Experiment No. 08

Title: Design a web page using React JS.

Batch:A1 **Roll No: 16010422013** **Experiment No:8**

Aim: To design a web page using React JS.

Resources needed: Notepad, any Web Browser and Internet.

Theory:

React (also known as React.js or ReactJS) is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM(Document Object Model), so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality. ReactJS is JavaScript library used for building reusable UI components.

Features of React

- **JSX** − JSX is JavaScript syntax extension. It isn't necessary to use JSX in React development, but

it is recommended.

- **Components** − React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale projects.
- **Unidirectional data flow and Flux** − React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keeping your data unidirectional.
- **License** − React is licensed under the Facebook Inc. Documentation is licensed under CC BY 4.0.

ReactJS - Environment Setup

1. First you need to install NodeJS
2. Second install ReactJS

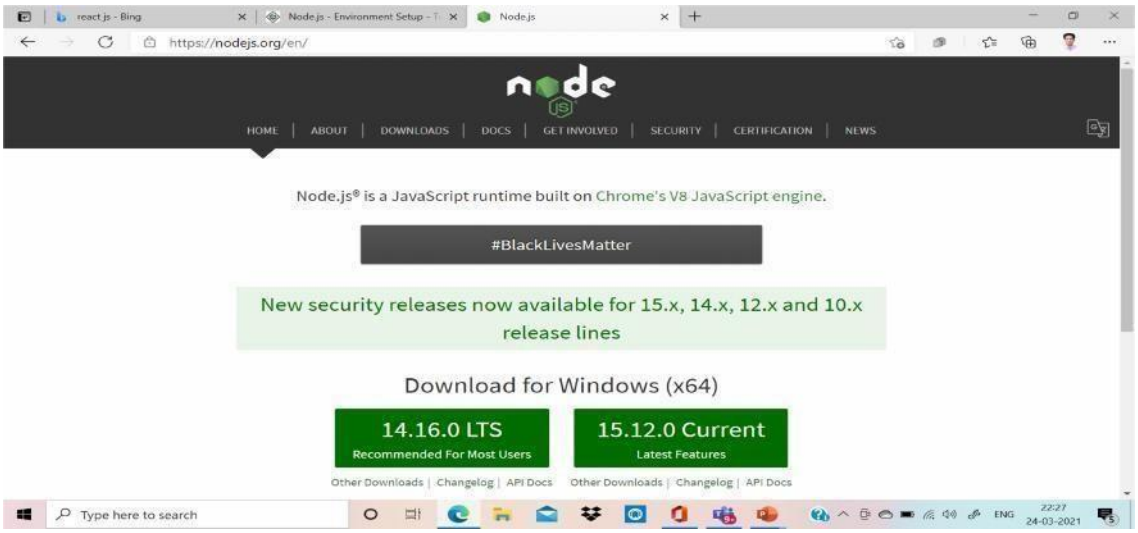
Install NodeJS:

Step 1:

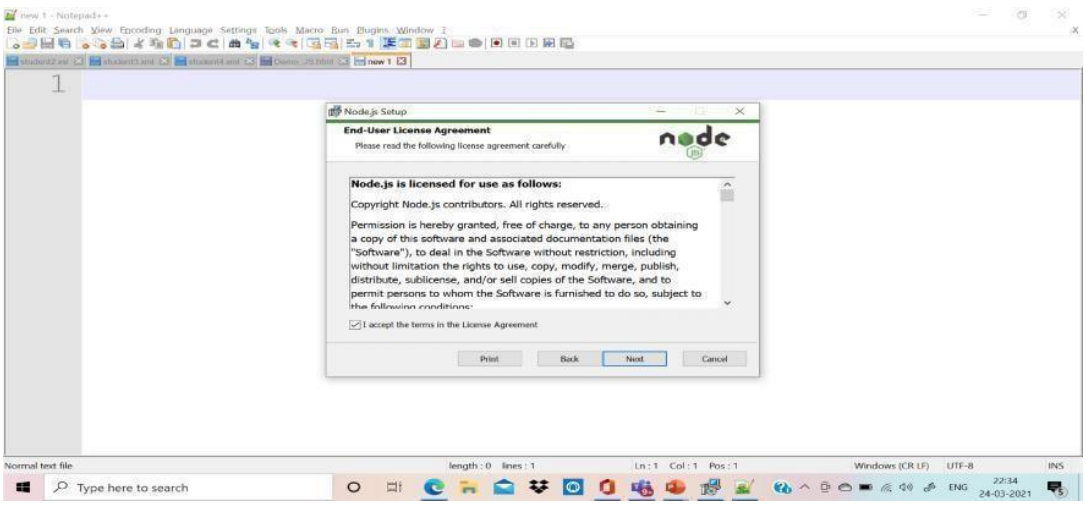
Visit the website www.Nodejs.org/en/. For installation on Windows ,you use the MSI file and follow the prompts to install the Node.js. By default, the installer uses the Node.js distribution in C:\Program Files\nodejs. The installer should set the C:\Program Files\nodejs\bin directory in window's PATH environment variable. Restart any open command prompts for the change to take effect. The source code written in source file is simply javascript. The Node.js interpreter will be used to interpret and execute your javascript code. Node.js distribution comes as a binary installable for SunOS , Linux, Mac OS X, and Windows operating systems with the 32-bit (386) and 64-bit (amd64) x86 processor architectures. Next step will guide to install Node.js binary distribution on windows OS.

Step 2:

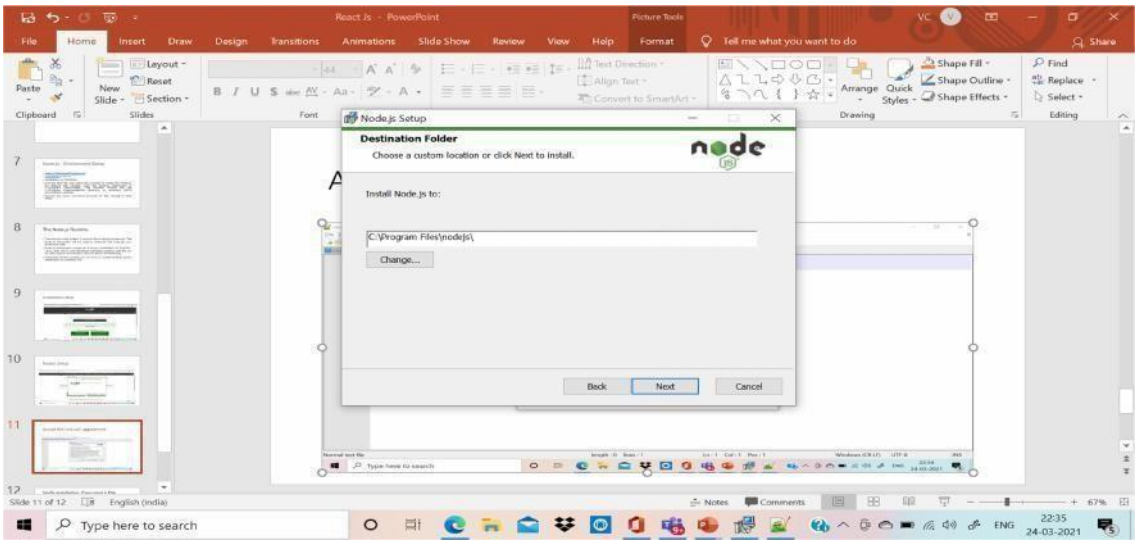
Step3:



