**Lesson 05: Create a Simple Tic Tac Toe Game using GitHub Copilot Generative AI**

**Overview:**

You are working as a developer at Tech Innovators Inc. and are tasked with creating a basic React application for a Tic Tac Toe game. To assist with the development, you are utilizing GitHub Copilot as your coding assistant. The project is broken down into several key phases: Project Setup, UI Creation, Game Logic, Bug Fixes, and UI Enhancement.

Throughout each phase, GitHub Copilot provides valuable code suggestions, streamlining the development process and significantly speeding up workflows. By leveraging AI tools like GitHub Copilot, Tech Innovators Inc. is able to efficiently complete the Tic Tac Toe game, demonstrating the power of AI in software development.

**Instructions:**

1. Initiate the project with GitHub Copilot to generate a React sample project
2. Collaborate with Copilot to create a basic Tic Tac Toe game, including UI design and game logic implementation
3. Update the CSS by generating a new script through ChatGPT to fix the header and footer to the top and bottom of the screen, respectively

**Tasks:**

1. Initialize the GitHub Copilot project to create a React sample project:
   1. Set up the workspace and open the generated React project
   2. Ensure the project environment is configured properly
2. Create a basic Tic Tac Toe Game using GitHub Copilot:
   1. Use prompts to create the basic user interface
   2. Use prompts to develop the basic game logic and determine when a   
      player wins
   3. Use prompts to enhance the board’s user interface aesthetics

**Tools required:**

1. GitHub Copilot
2. Node JS
3. Visual Studio Code

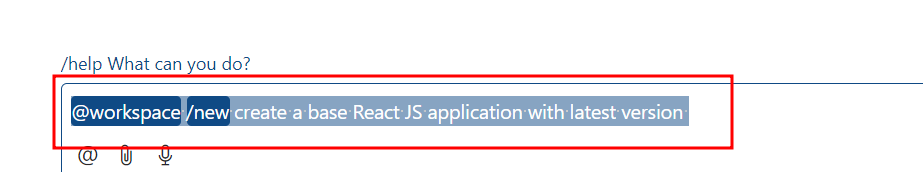
**Guided Practice Solution**

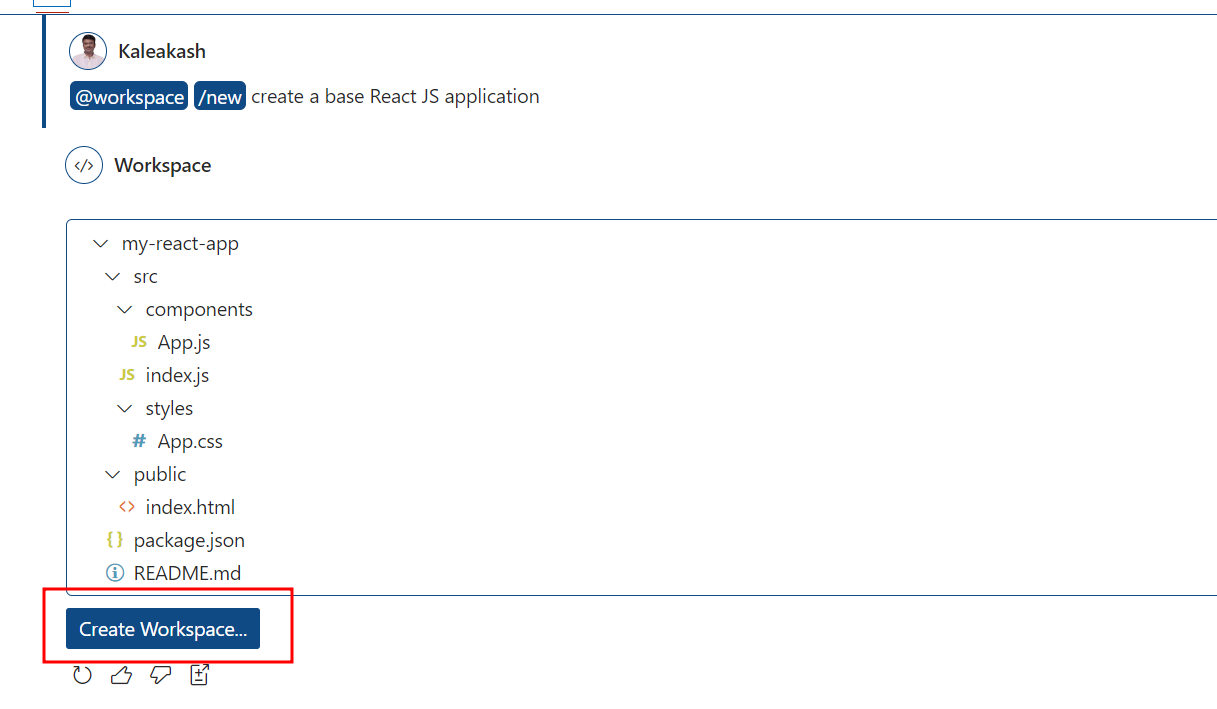
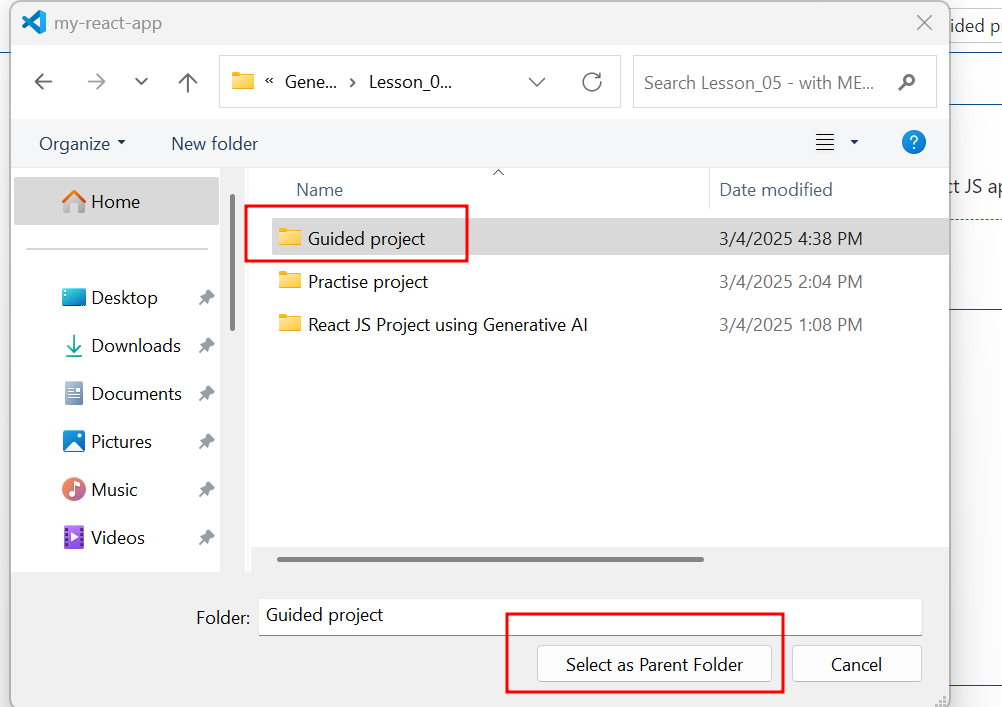
**Disclaimer:** Please note that all the GenAI tools used in this exercise can produce varied outputs even when presented with similar prompts. Thus, you may get different output for the same prompt.

