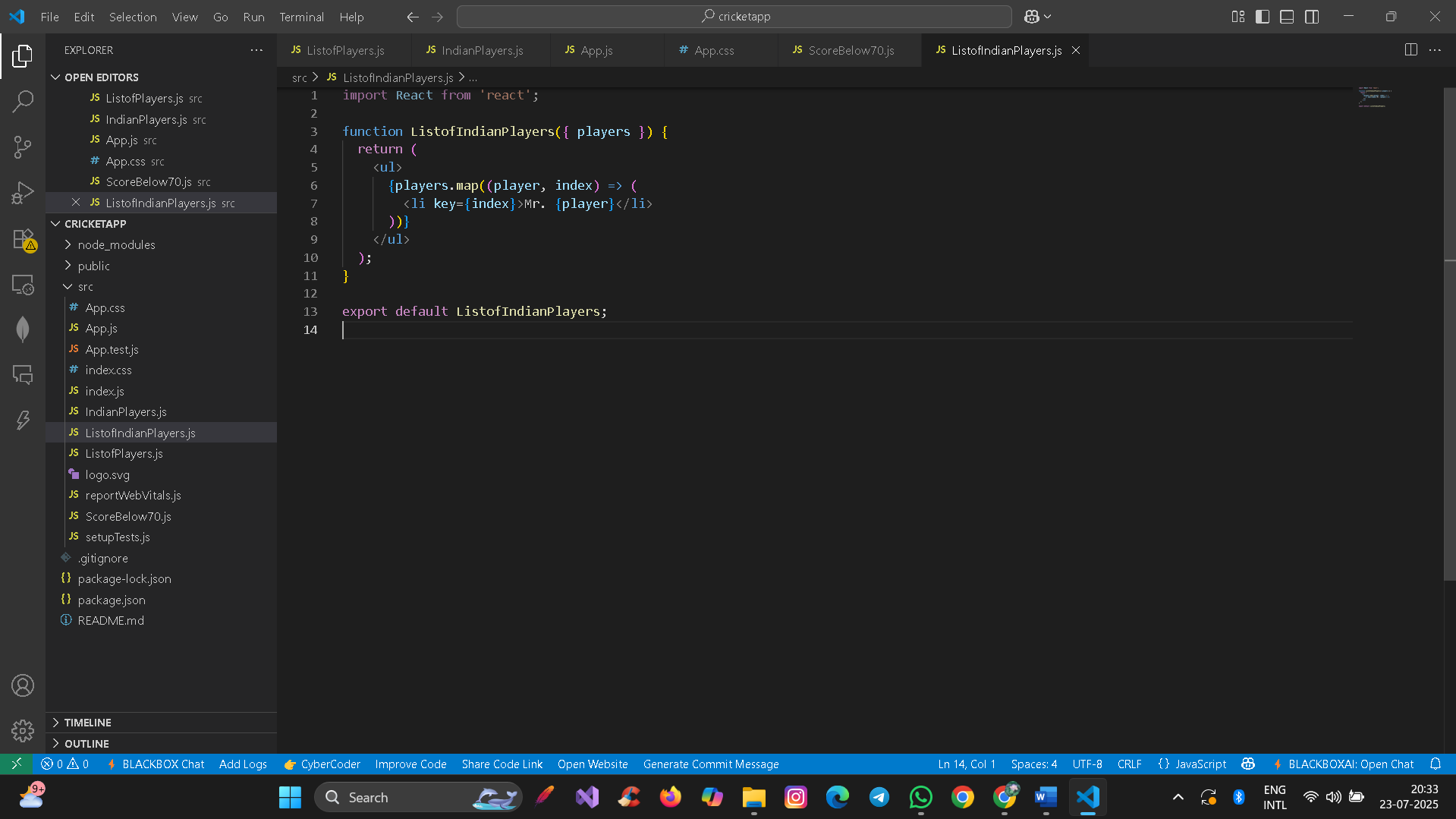