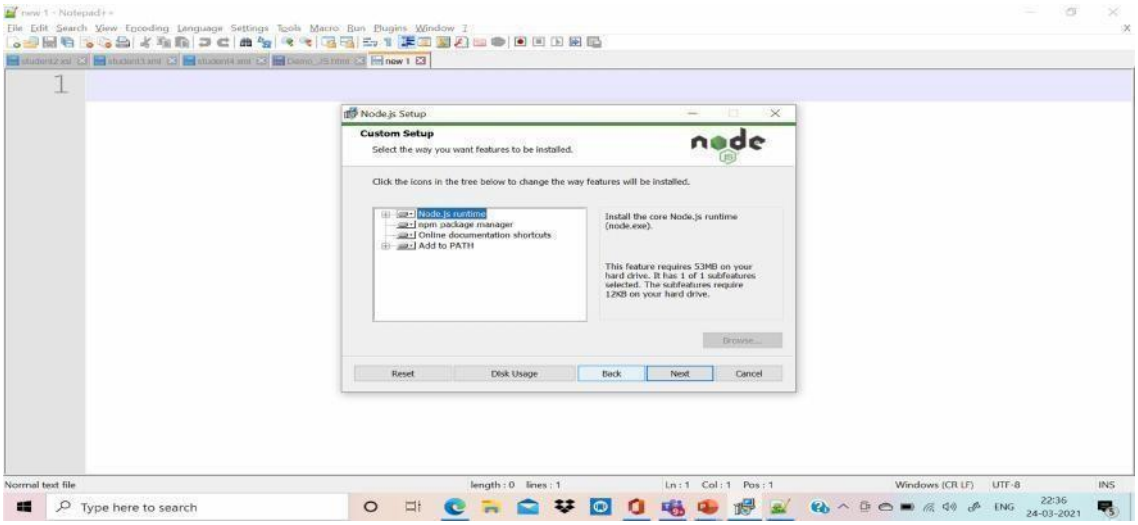
Step4: Accept the Agreement



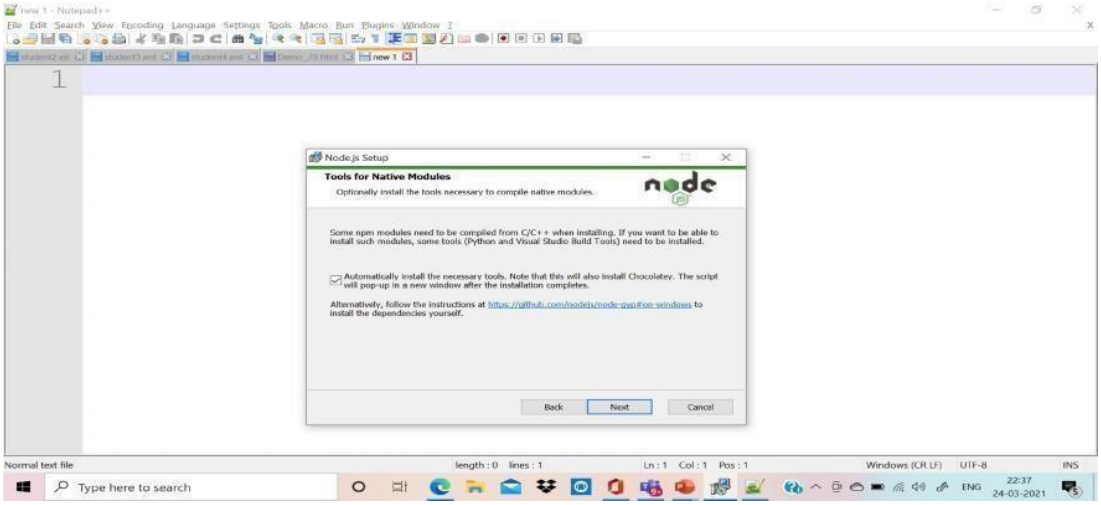
Step5: Choose Destination Folder



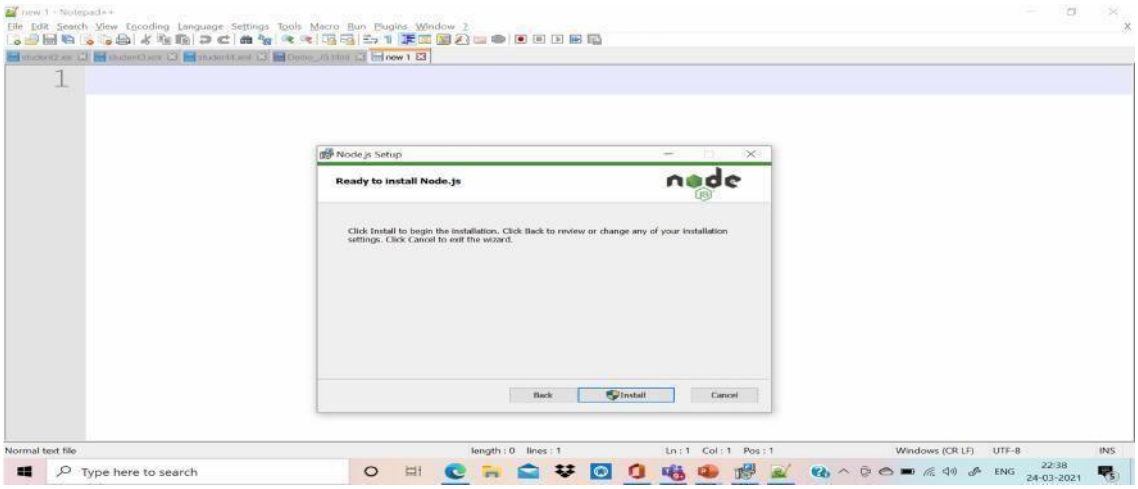
Step6: Custom Installation setup



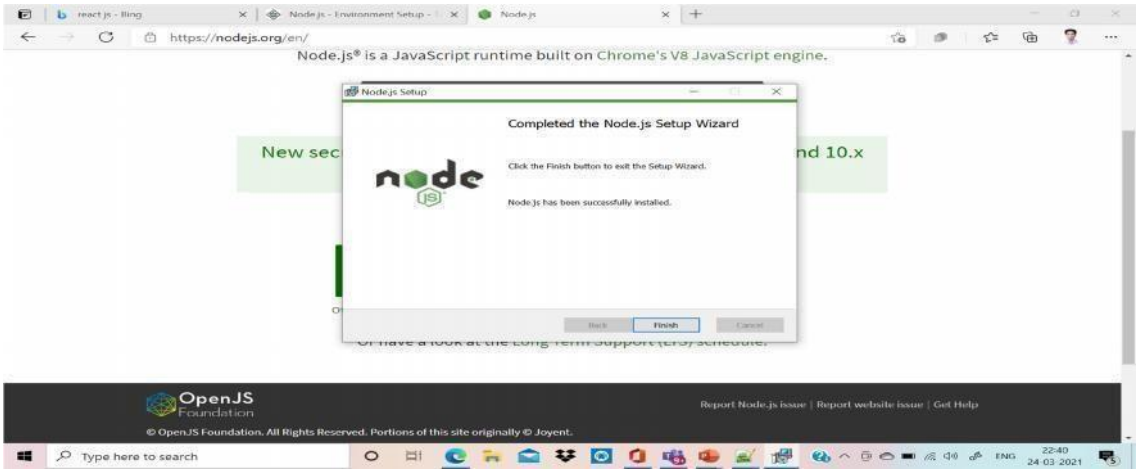
Step 7: Additional Tools Setup



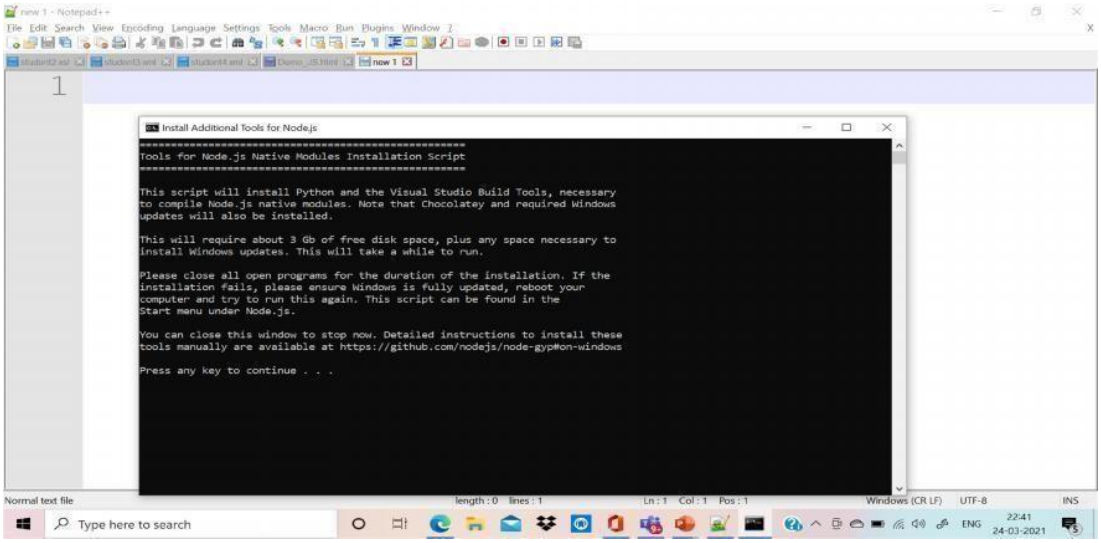
Step8: Ready to install Nodejs



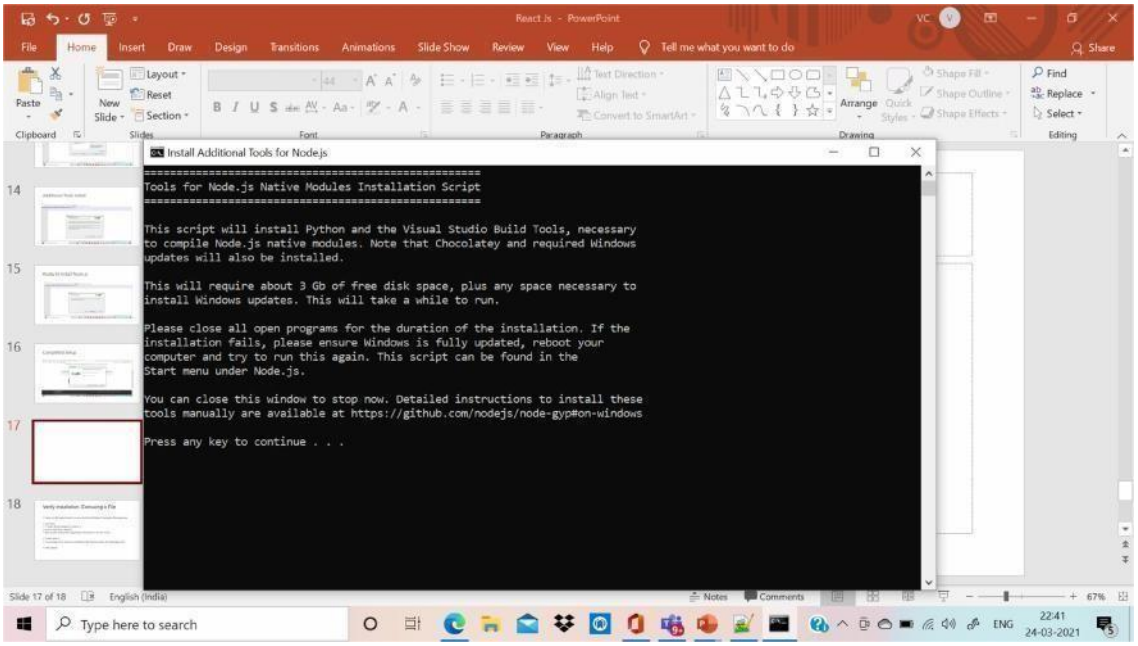
Step9: Completed Setup



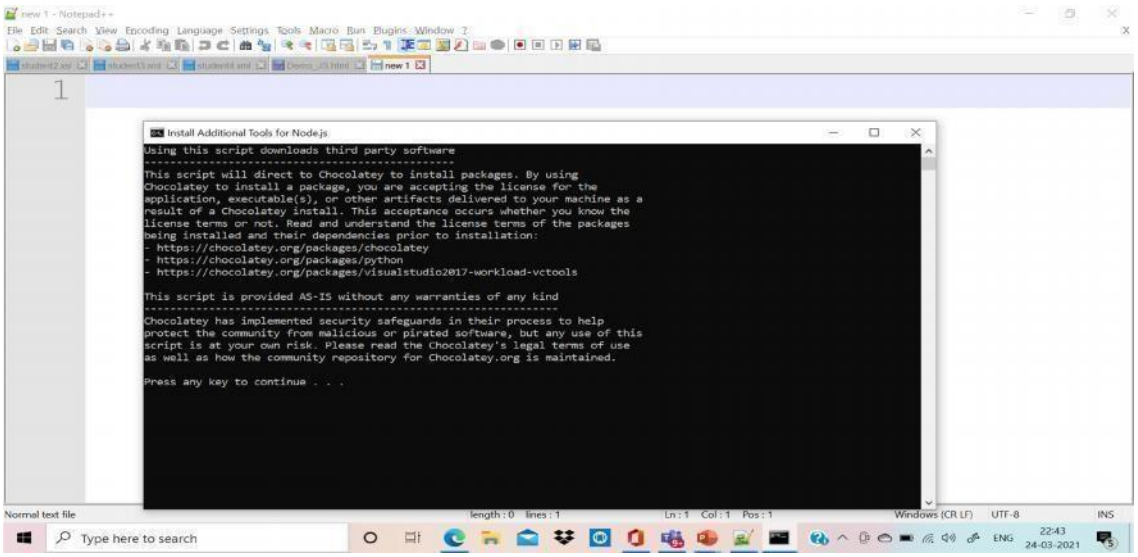
Step 10: Ready to start



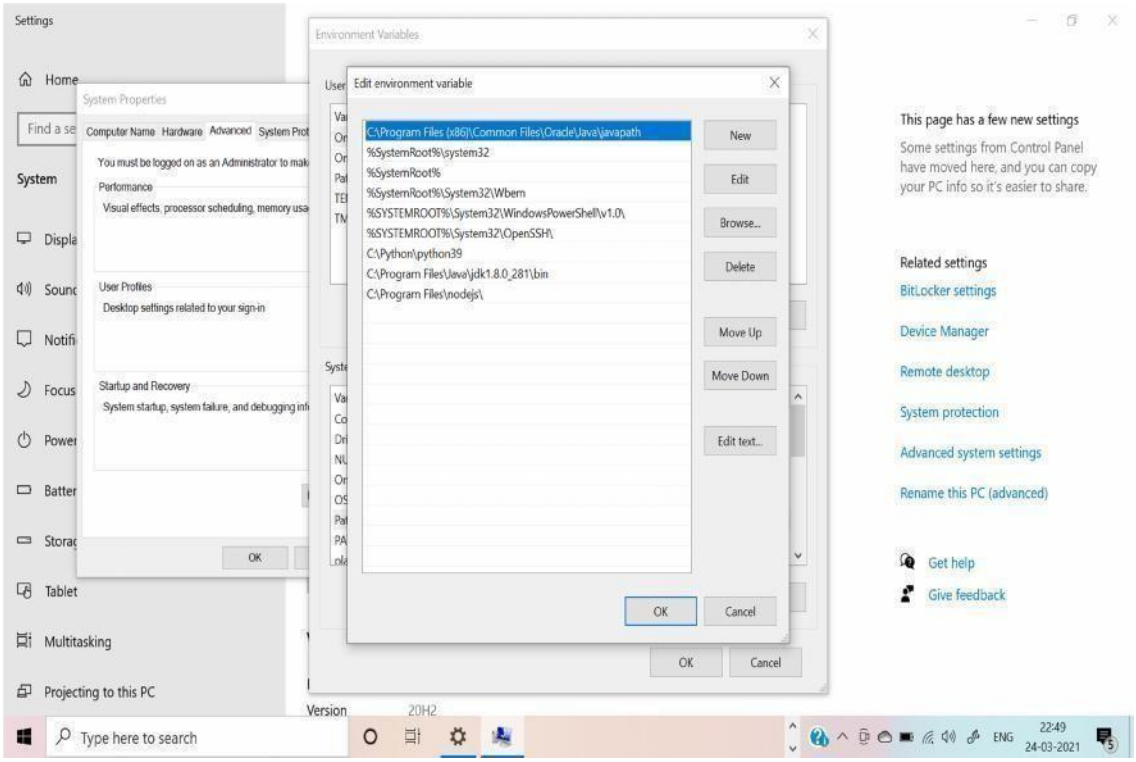
Step11: Install Native Module



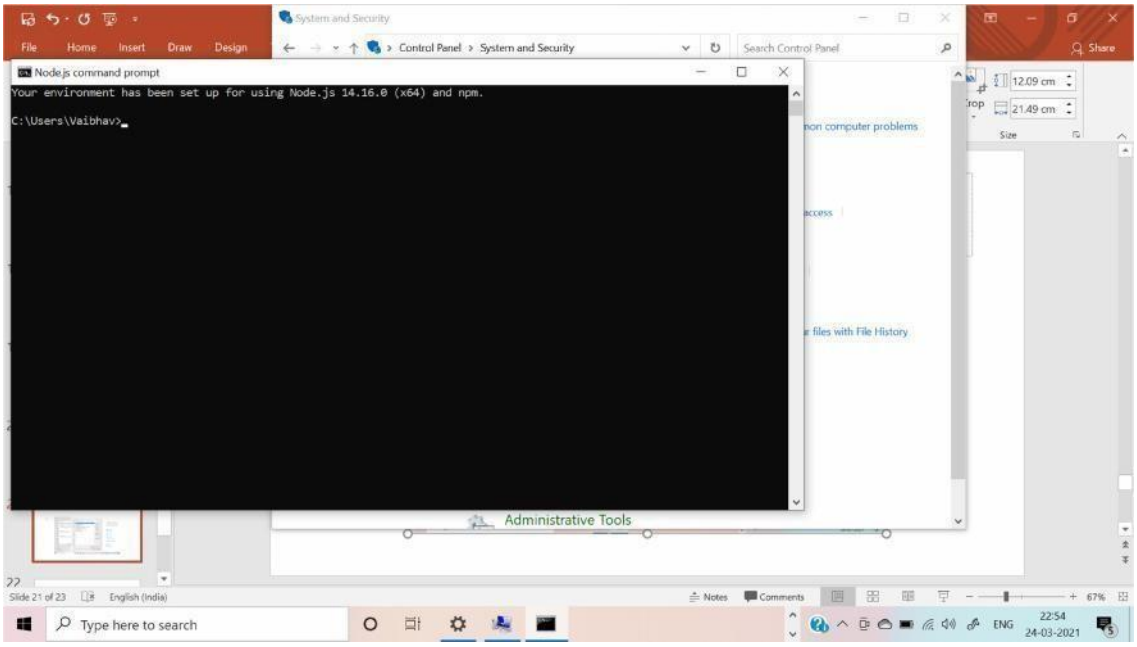
Step 12: Additional Installation



Step 13: Verify the setup



Step 14: Setup Message on Command Prompt: For this message go start menu and click the button you will find command prompt menu available with cmd .



Step 15: Check the version

- **C:\>node -v**
- **□v14.16.0**

Step 16: Type the Hello World and execute

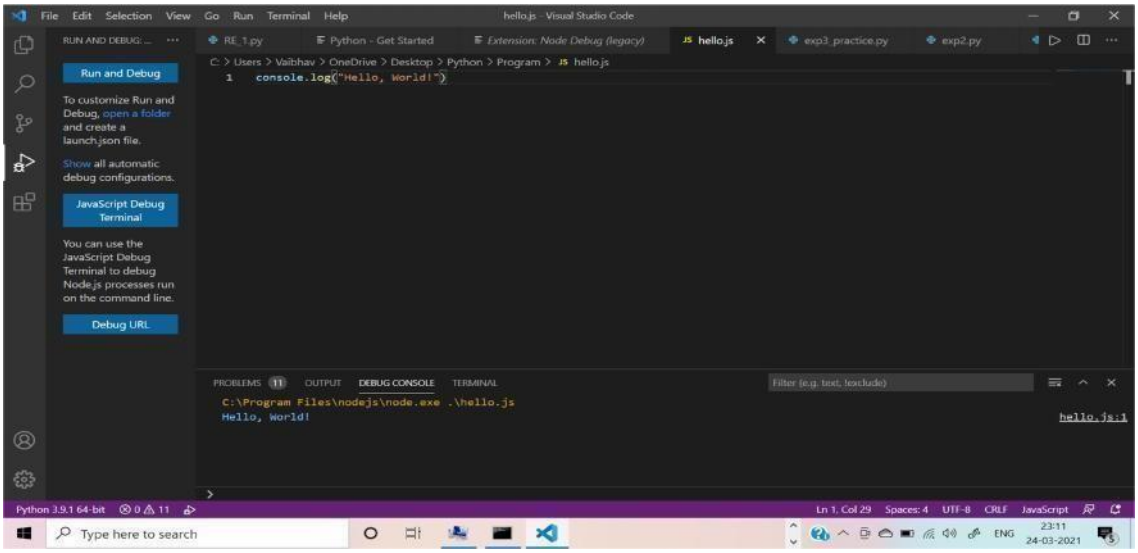
Type the command and install

gyrescipt : C:\>npm install -g

typescript

- Create a js file named main.js on your machine (Windows) having the following code.
- Live Demo
- /* Hello, World! program in node.js */
- console.log("Hello, World!")
- Now execute main.js file using Node.js interpreter to see the result −
- $ node main.js
- If everything is fine with your installation, this should produce the following result −
- Hello, World!

Step 17: Install VS Code: Editor Tool