**Task 1: Initialize the GitHub Copilot project to create a React sample project**

**Step 1: Set up the workspace and open the generated React project**

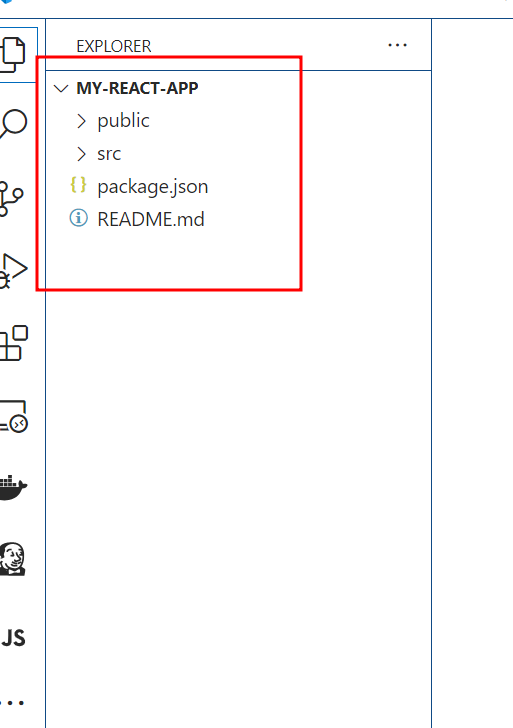
* 1. Navigate to Visual Studio Code and use the following prompt in GitHub Copilot to generate a basic React JS application:  
     **@workspace /new** **create a base React JS application**  
     @workspace /new create a base React JS application with latest version



* 1. Create the workspace by clicking on **Create Workspace**
  2. Choose the folder for your parent folder and then click on **Select as Parent Folder**

**Step 2: Ensure the project environment is configured properly**

2.1 open the created work space in VS Code editor.



2.2 open the command prompt and run the command as npm install which installed all required dependencies to run the react js project



* 1. Once the project runs successfully and displays the success message, click on the Local link, as shown in the screenshot below:
  2. **App.js file code** part of components folder

import React from 'react';

import '../styles/App.css';

const App = () => {

    return (

        <div className="App">

            <h1>Welcome to My React App</h1>

            {/\* Additional components or logic can be added here \*/}

        </div>

    );

};

export default App;

2.5 **App.css** file code part of styles folder

/\* App.css \*/

.App {

    text-align: center;

}

.App-header {

    background-color: #282c34;

    min-height: 100vh;

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: center;

    font-size: calc(10px + 2vmin);

    color: white;

}

.App-link {

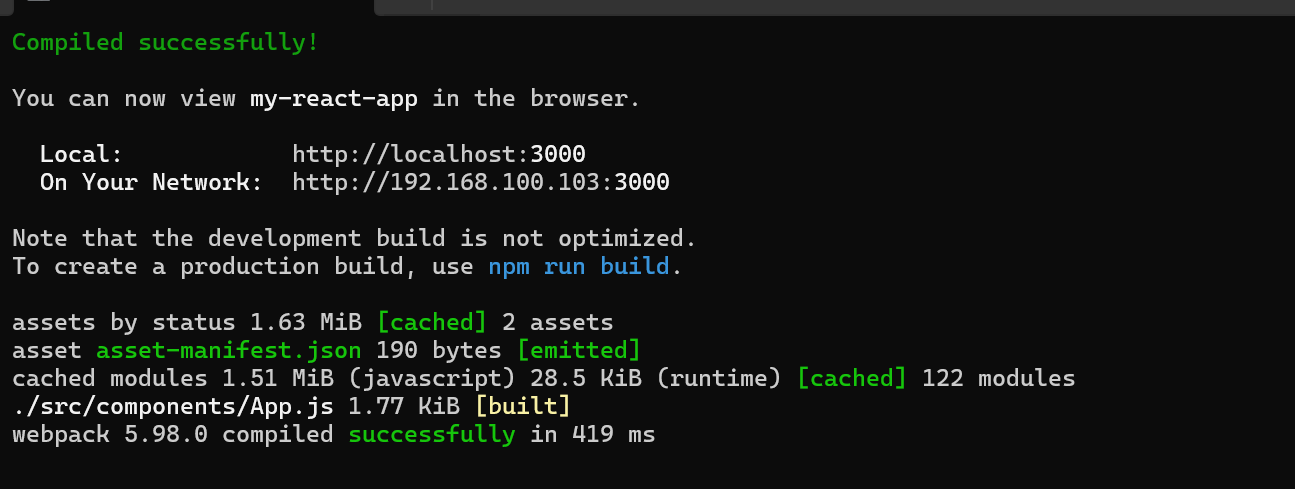
    color: #61dafb;

}

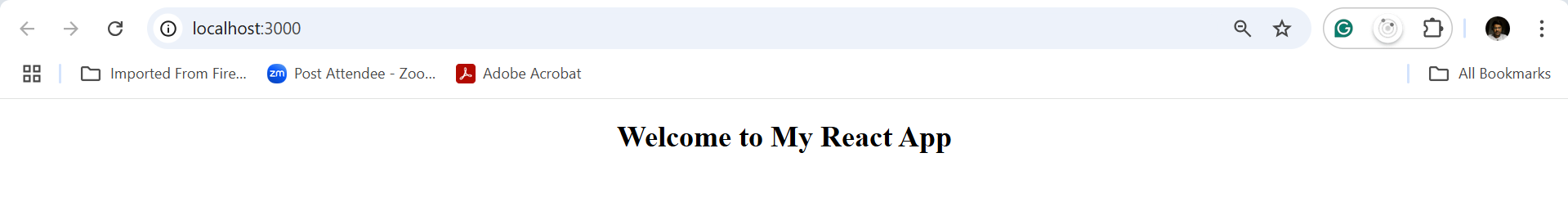
2.6 now run the project using the command as

