**PRACTICAL NO-7**

**Aim:** Build Tic Tac Toe with React

**Description:** React is a frontend open-source JavaScript library, used to build interactive User Interfaces. React is focused on Single Page applications and is more popularly known as a SPA.

Modules required:

* npm
* React

**Creating React App and Setting Up:**

Step 1:You will start a new project using [**create-react-app**](https://www.geeksforgeeks.org/reactjs-setting-development-environment/) so open your terminal and type.

npx create-react-app tic-tac-toe

Step 2: Switch to the tic-tac-toe-react folder using the following command.

cd tic-tac-toe

Step 3:Create Components folder in the src.

Step 4:In the components folder create Board.css, Board.js, Box.css, Box.js, ResetButton.css, ResetButton.js, ScoreBoard.css, ScoreBoard.js

**Code:**

1. **App.js-(src)**

import React, { useState } from "react";

import { Board } from "./components/Board";

import { ResetButton } from "./components/ResetButton";

import { ScoreBoard } from "./components/ScoreBoard";

import './App.css';

const App = () => {

const WIN\_CONDITIONS = [

[0, 1, 2],

[3, 4, 5],

[6, 7, 8],

[0, 3, 6],

[1, 4, 7],

[2, 5, 8],

[0, 4, 8],

[2, 4, 6]

]

const [xPlaying, setXPlaying] = useState(true);

const [board, setBoard] = useState(Array(9).fill(null))

const [scores, setScores] = useState({ xScore: 0, oScore: 0 })

const [gameOver, setGameOver] = useState(false);

const handleBoxClick = (boxIdx) => {

// Step 1: Update the board

const updatedBoard = board.map((value, idx) => {

if (idx === boxIdx) {

return xPlaying ? "X" : "O";

} else {

return value;

}

})

setBoard(updatedBoard);

// Step 2: Check if either player has won the game

const winner = checkWinner(updatedBoard);

if (winner) {

if (winner === "O") {

let { oScore } = scores;

oScore += 1;

setScores({ ...scores, oScore })

} else {

let { xScore } = scores;

xScore += 1;

setScores({ ...scores, xScore })

}

}

// Step 3: Change active player

setXPlaying(!xPlaying);

}

const checkWinner = (board) => {

for (let i = 0; i < WIN\_CONDITIONS.length; i++) {

const [x, y, z] = WIN\_CONDITIONS[i];

// Iterate through win conditions and check if either player satisfies them

if (board[x] && board[x] === board[y] && board[y] === board[z]) {

setGameOver(true);

return board[x];

}

}

}

const resetBoard = () => {

setGameOver(false);

setBoard(Array(9).fill(null));

}

return (

<div className="App">

<ScoreBoard scores={scores} xPlaying={xPlaying} />

<Board board={board} onClick={gameOver ? resetBoard : handleBoxClick} />

<ResetButton resetBoard={resetBoard} />

</div>

);

}

export default App;

1. **App.css-(src)**

\* {

box-sizing: border-box;

}

body {

margin: 0;

background-color: #efefef;

font-family: "Fredoka", sans-serif;

}

1. **index.css-(src)**

body {

margin: 0;

font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',

'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',

sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

code {

font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',

monospace;

}

1. **index.js-(src)**

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

ReactDOM.render(

<React.StrictMode>

<App />

</React.StrictMode>,

document.getElementById('root')

);

1. **index.html-(public)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Fredoka:wght@400;500;700&display=swap" rel="stylesheet">

<meta name="viewport" content="width=device-width, initial-scale=1" />

<meta name="theme-color" content="#000000" />

<meta name="description" content="Web site created using create-react-app" />

<link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

<!--

manifest.json provides metadata used when your web app is installed on a

user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/

-->

<link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

<!--

Notice the use of %PUBLIC\_URL% in the tags above.

It will be replaced with the URL of the `public` folder during the build.

Only files inside the `public` folder can be referenced from the HTML.

Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

work correctly both with client-side routing and a non-root public URL.

Learn how to configure a non-root public URL by running `npm run build`.

-->

<title>React App</title>

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="root"></div>

<!--

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add webfonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the <body> tag.

To begin the development, run `npm start` or `yarn start`.

To create a production bundle, use `npm run build` or `yarn build`.

-->

</body>

</html>

1. **Board.css-(Components)**

.board {

display: grid;

grid-template-columns: repeat(3, 6rem);

place-items: center;

justify-content: center;

}

1. **Board.js-(Components)**

import React from 'react'

import { Box } from "./Box"

import "./Board.css"

export const Board = ({ board, onClick }) => {

return (

<div className="board">

{

board.map((value, idx) => {

return <Box value={value} onClick={() => value === null && onClick(idx)} />;

})

}

</div>

)

}

1. **Box.css-(Components)**

.box {

background-color: #ffffff;

border: none;

border-radius: 10%;

box-shadow: 0px 0px 8px #888888;

width: 5rem;

height: 5rem;

text-align: center;

font-size: 5em;

font-family: "Fredoka", sans-serif;

font-weight: bold;

line-height: 5rem;

margin: 0.5rem;

}

.x {

color: rgb(255, 70, 37);

}

.o {

color: rgb(44, 135, 255);

}

.box:hover {

box-shadow: 0px 0px 15px #888888;

}

1. **Box.js-(Components)**

import React from 'react';

import "./Box.css";

export const Box = ({ value, onClick }) => {

const style = value === "X" ? "box x" : "box o";

return (

<button className={style} onClick={onClick}>{value}</button>

)

}

1. **ResetButton.css-(Components)**

.reset-btn {

border: none;

border-radius: 0.5rem;

background-color: rgb(0, 110, 255);

color: white;

font-size: 2rem;

padding: 0.5rem 1rem;

margin: 2rem auto;

display: block;

}

.reset-btn:hover {

background-color: rgb(0, 119, 255);

}

1. **ResetButton.js-(Components)**

import React from 'react';

import "./ResetButton.css";

export const ResetButton = ({ resetBoard }) => {

return (

<button className="reset-btn" onClick={resetBoard}>Reset</button>

)

}

1. **ScoreBoard.css-(Components)**

.scoreboard {

display: flex;

flex-direction: row;

align-items: center;

justify-content: space-evenly;

width: 20rem;

font-size: 1.5rem;

background-color: white;

margin: 3rem auto;

box-shadow: 0px 0px 8px #888888;

border-radius: 0.5rem;

font-weight: bold;

}

.score {

width: 100%;

text-align: center;

padding: 1rem 0rem;

}

.x-score {

color: rgb(255, 70, 37);

border-bottom: 5px solid rgb(255, 70, 37);

border-radius: 0.5rem 0rem 0rem 0.5rem;

}

.o-score {

color: rgb(44, 135, 255);

border-bottom: 5px solid rgb(44, 135, 255);

border-radius: 0rem 0.5rem 0.5rem 0rem;

}

.inactive {

border-bottom: 5px solid transparent;

}

1. **ScoreBoard.js-(Components)**

import React from 'react'

import "./ScoreBoard.css"

export const ScoreBoard = ({ scores, xPlaying }) => {

const { xScore, oScore } = scores;

return (

<div className="scoreboard">

<span className={`score x-score ${!xPlaying && "inactive"}`}>X - {xScore}</span>

<span className={`score o-score ${xPlaying && "inactive"}`}>O - {oScore}</span>

</div>

)

}

Save all files and start the application by running the following command:

npm start

**Output:**