Install ReactJS:

There are 3 ways to install ReactJS :

1) webpack
2) Babel
3) create-react-app

Webpack:

Webpack is a module bundler (manages and loads independent modules). It takes

dependent modules and compiles them to a single (file) bundle. You can use this bundle while developing apps using command line or, by configuring it using webpack.config file.

Steps:

1) Since we are using webpack to generate bundler install webpack, webpack-dev-server and webpack-cli.
2) C:\Users\username\Desktop\reactApp>npm install webpack --save
3) C:\Users**username\Desktop\reactApp>npm install webpack-dev-server** --save
4) C:\Users\username\Desktop\reactApp>npm install webpack-cli --save
5) Or, you can install all of them in single command as −
6) C:\Users\username\Desktop\reactApp>npm install webpack webpack-dev-server --save

Babel:

Babel is a JavaScript compiler and transpiler. It is used to convert one source code to other. Using this you will be able to use the new ES6 features in your code where, babel converts it into plain old ES5 which can be run on all browsers.

1) Install babel, and its plugins babel-core, babel-loader, babel-preset-env, babel-preset-react and, html-webpack-plugin
2) C:\Users\username\Desktop\reactApp>npm install babel-core --save-dev
3) C:\Users\username\Desktop\reactApp>npm install babel-loader --save-dev
4) C:\Users\username\Desktop\reactApp>npm install babel-preset-env --save-dev
5) C:\Users\username\Desktop\reactApp>npm install babel-preset-react --save-dev
6) C:\Users\username\Desktop\reactApp>npm install html-webpack-plugin --save-dev
7) Or, you can install all of them in single command as −
8) C:\Users\username\Desktop\reactApp>npm install babel-core babel-loader babel-preset-env
9) babel-preset-react html-webpack-plugin --save-dev

Working with ReactJS

In this experiment , we are going to use this steps of create react

app Steps are as follows:

1) Create a folder with name reactApp on the desktop to install all the required files, using the mkdir command.

C:\Users\username\Desktop>mkdir reactApp

2) Change the directory:

C:\Users\username\Desktop>cd reactApp

C:\Users\vaibhav>cd C:\Users\vaibhav\reactapp\

3) Install ReactJS:

C:\Users\vaibhav\reactapp>npx create-react-app my-app

This will create a folder named my-app on the desktop and installs all the required files in it.

npm (node package manager) is the dependency/package manager you get out of the box when you install Node.js. It provides a way for developers to install packages both globally and locally

npx: The npx stands for Node Package Execute and it comes with the npm, when you installed npm above 5.2.0 version then automatically npx will installed. It is an npm package runner that can execute any package that you want from the npm registry without even installing that package. The npx is useful during a single time use package. If you have installed npm below 5.2.0 then npx is not installed in your system.