**npm start**

Out can see the output as



On browser

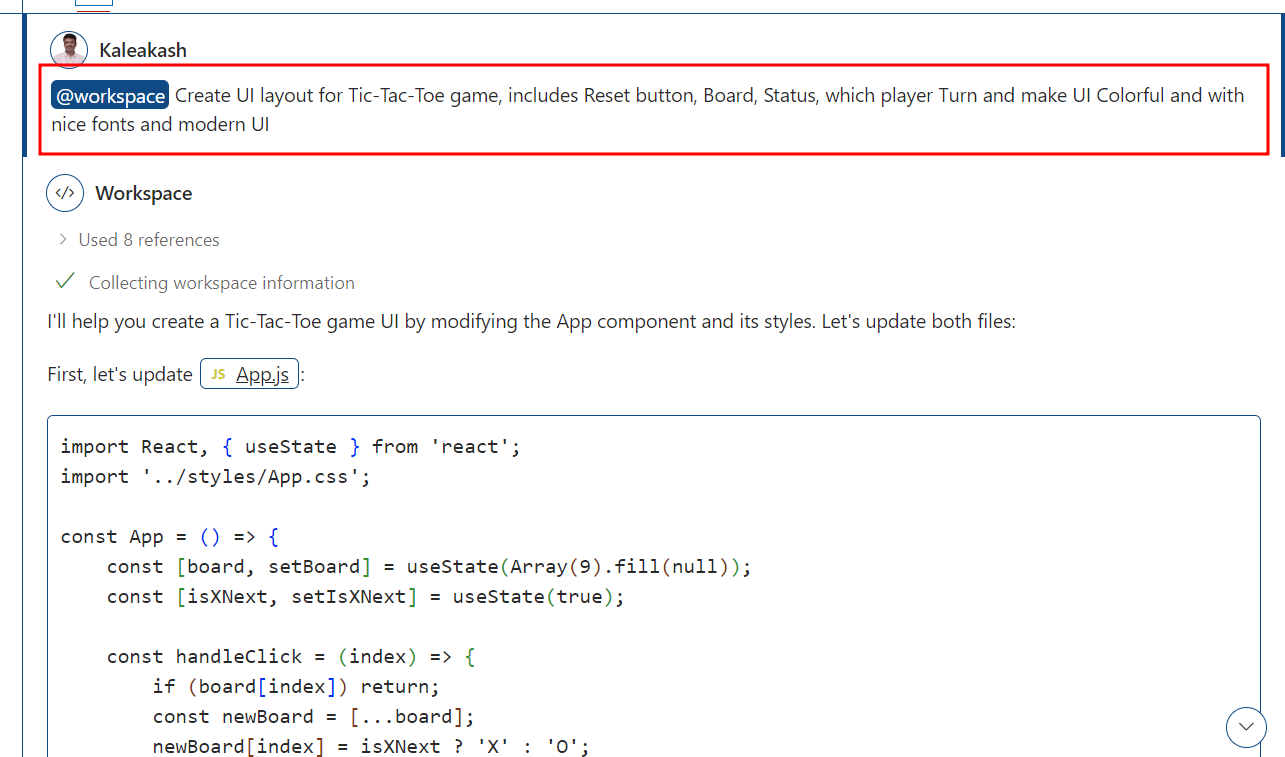


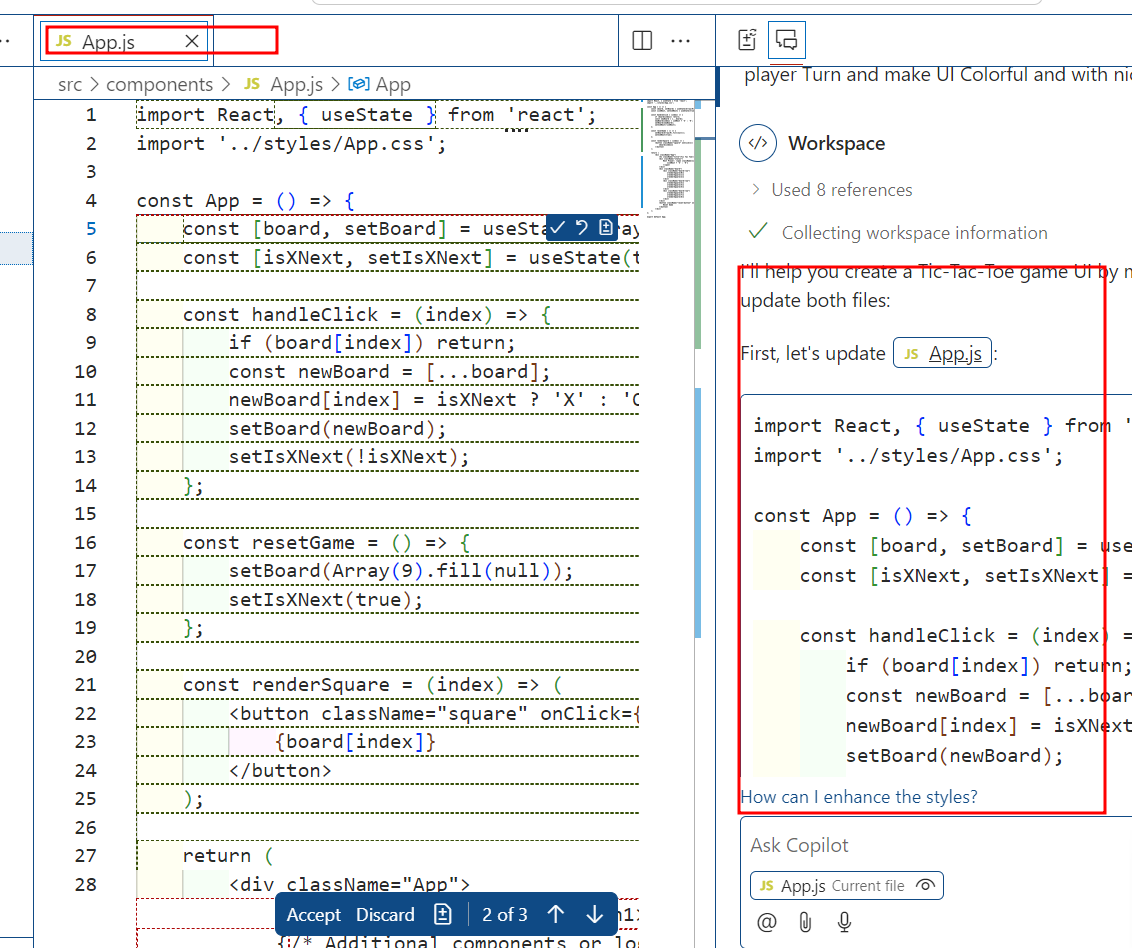
**Task 2: Create Basic Tic-Tac-Toe Game Using GitHub Copilot**

**Step 2: Use prompts to create the basic user interface**

* 1. Use the following prompt to generate a basic UI for the Tic-Tac-Toe game:

**Create UI layout for Tic-Tac-Toe game, includes Reset button, Board, Status, which player Turn and make UI Colorful and with nice fonts and modern UI**





2.2 So, apply this generated code to App.js file

App.js file code

import React, { useState } from 'react';

import '../styles/App.css';

const App = () => {

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

    const [isXNext, setIsXNext] = useState(true);

    const handleClick = (index) => {

        if (board[index]) return;

        const newBoard = [...board];

        newBoard[index] = isXNext ? 'X' : 'O';

        setBoard(newBoard);

        setIsXNext(!isXNext);

    };

    const resetGame = () => {

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

        setIsXNext(true);

    };

    const renderSquare = (index) => (

        <button className="square" onClick={() => handleClick(index)}>

            {board[index]}

        </button>

    );

    return (

        <div className="App">

            <h1 className="title">Tic Tac Toe</h1>

            <div className="status">

                Next Player: <span className={isXNext ? 'x-player' : 'o-player'}>

                    {isXNext ? 'X' : 'O'}

                </span>

            </div>

            <div className="board">

                <div className="board-row">

                    {renderSquare(0)}

                    {renderSquare(1)}

                    {renderSquare(2)}

                </div>

                <div className="board-row">

                    {renderSquare(3)}

                    {renderSquare(4)}

                    {renderSquare(5)}

                </div>

                <div className="board-row">

                    {renderSquare(6)}

                    {renderSquare(7)}

                    {renderSquare(8)}

                </div>

            </div>

            <button className="reset-button" onClick={resetGame}>

                Reset Game

            </button>

        </div>

    );

};

export default App;

2.3 Even apply App.css file generated code to existing App.css file

**App.css**

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap');

/\* App.css \*/

.App {

    text-align: center;

    min-height: 100vh;

    background: linear-gradient(135deg, #1a1a2e, #16213e);

    padding: 2rem;

    font-family: 'Poppins', sans-serif;

}

.App-header {

    background-color: #282c34;

    min-height: 100vh;

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: center;

    font-size: calc(10px + 2vmin);

    color: white;

}

.App-link {

    color: #61dafb;

}

.title {

    color: #fff;

    font-size: 3rem;

    margin-bottom: 2rem;

    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);

}

.status {

    color: #fff;

    font-size: 1.5rem;

    margin-bottom: 2rem;

}

.x-player {

    color: #00fff5;

    font-weight: bold;

}

.o-player {

    color: #ff6b6b;

    font-weight: bold;

}

.board {

    display: inline-block;

    background: rgba(255, 255, 255, 0.1);

    padding: 1rem;

    border-radius: 15px;

    box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37);

    backdrop-filter: blur(4px);

}

.board-row {

    display: flex;

}

.square {

    width: 80px;

    height: 80px;

    background: rgba(255, 255, 255, 0.05);

    border: 2px solid rgba(255, 255, 255, 0.1);

    margin: 4px;

    border-radius: 10px;

    font-size: 2.5rem;

    font-weight: bold;

    color: #fff;

    cursor: pointer;

    transition: all 0.3s ease;

    font-family: 'Poppins', sans-serif;

}

.square:hover {

    background: rgba(255, 255, 255, 0.1);

    transform: scale(1.05);

}

.square:focus {

    outline: none;

}

.reset-button {

    margin-top: 2rem;

    padding: 1rem 2rem;

    font-size: 1.2rem;

    background: #4CAF50;

    color: white;

    border: none;

    border-radius: 25px;

    cursor: pointer;

    transition: all 0.3s ease;

    font-family: 'Poppins', sans-serif;

    box-shadow: 0 4px 15px rgba(76, 175, 80, 0.3);

}

.reset-button:hover {

    background: #45a049;

    transform: translateY(-2px);