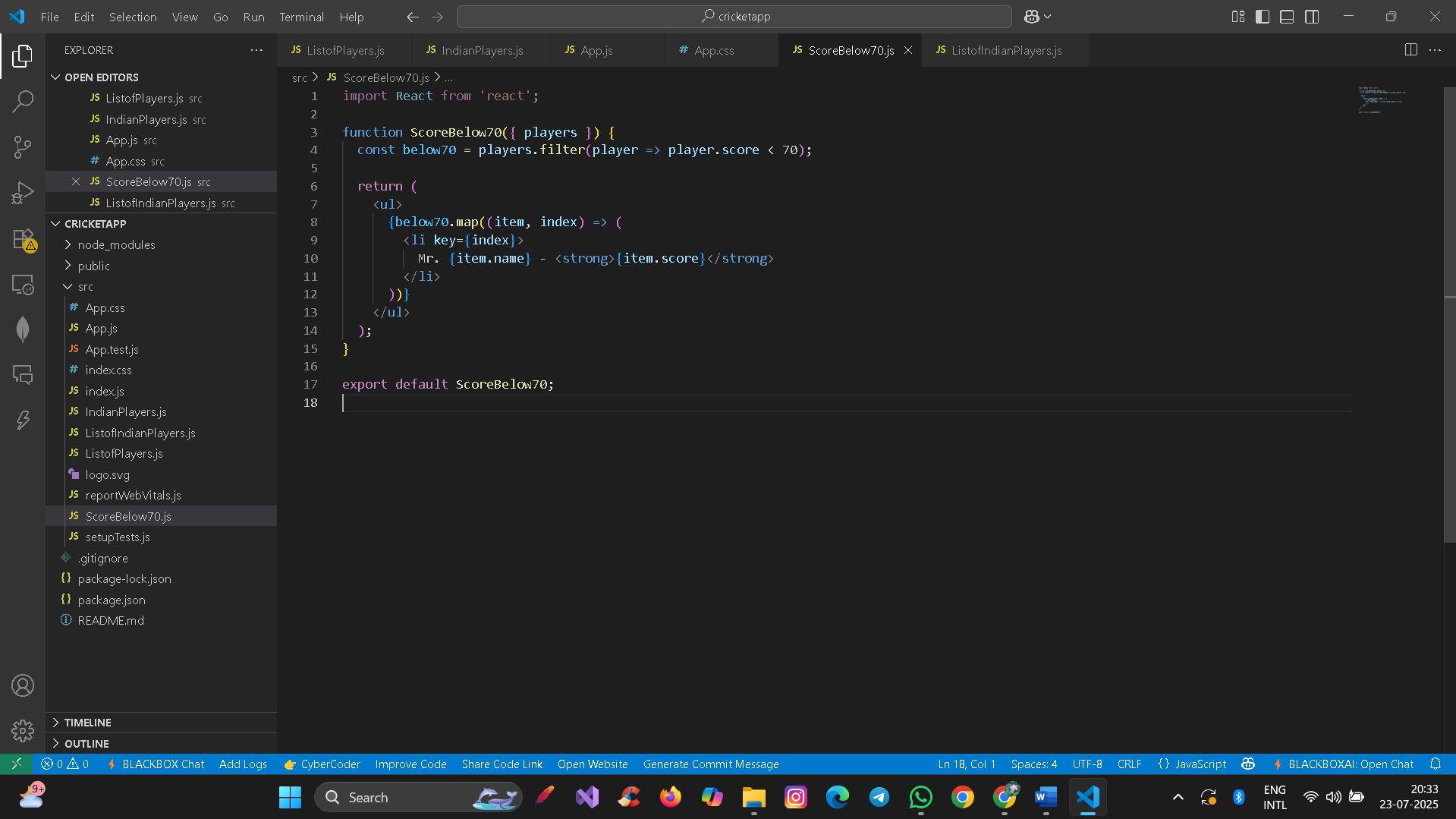