4) Delete all source files

- Browse through the src folder in the generated my-app folder and remove all the files in it as shown below −

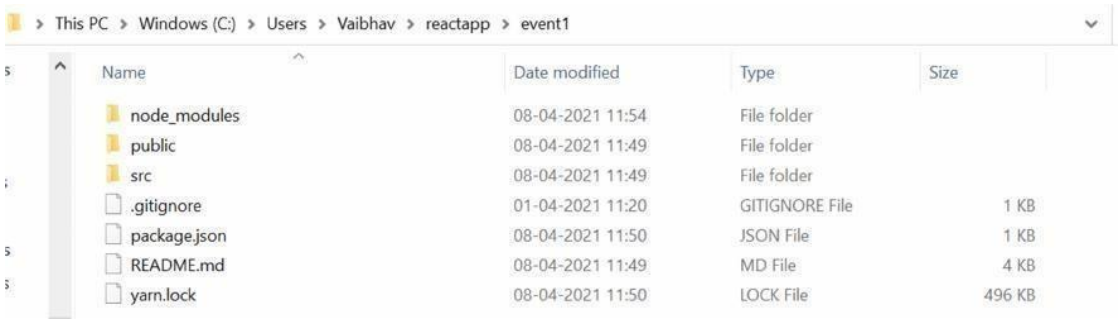
- C:\Users\Desktop>cd my-app/src
- C:\Users\Desktop\my-app\src>del *
- C:\Users\Desktop\my-app\src*, Are you sure (Y/N)? y

5) Add files with names index.css and index.js in the src folder as −

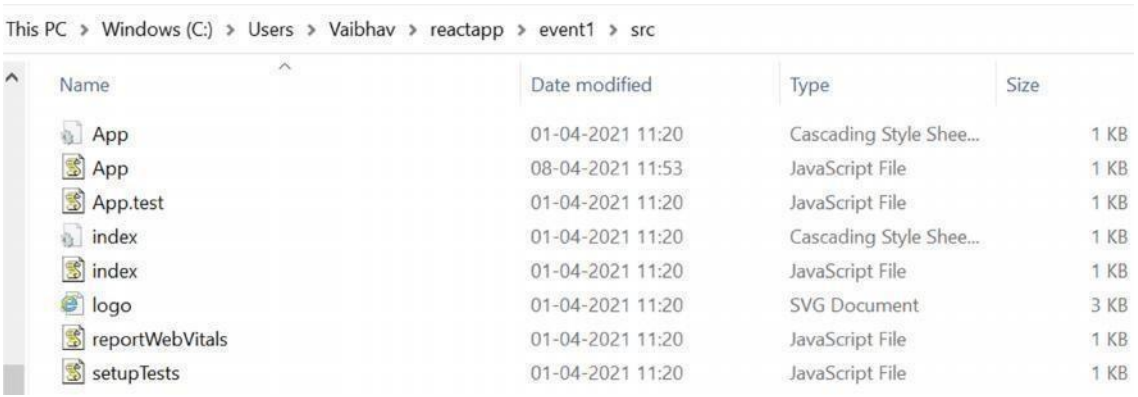
- C:\Users\Desktop\my-app\src>type nul > index.css
- C:\Users\Tutorialspoint\Desktop\my-app\src>type nul > index.js

6) Skip step 4 and 5 and only delete the file called App.js

7) Locate your code into these folder as event1 as shown below:



8) Open the src folder as shown below:



9) Choose the App.js file and indite code which is given below.

```
import React, {Component} from 'react';
class App extends
  React.Component {
  constructor(props) {
    super(props);
    this.state = {
      companyName: ''

    };

  }

  changeText(event) {
    this.setState({
      companyName: event.target.value

    });
```

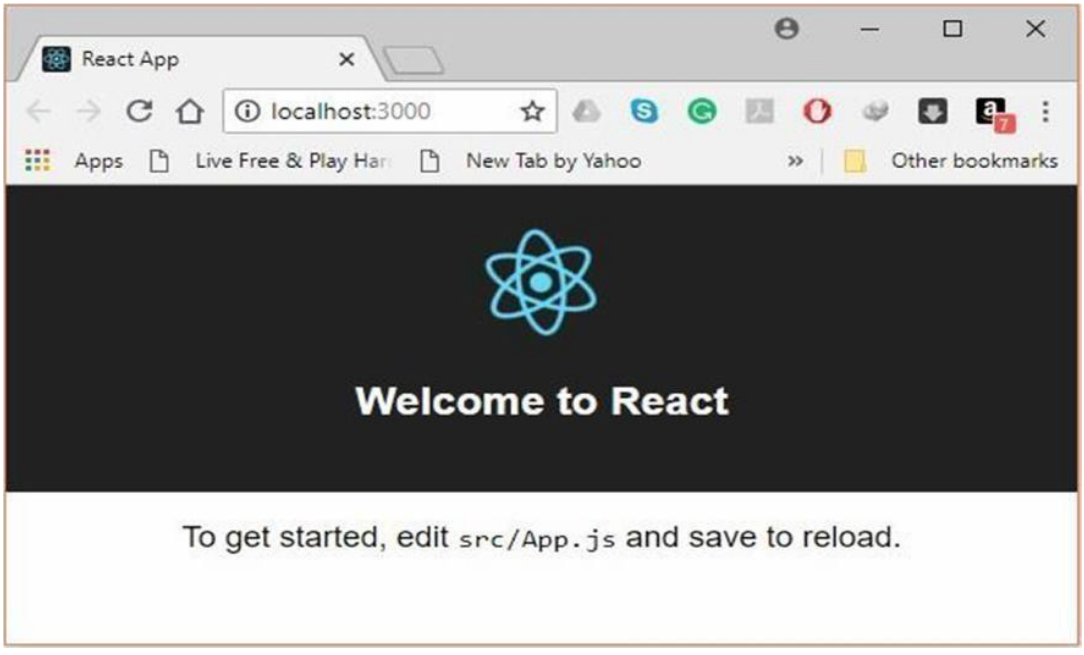
```
  }

  render() {
     return (
       <div>

          <h2>Simple Event Example</h2>

          <label htmlFor="name">Enter company name: </label>

          <input type="text" id="companyName" onChange={this.changeText.bind(this)}/>

          <h4>You entered: { this.state.companyName }</h4>

       </div>

    );

  }

} export default App;
```

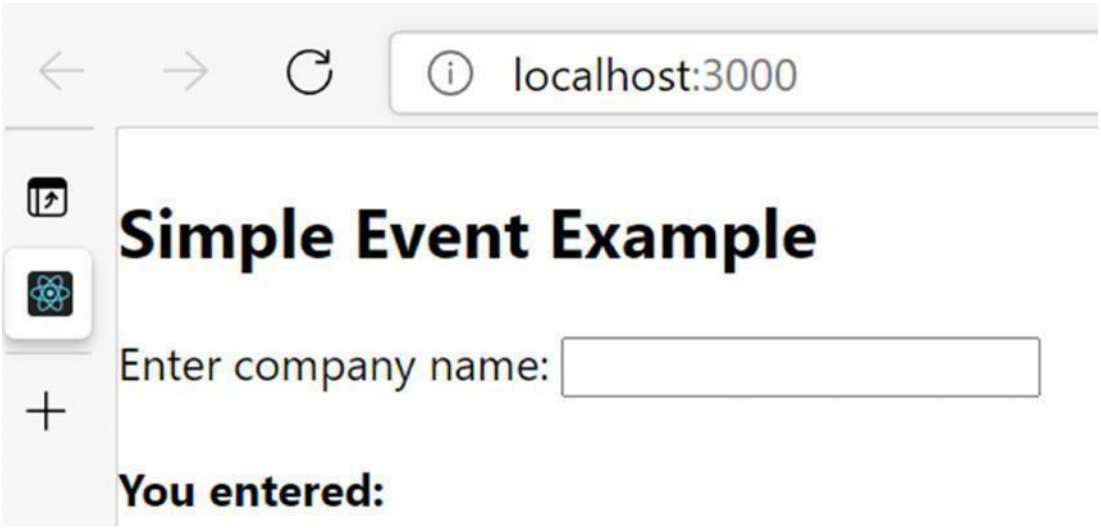
10 .Execute the React Code:

To run the react code you have to type the command on Vs Code Terminal as it is:

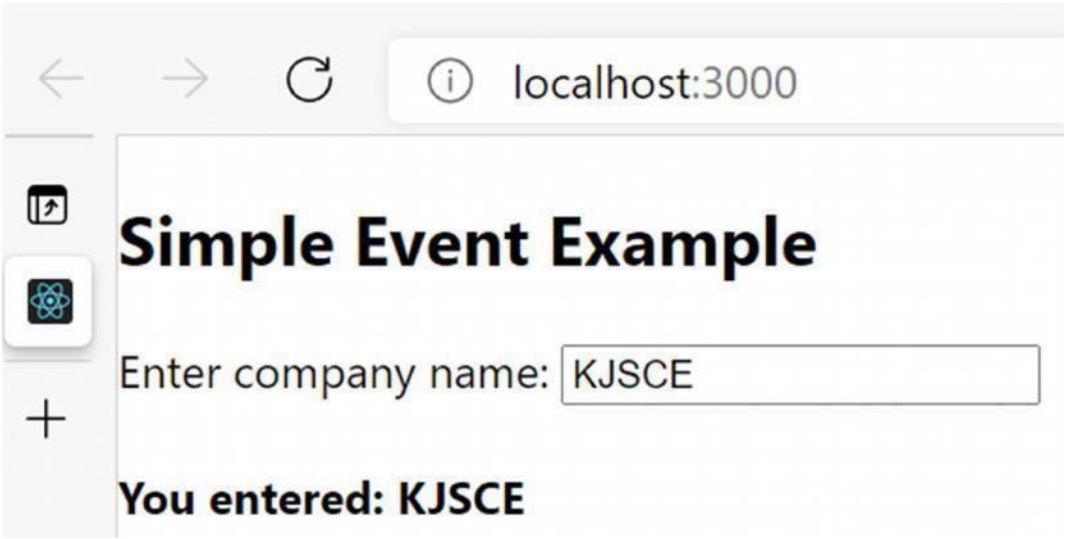
- npm start
- yarn start(if you install yarn utility) To do this on, terminal of VS CODE, C:\Users\Vaibhav\reactapp\myevent\npm start.

By default port number 3000 will start on web browser with the code execution.





11. After running the event the outlook:



Activities:

To design a web page using React JS on your theme to manipulate the DOM elements.

App.jsx

```
import React, { useState } from 'react';
import './App.css'; // Import CSS file

const TicTacToe = () => {
  const [board, setBoard] = useState(Array(9).fill(null)); // Represents the
game board
  const [xIsNext, setXIsNext] = useState(true); // Indicates whether it's X's
turn
  const [winner, setWinner] = useState(null); // Indicates the winner or if it's
a draw
```

```
  // Handles the click event when a square is clicked
  const handleClick = (index) => {
    if (winner || board[index]) return; // Ignore click if there's already a
winner or the square is occupied

    const newBoard = [...board]; // Create a copy of the current board
    newBoard[index] = xIsNext ? 'X' : 'O'; // Place X or O based on whose turn
it is
    setBoard(newBoard);

    // Check for winner or if it's a draw
    const winningLines = [
      [0, 1, 2], [3, 4, 5], [6, 7, 8], // Rows
      [0, 3, 6], [1, 4, 7], [2, 5, 8], // Columns
      [0, 4, 8], [2, 4, 6] // Diagonals
    ];

    for (let line of winningLines) {
      const [a, b, c] = line;
      if (board[a] && board[a] === board[b] && board[a] === board[c]) {
        setWinner(board[a]);
        return;
      }
    }

    // Check for draw
    if (!newBoard.includes(null)) {
      setWinner('Draw');
    }

    // Switch turn
    setXIsNext(!xIsNext);
  };

  // Renders a single square
  const renderSquare = (index) => {
    return (
      <button className="square" onClick={() => handleClick(index)}>
        {board[index] === 'X' && <span className="x">X</span>}
        {board[index] === 'O' && <span className="o">O</span>}
      </button>
    );
  };

  // Renders the game board
  const renderBoard = () => {
    return (
      <div className="board">
        {board.map((square, index) => (
          <div key={index} className="square-container">
            {renderSquare(index)}
```

```
          </div>
        ))}
      </div>
    );
  };

  // Renders the game status
  const renderStatus = () => {
    if (winner) {
      if (winner === 'Draw') {
        return <div className="status">It's a draw!</div>;
      } else {
        return <div className="status">Winner: {winner}</div>;
      }
    } else {
      return <div className="status">Next player: {xIsNext ? 'X' : 'O'}</div>;
    }
  };

  return (
    <div className="tic-tac-toe">
      <h1>Tic Tac Toe</h1>
      {renderStatus()}
      {renderBoard()}
    </div>
  );
};

export default TicTacToe;
```

Index.css

```
:root {
  font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;
  line-height: 1.5;
  font-weight: 400;
  color-scheme: light dark;
  color: rgba(4, 0, 0, 0.87);
  background-color: #242424;
  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

a {
  font-weight: 500;
  color: #050617;
  text-decoration: inherit;
}
a:hover {
```

```
  color: #535bf2;
}

body {
  margin: 0;
  display: flex;
  justify-content: center; /* Center the content horizontally */
  align-items: center; /* Center the content vertically */
  min-width: 320px;
  min-height: 100vh;
  background-color: skyblue; /* Set background color */
}

.tic-tac-toe {
  text-align: center;
}

.board {
  display: grid;
  grid-template-columns: repeat(3, 100px);
  grid-template-rows: repeat(3, 100px);
  gap: 5px;
}

.square {
  width: 100%;
  height: 100%;
  font-size: 2rem;
  display: flex;
  align-items: center;
  justify-content: center;
  border: 1px solid #000; /* Set border color to black */
  background-color: #e86f06;
}

.x,
.o {
  font-size: 3rem;
  line-height: 0.7;
}

.status {
  font-size: 1.5rem;
  margin-bottom: 20px;
  color: #070404; /* Set text color */
}

button {
  border-radius: 8px;
  border: 1px solid transparent;
  padding: 0.6em 1.2em;
  font-size: 1em;
```

```
  font-weight: 500;
  font-family: inherit;
  background-color: #1a1a1a;
  cursor: pointer;
  transition: border-color 0.25s;
  color: #fff; /* Set text color to white */
}
button:hover {
  border-color: #646cff;
}
button:focus,
button:focus-visible {
  outline: 4px auto -webkit-focus-ring-color;
}
```

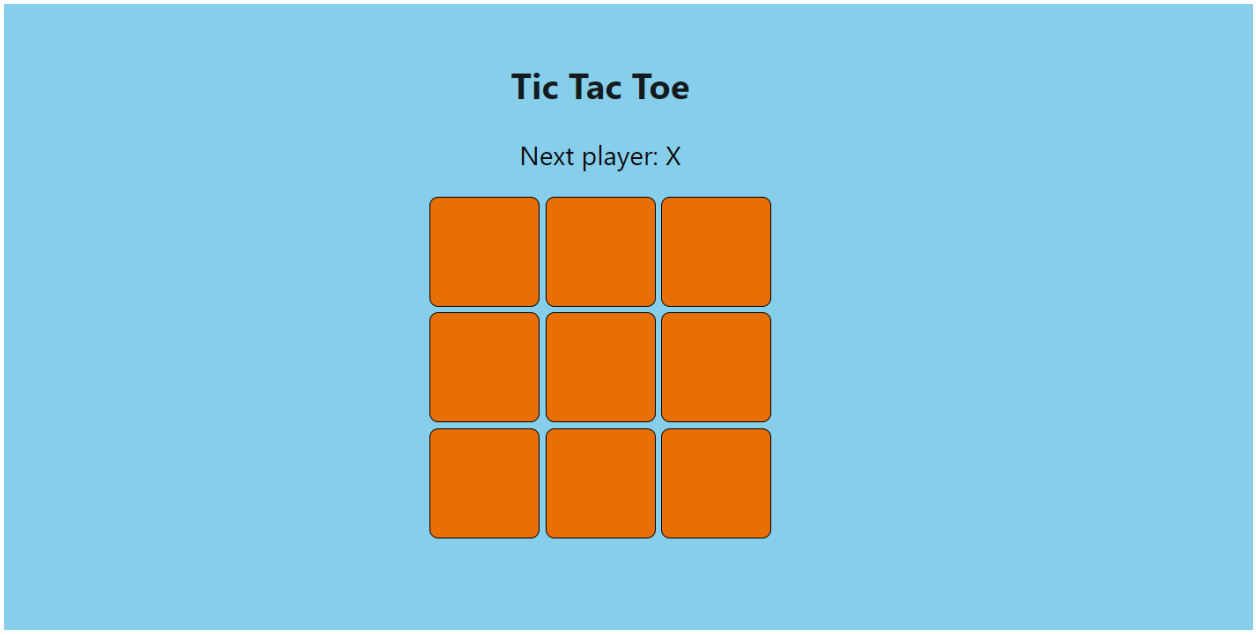
App.css

```
.tic-tac-toe {
  text-align: center;
}

.board {
  display: grid;
  grid-template-columns: repeat(3, 100px);
  grid-template-rows: repeat(3, 100px);
  gap: 5px;
}

.square {
  width: 100%;
  height: 100%;
  font-size: 2rem;
  display: flex;
  align-items: center;
  justify-content: center;
  border: 1px solid #333;
  background-color: #f0f0f0;
}

.x,
.o {
  font-size: 3rem;
  line-height: 0.7;
}

.status {
  font-size: 1.5rem;
  margin-bottom: 20px;
}
```

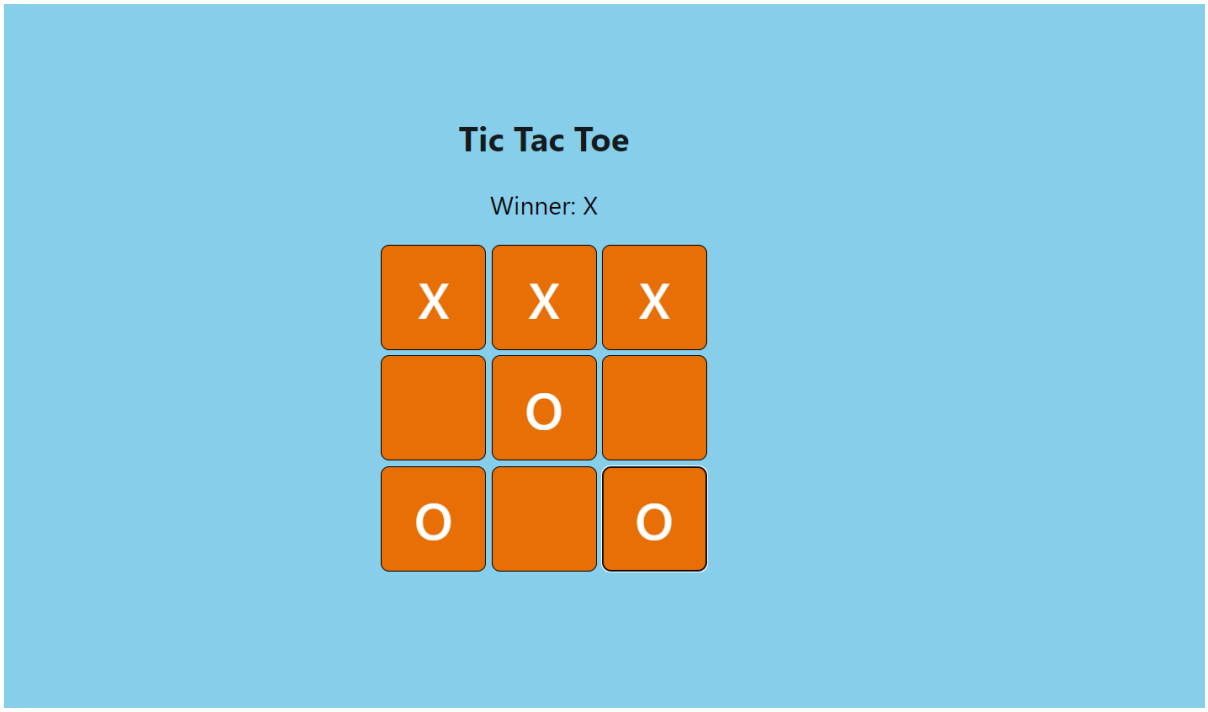
Results: (Document printout as per the format discussed by the faculty *t*)

Display of the designed webpage along with the code.

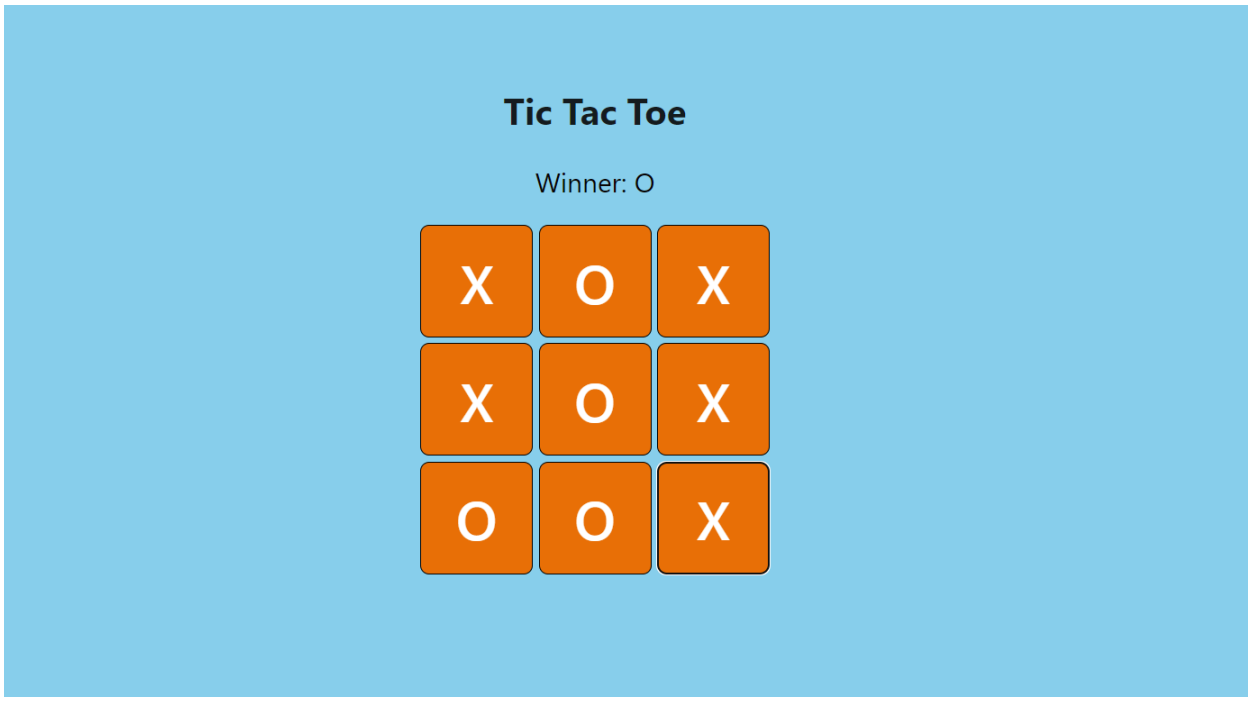
Initial:



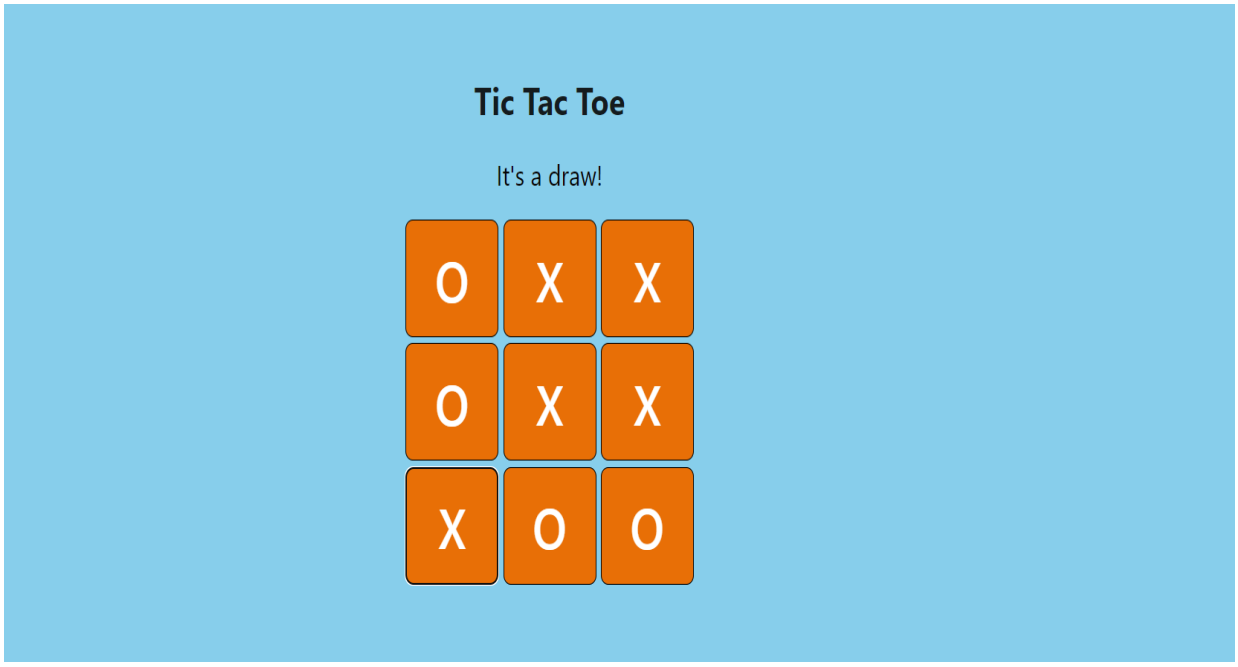
Winner X:



Winner O:



Draw:



.

Questions:

1. **What are the different components of ReactJS?**
2. **What is Virtual DOM? How virtual DOM Works? What is the purpose of render of react DOM?**

Ans1)

React.js is a JavaScript library for building user interfaces, primarily focused on building single-page applications. The different components of React.js include:

1. **Components**: Components are the building blocks of a React application. They are reusable, independent pieces of UI that can contain both structure and behavior. Components can be either functional components (stateless) or class components (stateful).
2. **JSX**: JSX is a syntax extension for JavaScript that allows developers to write HTML-like code within JavaScript. It makes it easier to write and understand the structure of React components.
3. **Props (Properties)**: Props are short for properties. They are used to pass data from a parent component to a child component. Props are read-only and help to make components reusable and dynamic.
4. **State**: State is an object that represents the current condition of a component. It allows components to manage and update their own data. Stateful components use the **useState** hook or class-based **setState** method to manage state.
5. **Lifecycle Methods (in class components)**: Lifecycle methods are special methods that are automatically invoked at various points in a component's lifecycle. They allow developers to perform certain actions at specific times, such as when a component is mounted, updated, or unmounted. With the introduction of React hooks, lifecycle methods are less commonly used in functional components.
6. **Hooks**: Hooks are functions that allow functional components to use state and other React features without writing a class. Some commonly used hooks include **useState**, **useEffect**, **useContext**, **useRef**, etc. They provide a way to reuse stateful logic across components.
7. **Context**: Context provides a way to pass data through the component tree without having to pass props manually at every level. It is often used to share data that is considered global within an application, such as a user's authentication status or theme preferences.
8. **Event Handling**: React allows developers to handle events, such as onClick, onChange, onSubmit, etc., similar to handling events in traditional HTML. Event handlers are typically defined as functions and passed to the appropriate JSX elements.
9. **Refs**: Refs provide a way to access DOM nodes or React elements created in the render method. They are commonly used to interact with DOM elements imperatively or to obtain references to child components.
10. **Fragment**: Fragments allow developers to group multiple children elements without adding extra nodes to the DOM. They are useful when a component needs to return multiple elements, but a wrapping parent element is not necessary.

    These are some of the core concepts and components of React.js, essential for building modern web applications.

Ans2)
The Virtual DOM is a concept in React.js that represents a lightweight copy of the actual DOM (Document Object Model). It's a virtual representation of the UI that React keeps in memory and syncs with the real DOM when necessary. The Virtual DOM is used by React to optimize the updating process and improve performance.
Here's how the Virtual DOM works in React:

1. Initial Render: When a React component is first rendered, it creates a virtual representation of the UI, known as the Virtual DOM. This virtual representation consists of React elements, which are lightweight JavaScript objects.
2. Updating: When a component's state or props change, React re-renders the component and updates the Virtual DOM with the new representation of the UI.
3. Diffing: After updating the Virtual DOM, React performs a process called "diffing" to identify the differences between the new Virtual DOM and the previous one. It calculates the minimum number of changes needed to update the real DOM to match the new Virtual DOM.
4. Reconciliation: Once the differences are identified, React applies the necessary changes to the real DOM through a process called reconciliation. React efficiently updates only the parts of the real DOM that have changed, rather than re-rendering the entire UI.
5. DOM Update: Finally, the real DOM is updated to reflect the changes made in the Virtual DOM. React uses efficient DOM manipulation techniques, such as batch updates and key-based reconciliation, to minimize the impact on performance.

The purpose of the render method in React DOM is to define the UI components and their structure. The render method returns a React element, which describes what the component should look like. React uses this information to construct the initial Virtual DOM and render the component to the real DOM.
In summary, the Virtual DOM in React provides a performant way to manage and update the UI by minimizing the number of DOM manipulations needed. It helps improve the overall efficiency and responsiveness of React applications, especially when dealing with complex UIs and frequent updates.

Outcomes:
CO 4: Implement web application using React JS, Angular JS, JSON and CBOR.

Conclusion: (Conclusion to be based on objectives and outcomes achieved)
Through this experiment we understood that React.js, with its component-based architecture, efficiently manages UI rendering and updates. Leveraging concepts like Virtual DOM, JSX syntax, and hooks, React optimizes performance and enhances developer productivity. By abstracting away direct DOM manipulation and offering a declarative approach, React simplifies the process of building interactive and scalable web applications. Through its ecosystem and core principles, React remains a leading choice for crafting modern, dynamic, and responsive user interfaces.

Grade: AA/AB/BB/BC/CC/CD/DD/FF

Signature of faculty in-charge with date

References: Books/ Journals/ Websites:

1. React – A JavaScript library for building user interfaces (reactjs.org)
2. "React - A JavaScript library for building user interfaces". *React*. Retrieved 7 April 2018.
3. *Krill, Paul (May 15, 2014).* "React: Making faster, smoother UIs for data-driven Web apps". InfoWorld.
4. *Hemel, Zef (June 3, 2013).* "Facebook's React JavaScript User Interfaces Library Receives Mixed Reviews". *InfoQ*.
5. *Dawson, Chris (July 25, 2014).* "JavaScript's History and How it Led To ReactJS". *The New Stack.*
6. *Dere, Mohan (2018-02-19).* "How to integrate create-react-app with all the libraries you need to make a great app". *freeCodeCamp*. Retrieved 2018-06-14.
7. React Tutorial (w3schools.com)
8. ReactJS - Overview - Tutorialspoint
9. ReactJS Tutorials - GeeksforGeeks