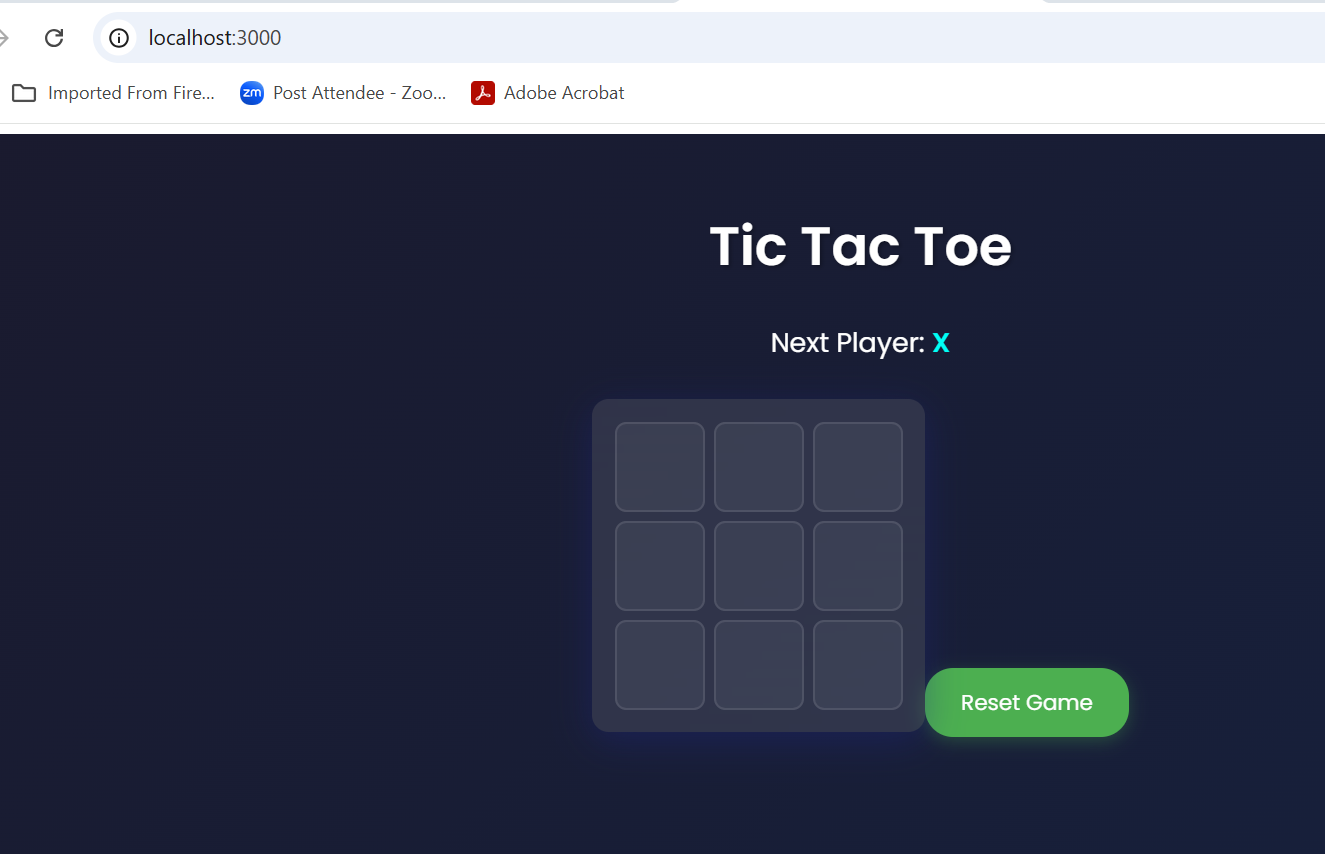
}

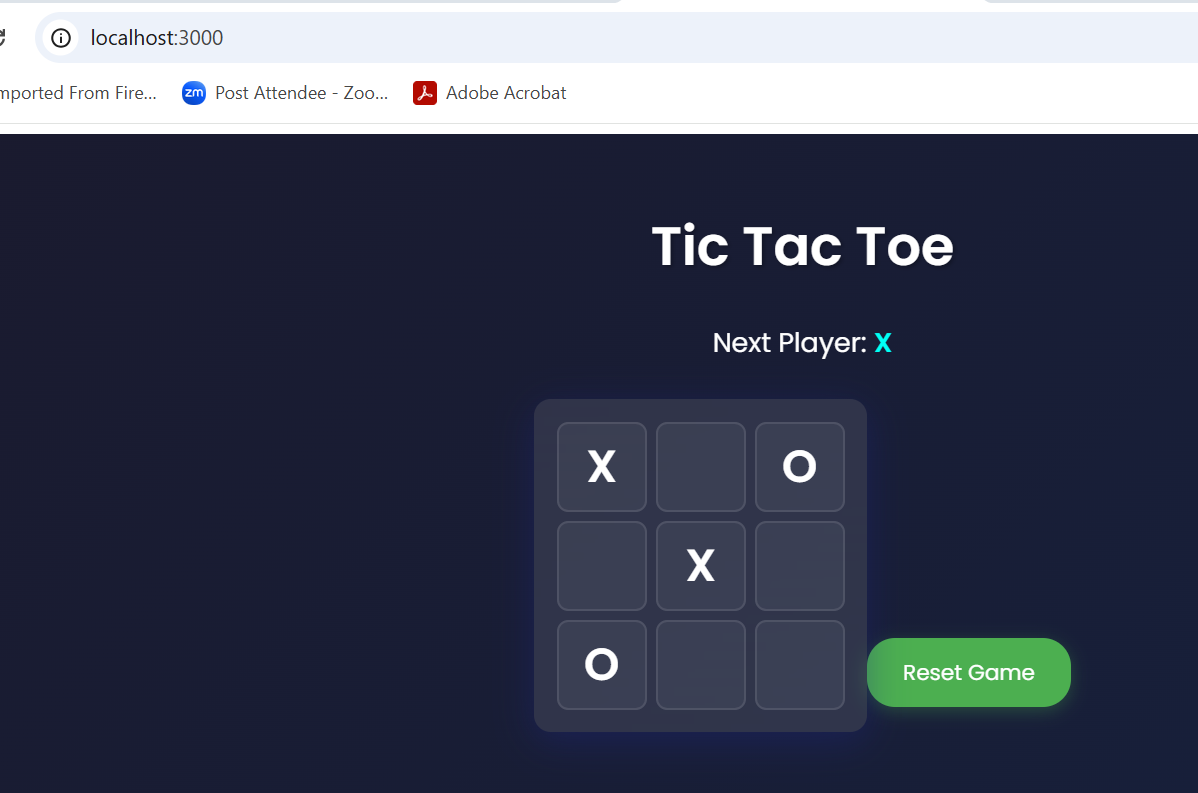
.reset-button:active {

    transform: translateY(0);

}

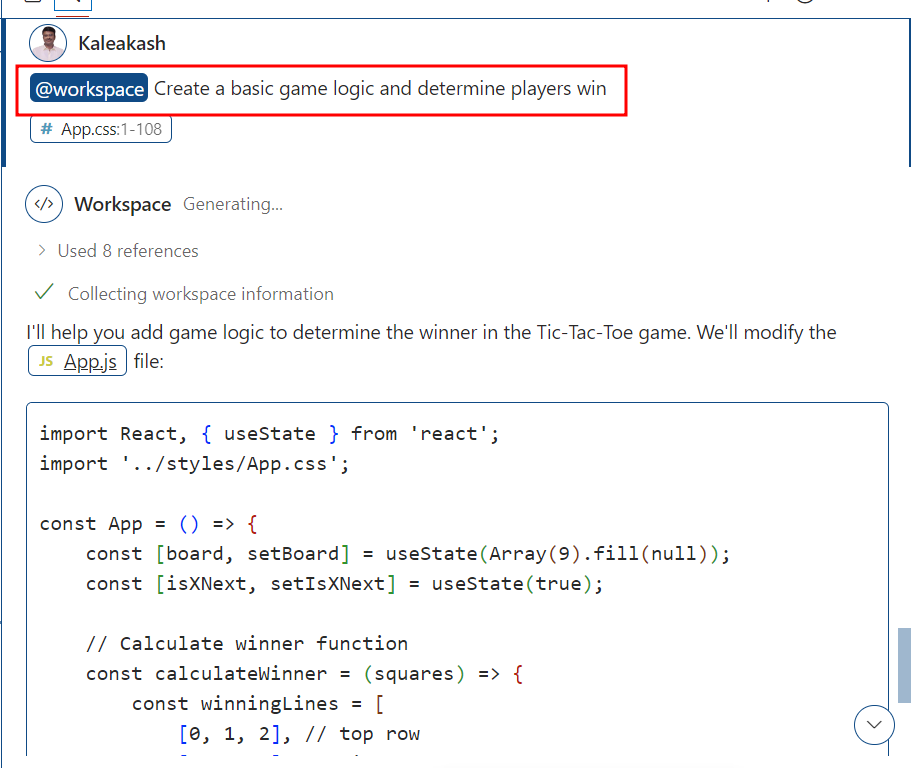
**2.4** The output after executing the basic Tic-Tac-Toe game is as follows:



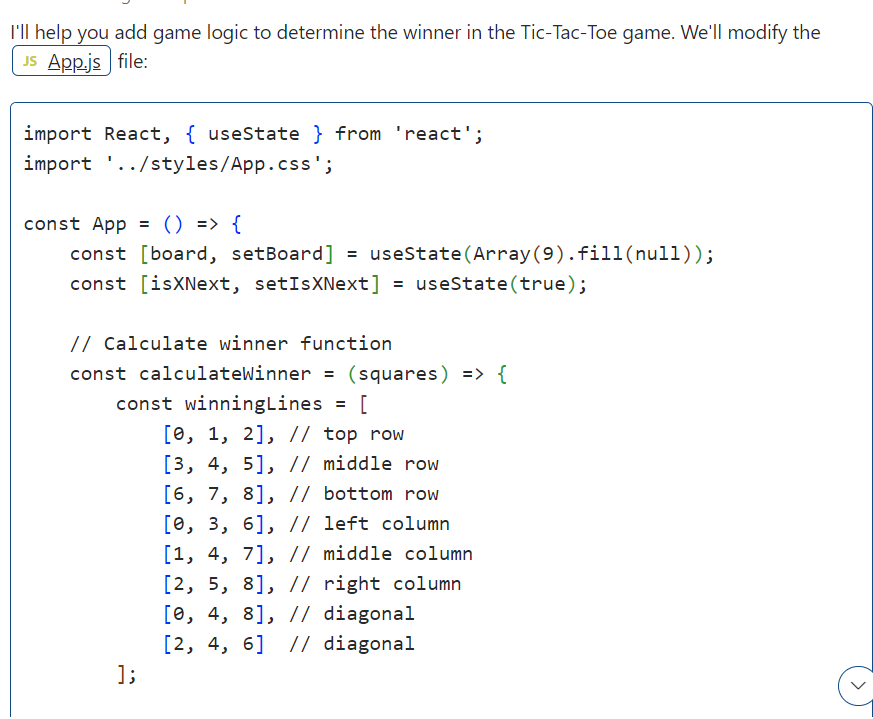


**Step 3: Use prompts to develop the basic game logic and determine when a   
 player wins**

* 1. Use the following prompt to create a basic game logic and determine when a   
     player wins:  
     **@workspace Create a basic game logic and determine players win**



The response from GitHub Copilot is as follows:



* 1. Add the following prompt inside App.js to enhance the user experience:

**App.js file code**

import React, { useState } from 'react';

import '../styles/App.css';

const App = () => {

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

    const [isXNext, setIsXNext] = useState(true);

    // Calculate winner function

    const calculateWinner = (squares) => {

        const winningLines = [

            [0, 1, 2], // top row

            [3, 4, 5], // middle row

            [6, 7, 8], // bottom row

            [0, 3, 6], // left column

            [1, 4, 7], // middle column

            [2, 5, 8], // right column

            [0, 4, 8], // diagonal

            [2, 4, 6]  // diagonal

        ];

        for (let line of winningLines) {

            const [a, b, c] = line;

            if (squares[a] && squares[a] === squares[b] && squares[a] === squares[c]) {

                return squares[a];

            }

        }

        return null;

    };

    // Check if game is draw

    const isDraw = (squares) => {

        return squares.every(square => square !== null);

    };

    const handleClick = (index) => {

        if (board[index] || calculateWinner(board)) return;

        const newBoard = [...board];

        newBoard[index] = isXNext ? 'X' : 'O';

        setBoard(newBoard);

        setIsXNext(!isXNext);

    };

    const resetGame = () => {

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

        setIsXNext(true);

    };

    const renderSquare = (index) => (

        <button className="square" onClick={() => handleClick(index)}>

            {board[index]}

        </button>

    );

    const winner = calculateWinner(board);

    const gameIsDraw = isDraw(board);

    let status;

    if (winner) {

        status = `Winner: ${winner}`;

    } else if (gameIsDraw) {

        status = "Game is a Draw!";

    } else {

        status = `Next Player: ${isXNext ? 'X' : 'O'}`;

    }

    return (

        <div className="App">

            <h1 className="title">Tic Tac Toe</h1>

            <div className="status">

                <span className={winner ? (winner === 'X' ? 'x-player' : 'o-player') : (isXNext ? 'x-player' : 'o-player')}>

                    {status}

                </span>

            </div>

            <div className="board">

                <div className="board-row">

                    {renderSquare(0)}

                    {renderSquare(1)}

                    {renderSquare(2)}

                </div>

                <div className="board-row">

                    {renderSquare(3)}

                    {renderSquare(4)}

                    {renderSquare(5)}

                </div>

                <div className="board-row">

                    {renderSquare(6)}

                    {renderSquare(7)}

                    {renderSquare(8)}

                </div>

            </div>

            <button className="reset-button" onClick={resetGame}>

                Reset Game

            </button>

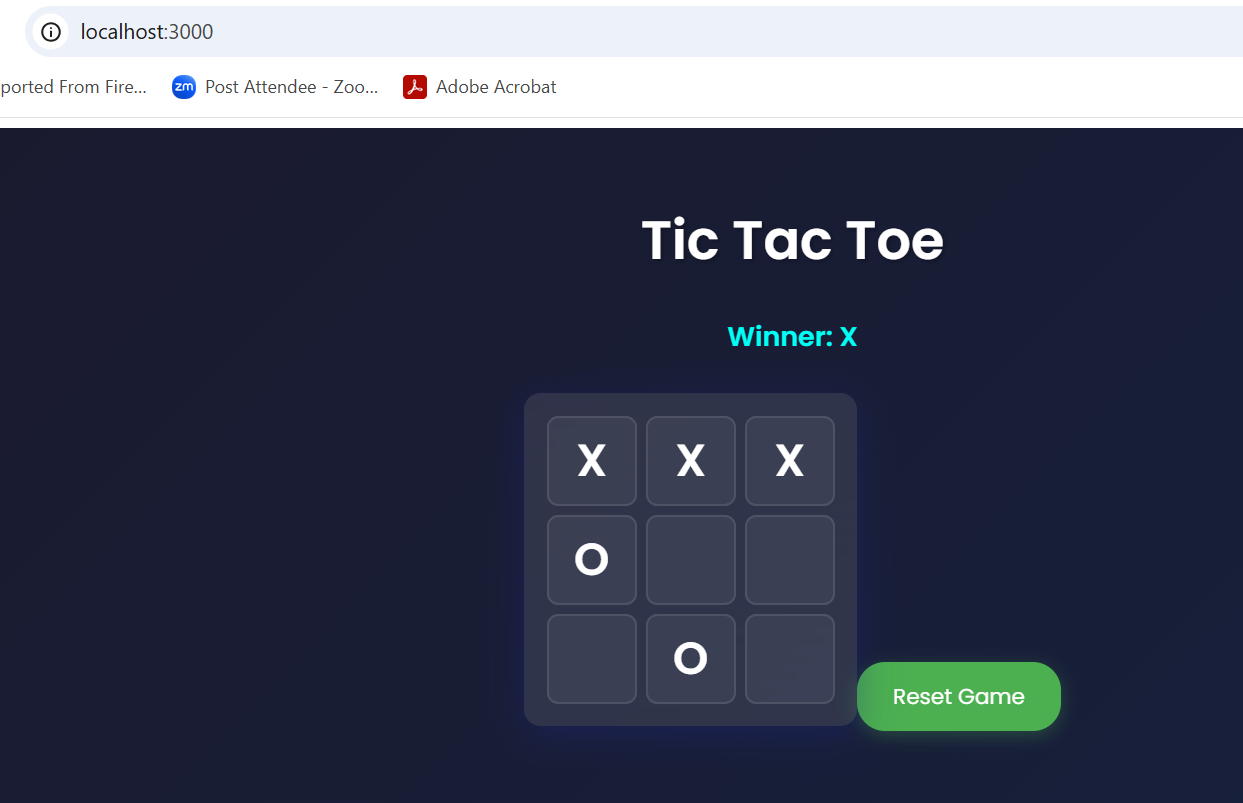
        </div>

    );

};