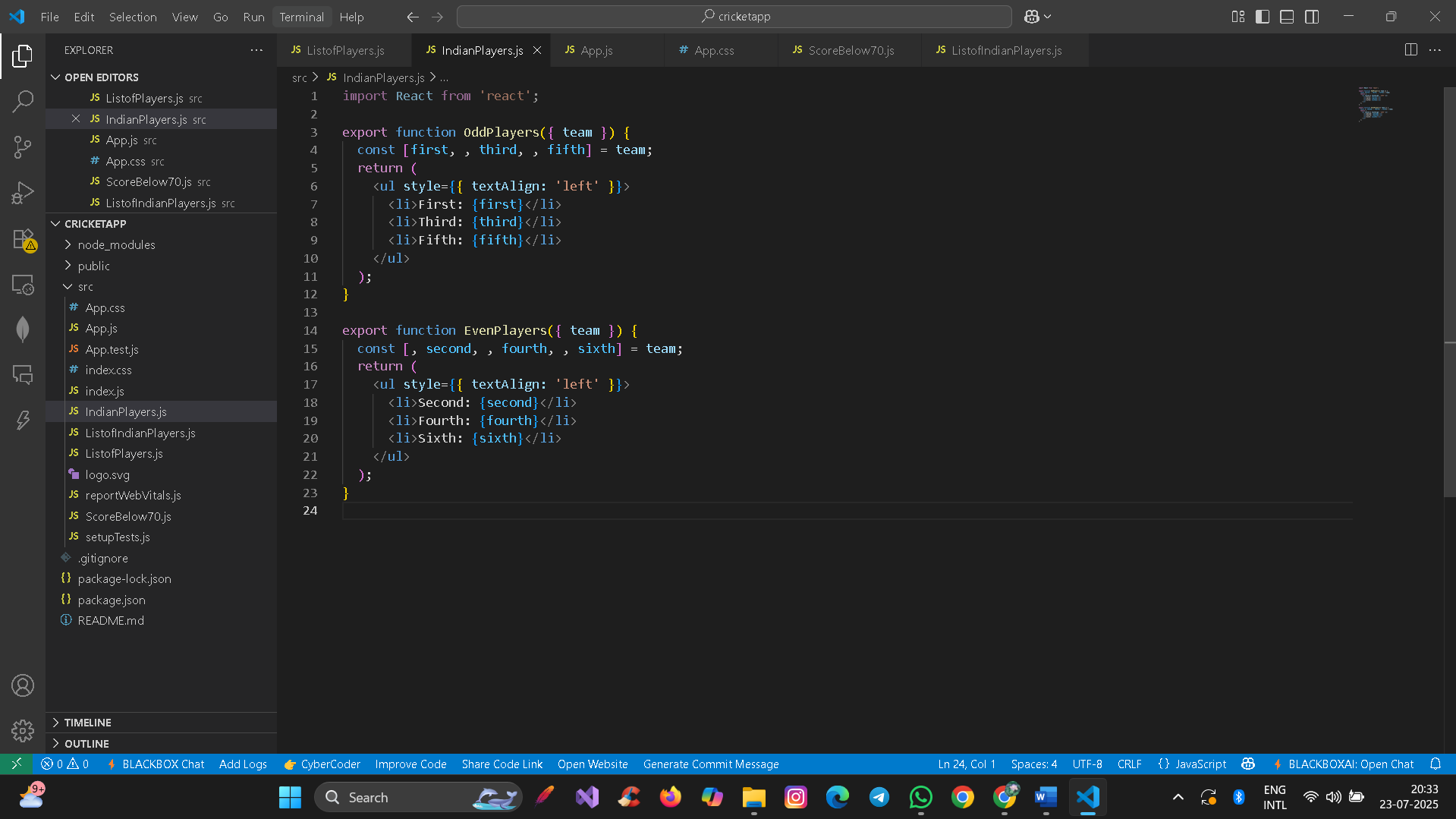
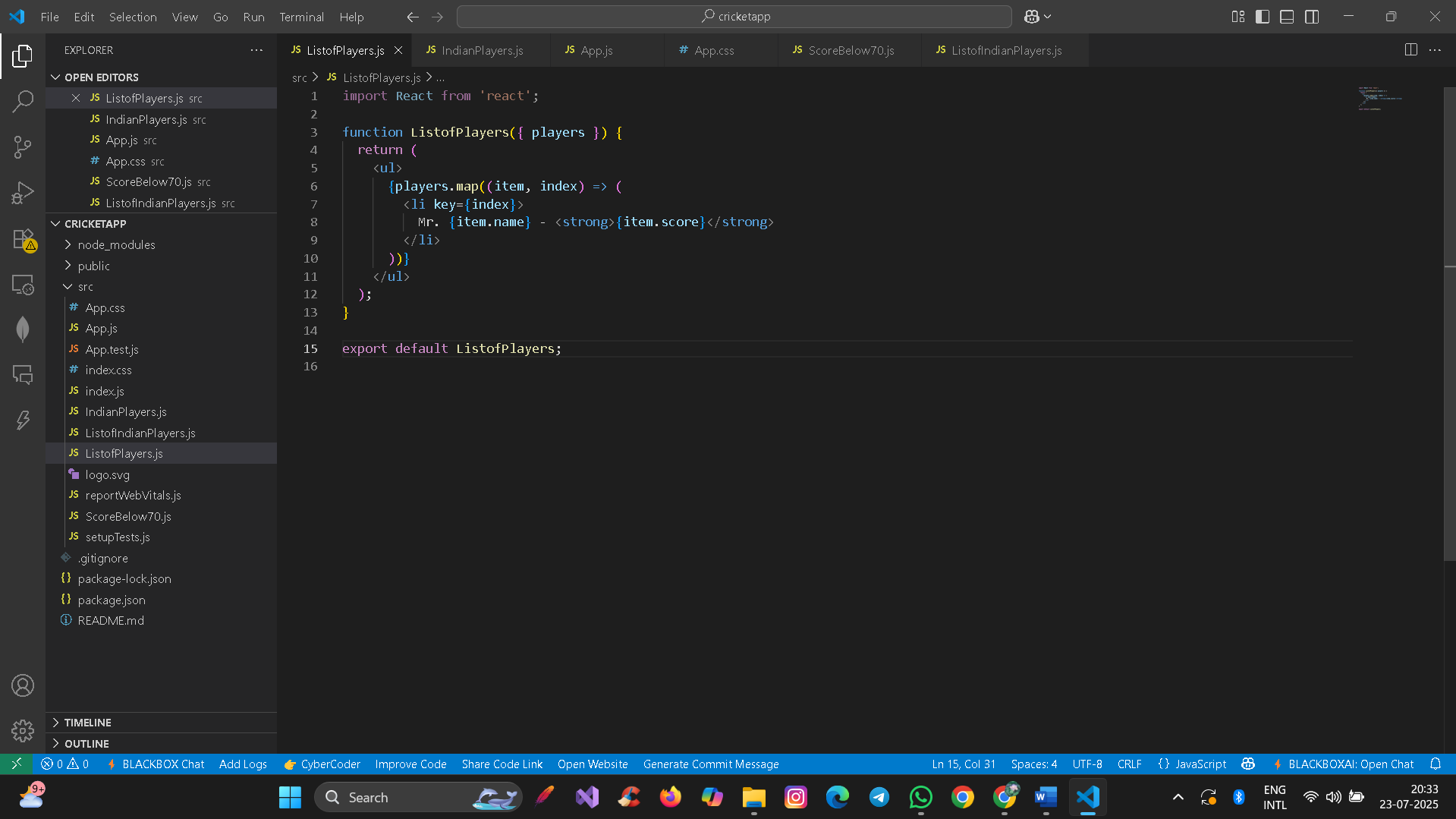