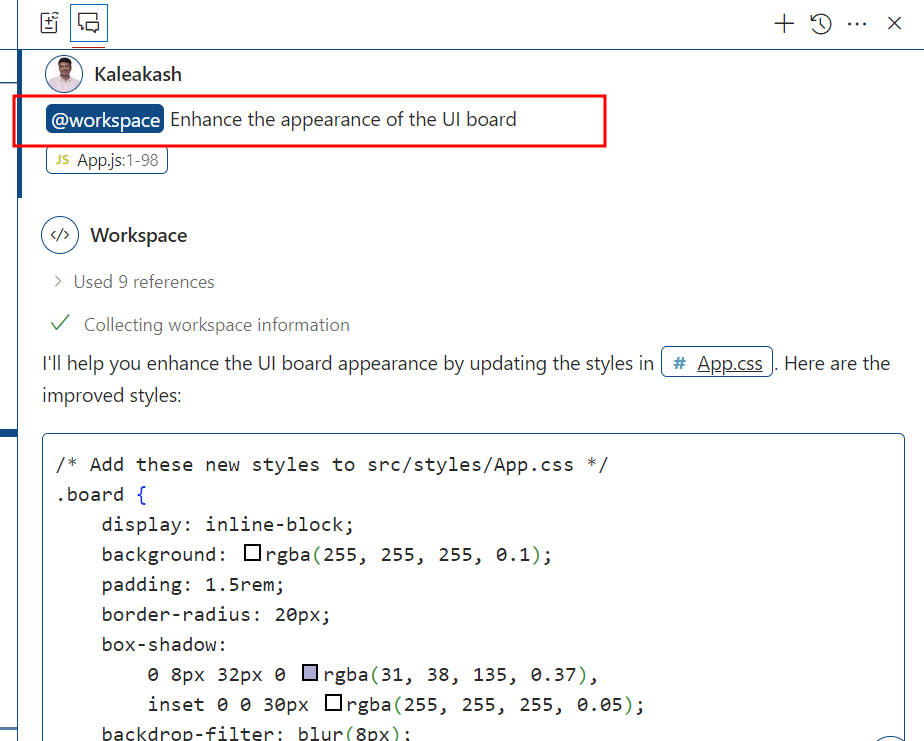
export default App;

**After run the application output as**



**Step 3: Use prompts to enhance the board’s user interface aesthetics**

* 1. Use the following prompt to enhance the appearance of the UI board:  
     **@workspace Enhance the appearance of the UI board**



App.css file code

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap');

/\* App.css \*/

.App {

    text-align: center;

    min-height: 100vh;

    background: linear-gradient(135deg, #1a1a2e, #16213e);

    padding: 2rem;

    font-family: 'Poppins', sans-serif;

}

.App-header {

    background-color: #282c34;

    min-height: 100vh;

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: center;

    font-size: calc(10px + 2vmin);

    color: white;

}

.App-link {

    color: #61dafb;

}

.title {

    color: #fff;

    font-size: 3rem;

    margin-bottom: 2rem;

    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);

}

.status {

    color: #fff;

    font-size: 1.5rem;

    margin-bottom: 2rem;

}

.x-player {

    color: #00fff5;

    font-weight: bold;

}

.o-player {

    color: #ff6b6b;

    font-weight: bold;

}

.board {

    display: inline-block;

    background: rgba(255, 255, 255, 0.1);

    padding: 1.5rem;

    border-radius: 20px;

    box-shadow:

        0 8px 32px 0 rgba(31, 38, 135, 0.37),

        inset 0 0 30px rgba(255, 255, 255, 0.05);

    backdrop-filter: blur(8px);

    border: 1px solid rgba(255, 255, 255, 0.1);

    transform: perspective(1000px) rotateX(5deg);

    transition: transform 0.3s ease;

}

.board:hover {

    transform: perspective(1000px) rotateX(0deg);

}

.board-row {

    display: flex;

    gap: 8px;

    margin-bottom: 8px;

}

.board-row:last-child {

    margin-bottom: 0;

}

.square {

    width: 90px;

    height: 90px;

    background: linear-gradient(

        145deg,

        rgba(255, 255, 255, 0.05),

        rgba(0, 0, 0, 0.1)

    );

    border: 2px solid rgba(255, 255, 255, 0.1);

    border-radius: 15px;

    font-size: 2.8rem;

    font-weight: bold;

    color: #fff;

    cursor: pointer;

    transition: all 0.3s ease;

    font-family: 'Poppins', sans-serif;

    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);

    box-shadow:

        0 4px 15px rgba(0, 0, 0, 0.2),

        inset 0 0 15px rgba(255, 255, 255, 0.05);

}

.square:hover {

    background: linear-gradient(

        145deg,

        rgba(255, 255, 255, 0.1),

        rgba(0, 0, 0, 0.15)

    );

    transform: translateY(-2px);

    box-shadow:

        0 8px 20px rgba(0, 0, 0, 0.3),

        inset 0 0 15px rgba(255, 255, 255, 0.1);

}

.square:focus {

    outline: none;

}

.square:active {

    transform: translateY(0);

}

.reset-button {

    margin-top: 2rem;

    padding: 1rem 2rem;

    font-size: 1.2rem;

    background: #4CAF50;

    color: white;

    border: none;

    border-radius: 25px;

    cursor: pointer;

    transition: all 0.3s ease;

    font-family: 'Poppins', sans-serif;

    box-shadow: 0 4px 15px rgba(76, 175, 80, 0.3);

}

.reset-button:hover {

    background: #45a049;

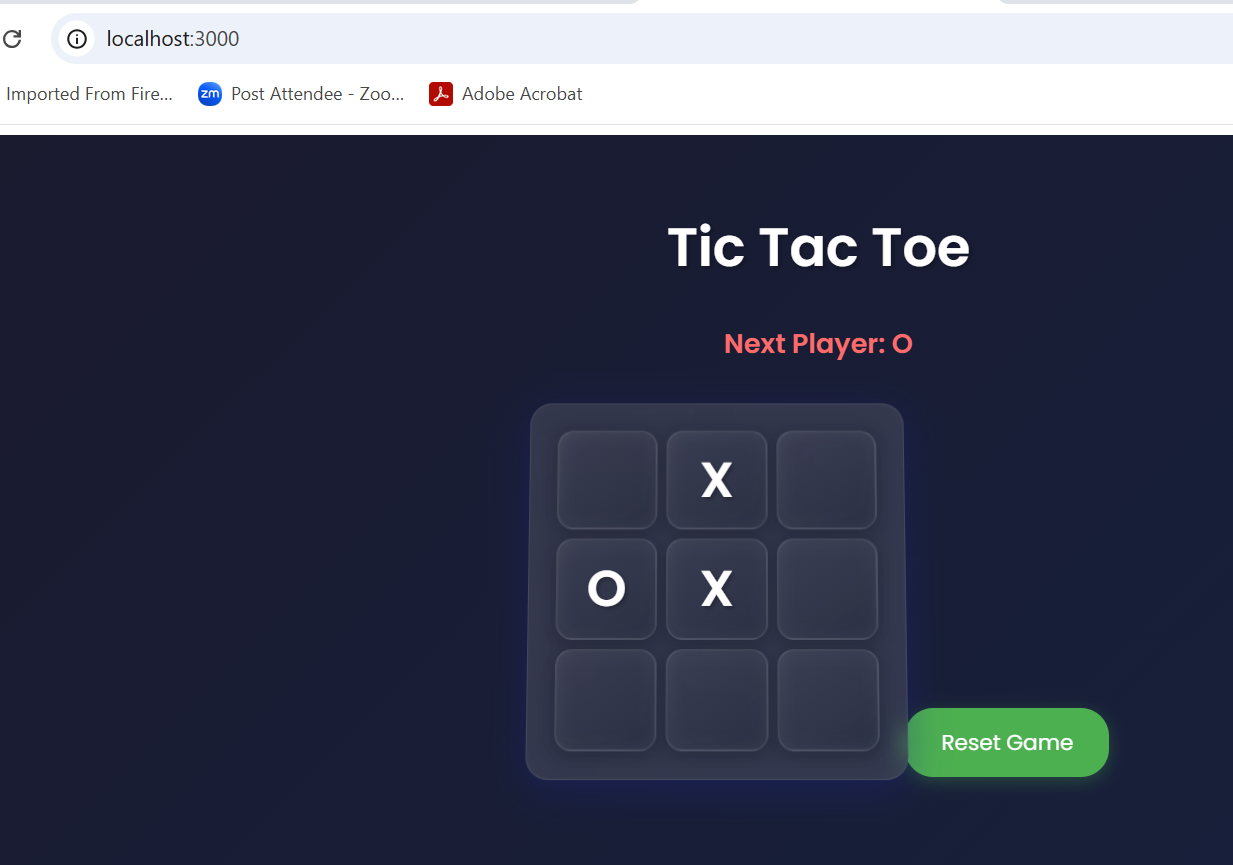
    transform: translateY(-2px);

}

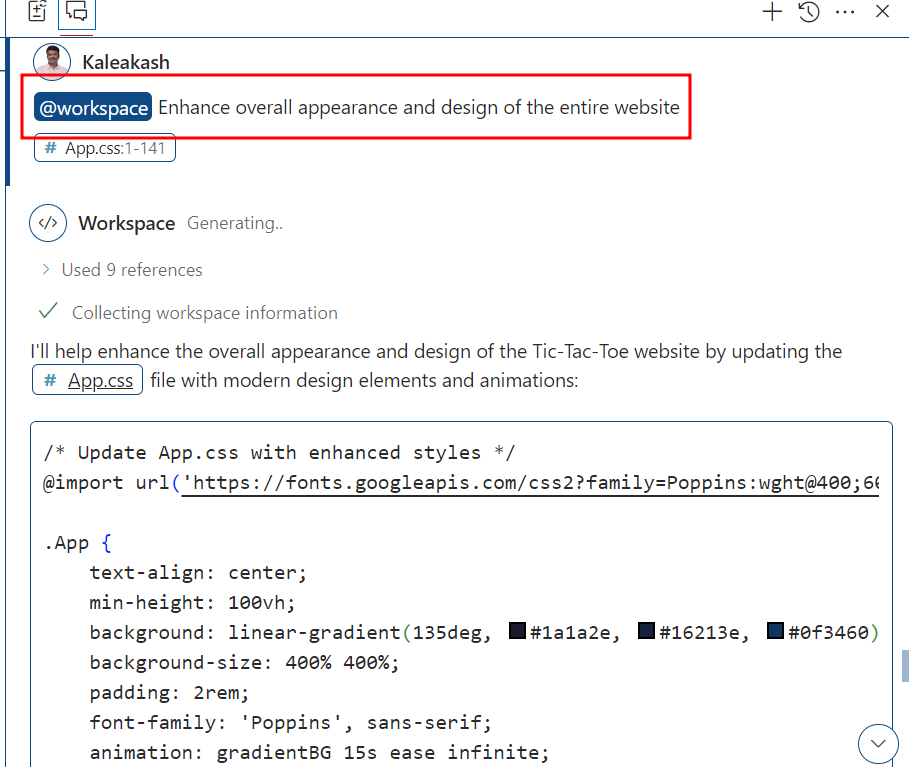
.reset-button:active {

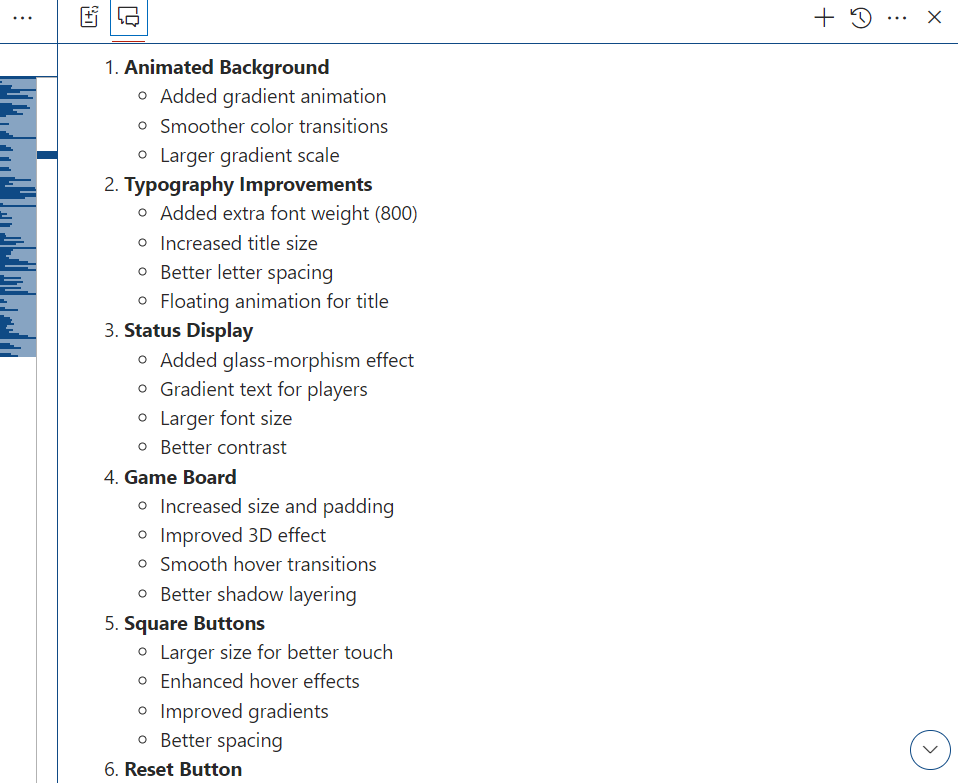
    transform: translateY(0);

}



* 1. Use the following prompt to enhance the overall appearance and design of the website:  
     **@workspace Enhance overall appearance and design of the entire website**





App.js final code

import React, { useState } from 'react';

import '../styles/App.css';

const App = () => {

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

    const [isXNext, setIsXNext] = useState(true);

    // Calculate winner function

    const calculateWinner = (squares) => {

        const winningLines = [

            [0, 1, 2], // top row

            [3, 4, 5], // middle row

            [6, 7, 8], // bottom row

            [0, 3, 6], // left column

            [1, 4, 7], // middle column

            [2, 5, 8], // right column

            [0, 4, 8], // diagonal

            [2, 4, 6]  // diagonal

        ];

        for (let line of winningLines) {

            const [a, b, c] = line;

            if (squares[a] && squares[a] === squares[b] && squares[a] === squares[c]) {

                return squares[a];

            }

        }

        return null;

    };

    // Check if game is draw

    const isDraw = (squares) => {

        return squares.every(square => square !== null);

    };

    const handleClick = (index) => {

        if (board[index] || calculateWinner(board)) return;

        const newBoard = [...board];

        newBoard[index] = isXNext ? 'X' : 'O';

        setBoard(newBoard);

        setIsXNext(!isXNext);

    };

    const resetGame = () => {

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

        setIsXNext(true);

    };

    const renderSquare = (index) => (

        <button className="square" onClick={() => handleClick(index)}>

            {board[index]}

        </button>

    );

    const winner = calculateWinner(board);

    const gameIsDraw = isDraw(board);

    let status;

    if (winner) {

        status = `Winner: ${winner}`;

    } else if (gameIsDraw) {

        status = "Game is a Draw!";

    } else {

        status = `Next Player: ${isXNext ? 'X' : 'O'}`;

    }

    return (

        <div className="App">

            <h1 className="title">Tic Tac Toe</h1>

            <div className="status">

                <span className={winner ? (winner === 'X' ? 'x-player' : 'o-player') : (isXNext ? 'x-player' : 'o-player')}>

                    {status}

                </span>

            </div>

            <div className="board">

                <div className="board-row">

                    {renderSquare(0)}

                    {renderSquare(1)}

                    {renderSquare(2)}

                </div>

                <div className="board-row">

                    {renderSquare(3)}

                    {renderSquare(4)}

                    {renderSquare(5)}

                </div>

                <div className="board-row">

                    {renderSquare(6)}

                    {renderSquare(7)}

                    {renderSquare(8)}

                </div>

            </div>

            <button className="reset-button" onClick={resetGame}>

                Reset Game

            </button>

        </div>

    );

};

export default App;

**App.css file code**

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600;800&display=swap');

/\* App.css \*/

.App {

    text-align: center;

    min-height: 100vh;

    background: linear-gradient(135deg, #1a1a2e, #16213e, #0f3460);

    background-size: 400% 400%;

    padding: 2rem;

    font-family: 'Poppins', sans-serif;

    animation: gradientBG 15s ease infinite;