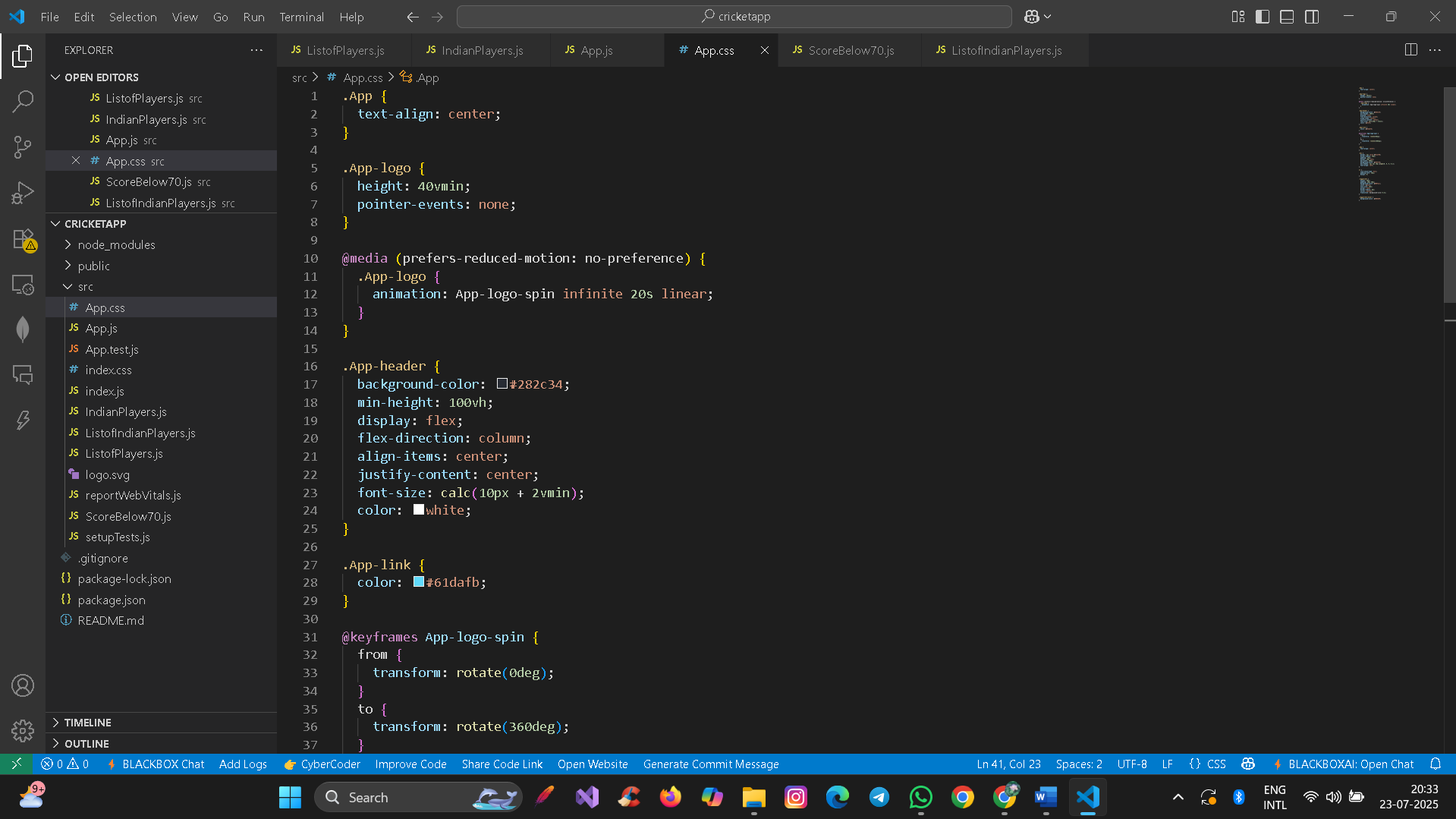
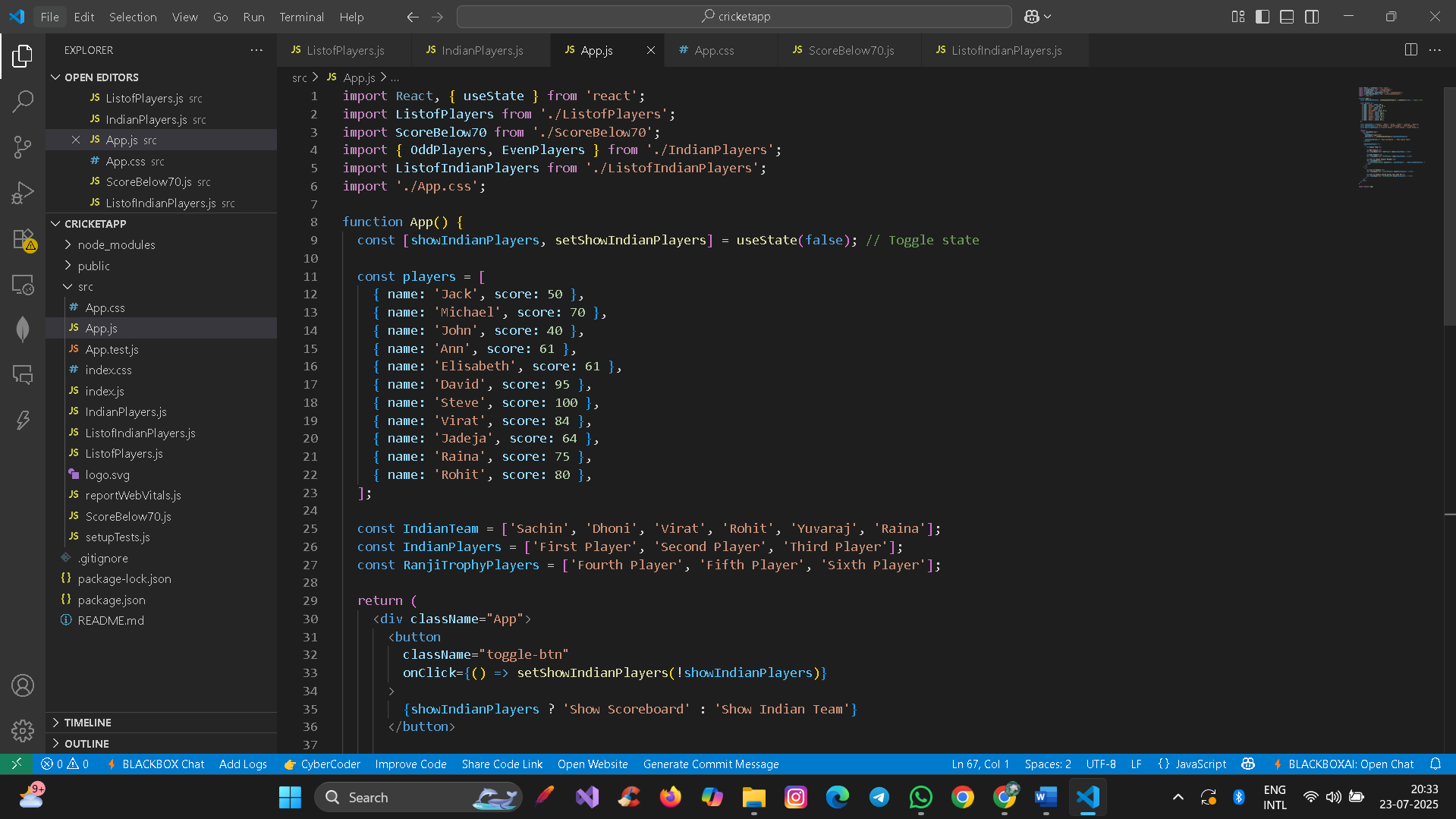


When Flag=false



**HandsOn:**

****

****



**HandsOn:**