}

@keyframes gradientBG {

    0% { background-position: 0% 50% }

    50% { background-position: 100% 50% }

    100% { background-position: 0% 50% }

}

.App-header {

    background-color: #282c34;

    min-height: 100vh;

    display: flex;

    flex-direction: column;

    align-items: center;

    justify-content: center;

    font-size: calc(10px + 2vmin);

    color: white;

}

.App-link {

    color: #61dafb;

}

.title {

    color: #fff;

    font-size: 4rem;

    font-weight: 800;

    margin-bottom: 2rem;

    text-shadow:

        2px 2px 4px rgba(0, 0, 0, 0.3),

        0 0 20px rgba(255, 255, 255, 0.1);

    letter-spacing: 3px;

    animation: titleFloat 3s ease-in-out infinite;

}

@keyframes titleFloat {

    0%, 100% { transform: translateY(0); }

    50% { transform: translateY(-10px); }

}

.status {

    color: #fff;

    font-size: 1.8rem;

    margin-bottom: 3rem;

    padding: 1rem 2rem;

    background: rgba(255, 255, 255, 0.05);

    border-radius: 15px;

    backdrop-filter: blur(10px);

    display: inline-block;

}

.x-player {

    background: linear-gradient(45deg, #00fff5, #0099ff);

    -webkit-background-clip: text;

    background-clip: text;

    color: transparent;

    font-weight: bold;

    text-shadow: none;

}

.o-player {

    background: linear-gradient(45deg, #ff6b6b, #ff3366);

    -webkit-background-clip: text;

    background-clip: text;

    color: transparent;

    font-weight: bold;

    text-shadow: none;

}

.board {

    display: inline-block;

    background: rgba(255, 255, 255, 0.05);

    padding: 2rem;

    border-radius: 25px;

    box-shadow:

        0 8px 32px 0 rgba(31, 38, 135, 0.37),

        inset 0 0 30px rgba(255, 255, 255, 0.05);

    backdrop-filter: blur(10px);

    border: 1px solid rgba(255, 255, 255, 0.1);

    transform: perspective(1000px) rotateX(5deg);

    transition: all 0.5s ease;

}

.board:hover {

    transform: perspective(1000px) rotateX(0deg) scale(1.02);

}

.board-row {

    display: flex;

    gap: 8px;

    margin-bottom: 8px;

}

.board-row:last-child {

    margin-bottom: 0;

}

.square {

    width: 100px;

    height: 100px;

    background: linear-gradient(

        145deg,

        rgba(255, 255, 255, 0.05),

        rgba(0, 0, 0, 0.1)

    );

    border: 2px solid rgba(255, 255, 255, 0.1);

    border-radius: 20px;

    font-size: 3rem;

    font-weight: bold;

    color: #fff;

    cursor: pointer;

    transition: all 0.3s ease;

    margin: 6px;

    font-family: 'Poppins', sans-serif;

    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);

    box-shadow:

        0 4px 15px rgba(0, 0, 0, 0.2),

        inset 0 0 15px rgba(255, 255, 255, 0.05);

}

.square:hover {

    background: linear-gradient(

        145deg,

        rgba(255, 255, 255, 0.1),

        rgba(0, 0, 0, 0.15)

    );

    transform: translateY(-5px);

    box-shadow:

        0 8px 25px rgba(0, 0, 0, 0.3),

        inset 0 0 20px rgba(255, 255, 255, 0.1);

}

.square:focus {

    outline: none;

}

.square:active {

    transform: translateY(0);

}

.reset-button {

    margin-top: 3rem;

    padding: 1.2rem 3rem;

    font-size: 1.3rem;

    background: linear-gradient(45deg, #4CAF50, #45a049);

    color: white;

    border: none;

    border-radius: 30px;

    cursor: pointer;

    transition: all 0.3s ease;

    font-family: 'Poppins', sans-serif;

    font-weight: 600;

    text-transform: uppercase;

    letter-spacing: 2px;

    box-shadow:

        0 4px 15px rgba(76, 175, 80, 0.3),

        0 8px 25px rgba(0, 0, 0, 0.2);

}

.reset-button:hover {

    transform: translateY(-3px) scale(1.05);

    box-shadow:

        0 6px 20px rgba(76, 175, 80, 0.4),

        0 10px 30px rgba(0, 0, 0, 0.3);

}

.reset-button:active {

    transform: translateY(0) scale(1);

}

@media (max-width: 600px) {

    .title {

        font-size: 3rem;

    }

    .square {

        width: 80px;

        height: 80px;

        font-size: 2.5rem;

    }

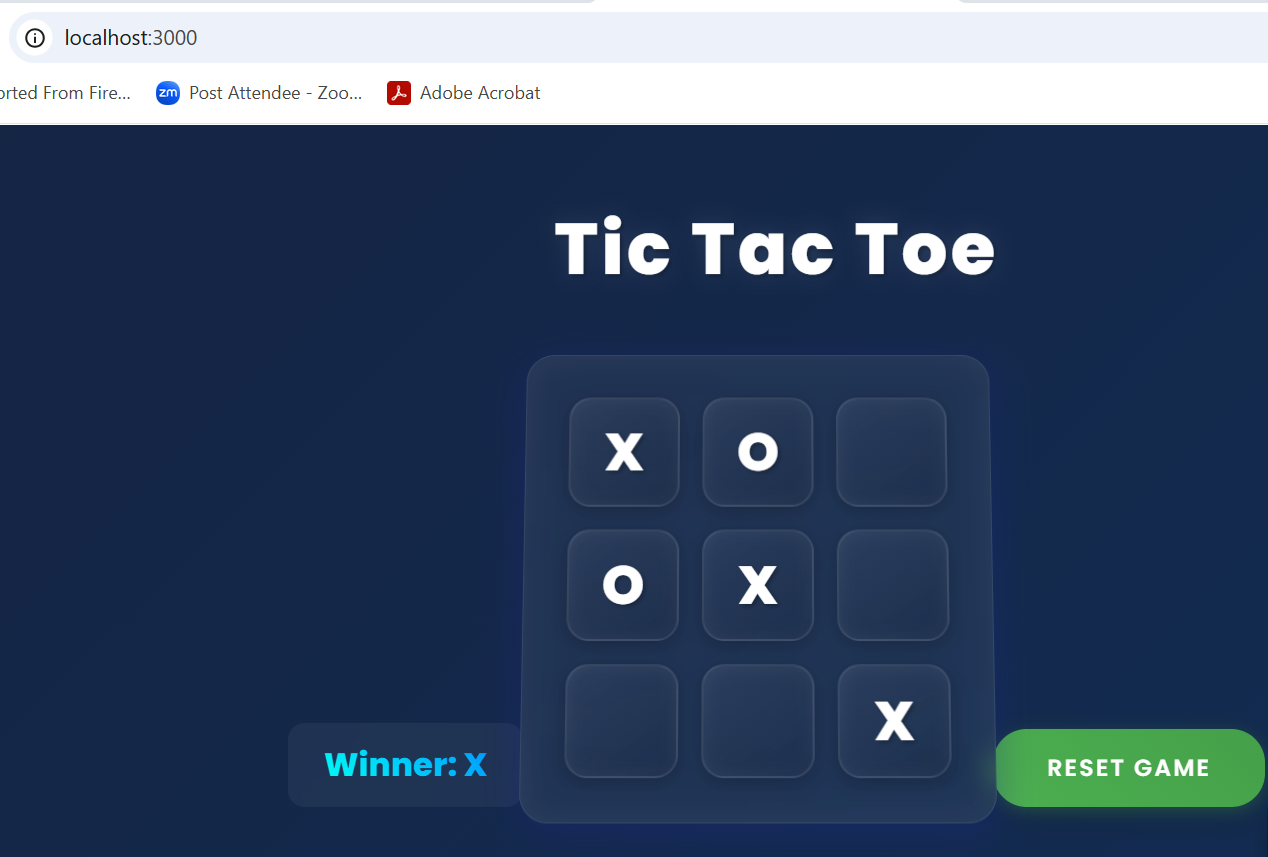
    .status {

        font-size: 1.4rem;

    }

}

**Output as**



The overall output of the game is: