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

Tools required:
1. GitHub Copilot
2. Node JS
3. Visual Studio Code

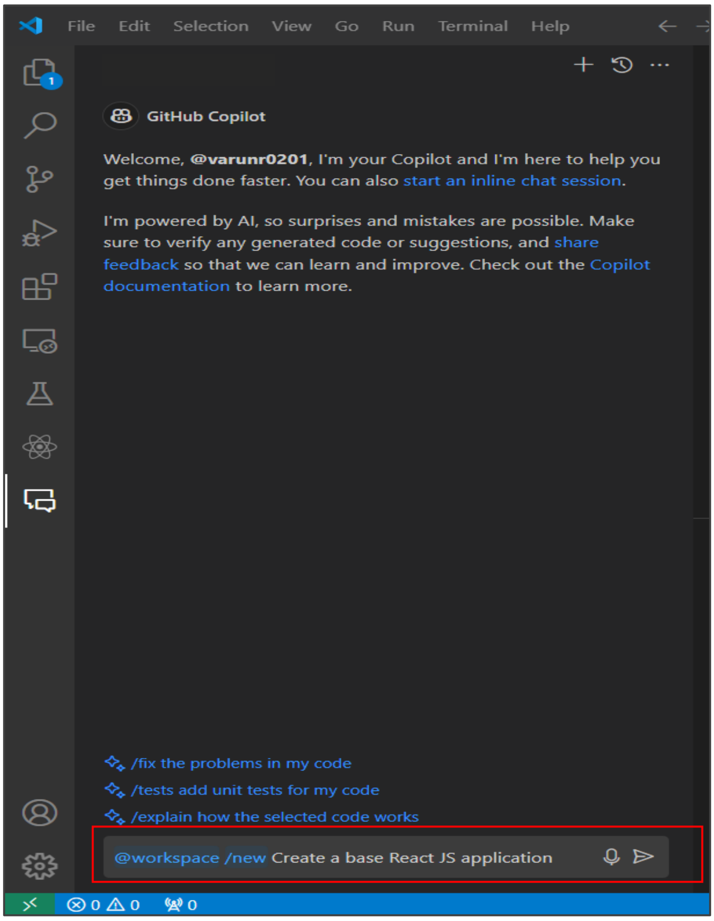
Guided Practice Solution

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

1.1 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

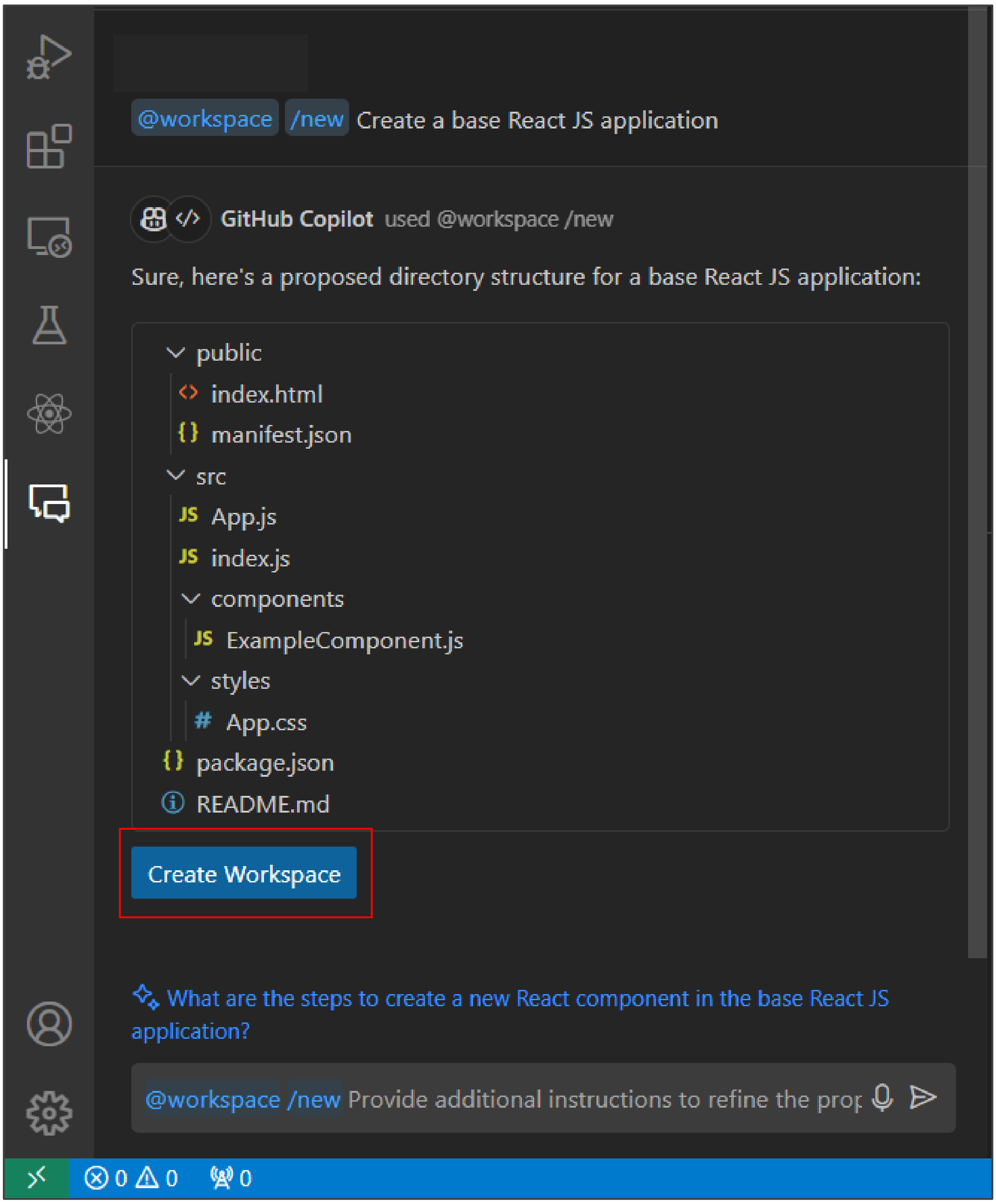

File
Edit
Selection
View
Go
Run
Terminal
Help
GitHub Copilot
/fix the problems in my code
/tests add unit tests for my code
/explain how the selected code works
0
0
0

Welcome, **@varunr0201**, I'm your Copilot and I'm here to help you get things done faster. You can also start an inline chat session.

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback so that we can learn and improve. Check out the Copilot documentation to learn more.

@workspace /new Create a base React JS application

1.2 Create the workspace by clicking on **Create Workspace**



@workspace /new Create a base React JS application

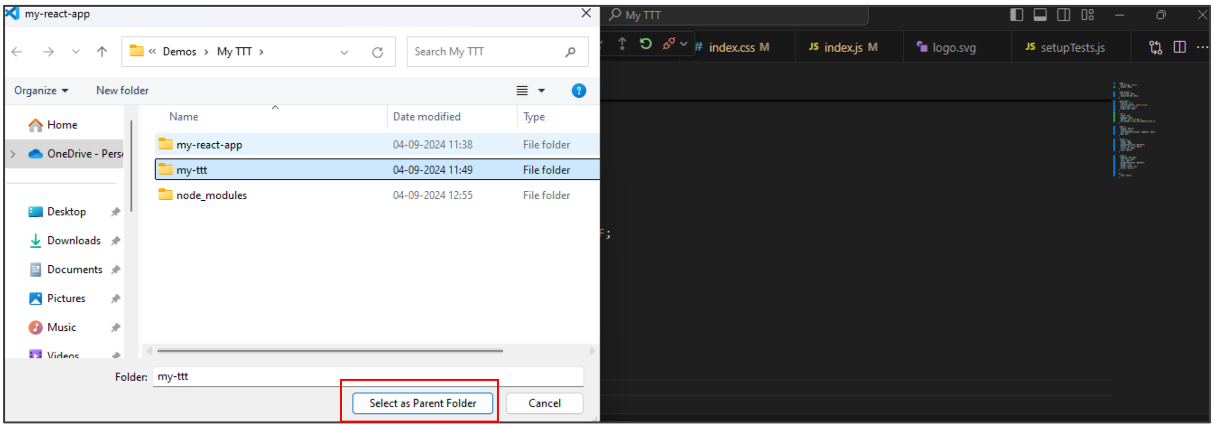
GitHub Copilot used @workspace /new

Sure, here's a proposed directory structure for a base React JS application:

```
public
  index.html
  manifest.json
src
  App.js
  index.js
  components
    ExampleComponent.js
  styles
    App.css
package.json
README.md
```

Create Workspace
What are the steps to create a new React component in the base React JS application?
@workspace /new Provide additional instructions to refine the prop
0 0
0

1.3 Choose the folder for your parent folder and then click on **Select as Parent Folder**

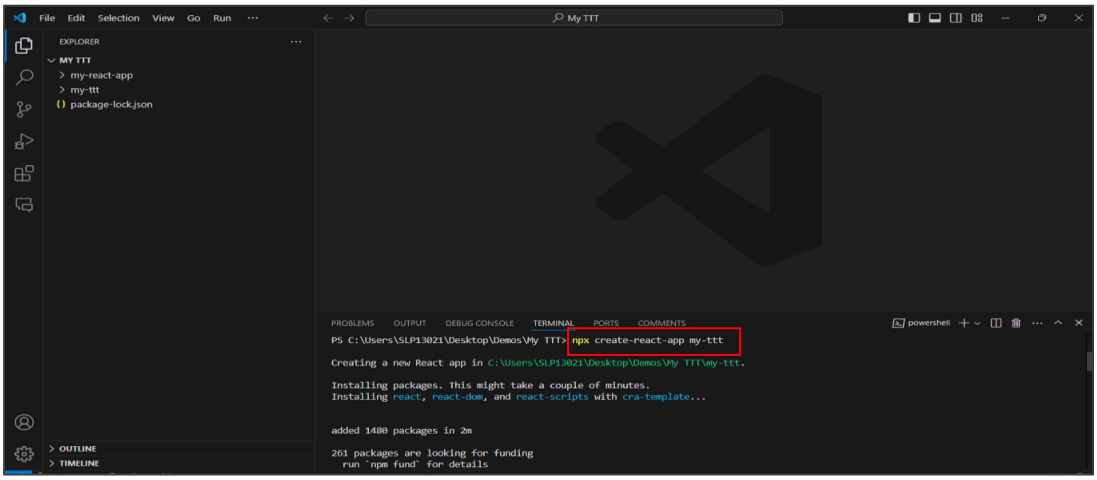


Name	Date modified	Type
my-react-app	04-09-2024 11:38	File folder
my-ttt	04-09-2024 11:49	File folder
node_modules	04-09-2024 12:55	File folder

Folder: my-ttt

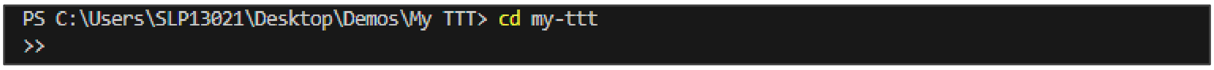
Step 2: Ensure the project environment is configured properly

2.1 Run the following command to initiate and create a React application:
npx create-react-app my-ttt



```
PS C:\Users\SLP13021\Desktop\Demos\My TTT> npx create-react-app my-ttt

Creating a new React app in C:\Users\SLP13021\Desktop\Demos\My TTT\my-ttt.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


added 1480 packages in 2m

261 packages are looking for funding
  run `npm fund` for details
```

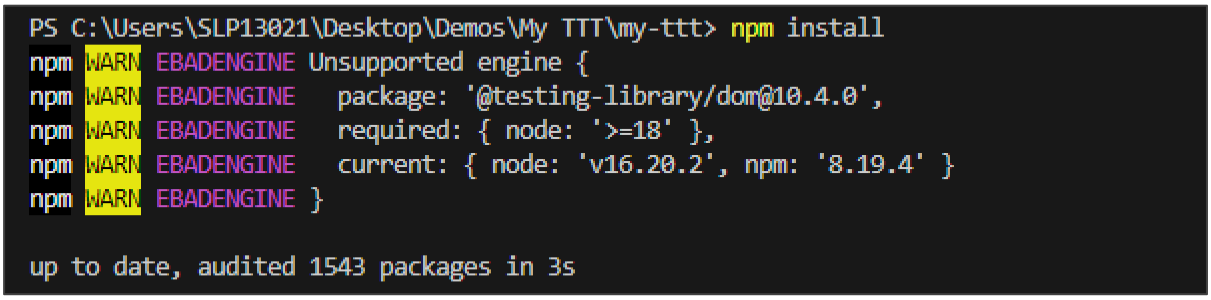
2.2 Run the following command to navigate inside the parent folder:
cd parent_folder_name



```
PS C:\Users\SLP13021\Desktop\Demos\My TTT> cd my-ttt
>>
```

2.3 Run the following command to install all the dependencies listed in the project's package.json file:

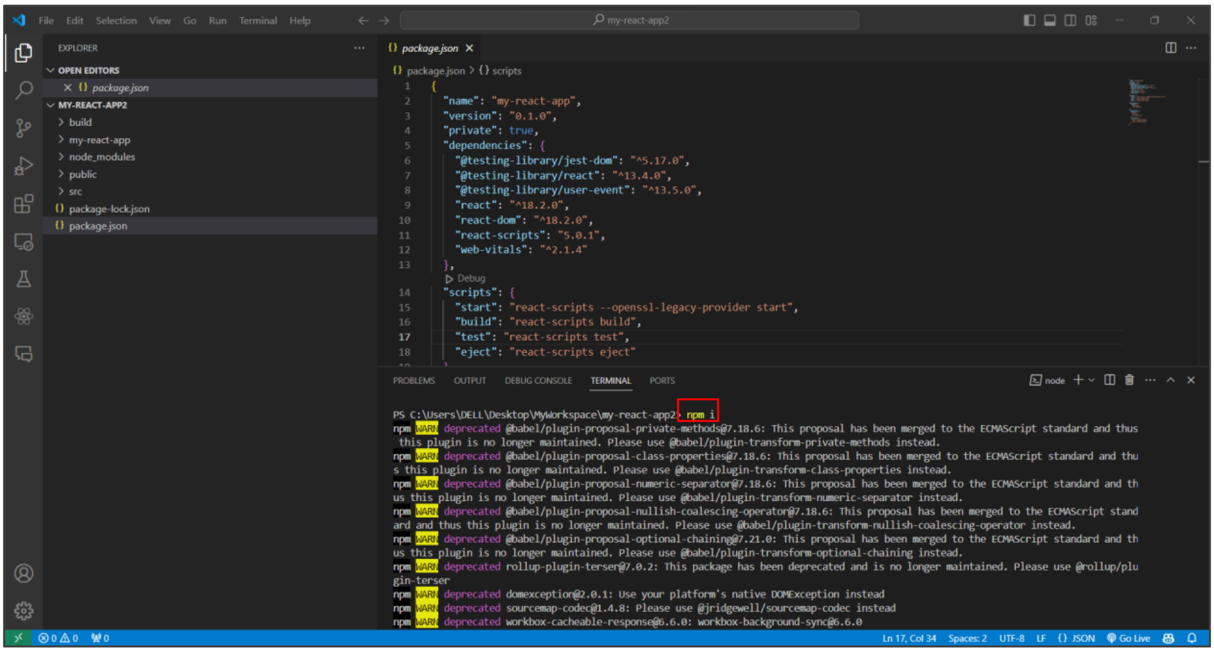
npm install



```
PS C:\Users\SLP13021\Desktop\Demos\My TTT\my-ttt> npm install
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: '@testing-library/dom@10.4.0',
npm WARN EBADENGINE   required: { node: '>=18' },
npm WARN EBADENGINE   current: { node: 'v16.20.2', npm: '8.19.4' }
npm WARN EBADENGINE }

up to date, audited 1543 packages in 3s
```

2.4 Run the following command to install packages:

npm i

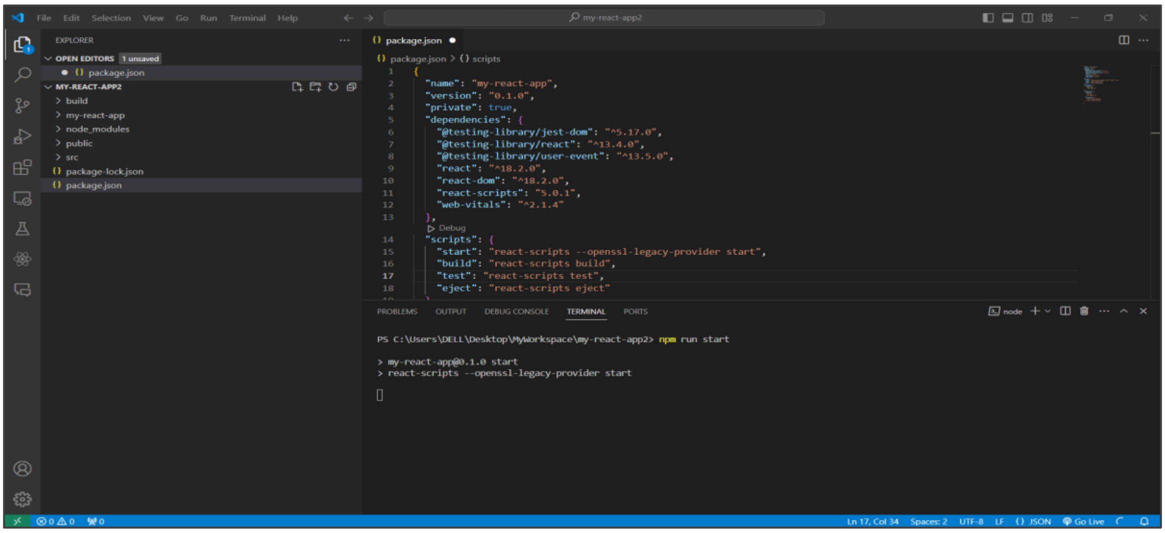


```json
{
  "name": "my-react-app",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts --openssl-legacy-provider start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
```

```
PS C:\Users\DELL\Desktop\MyWorkspace\my-react-app2> npm i
npm WARN deprecated @babel/plugin-proposal-private-methods@7.18.6: This proposal has been merged to the ECMAScript standard and thus
 this plugin is no longer maintained. Please use @babel/plugin-transform-private-methods instead.
npm WARN deprecated @babel/plugin-proposal-class-properties@7.18.6: This proposal has been merged to the ECMAScript standard and thu
s this plugin is no longer maintained. Please use @babel/plugin-transform-class-properties instead.
npm WARN deprecated @babel/plugin-proposal-numeric-separator@7.18.6: This proposal has been merged to the ECMAScript standard and th
us this plugin is no longer maintained. Please use @babel/plugin-transform-numeric-separator instead.
npm WARN deprecated @babel/plugin-proposal-nullish-coalescing-operator@7.18.6: This proposal has been merged to the ECMAScript stand
ard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-nullish-coalescing-operator instead.
npm WARN deprecated @babel/plugin-proposal-optional-chaining@7.21.0: This proposal has been merged to the ECMAScript standard and th
us this plugin is no longer maintained. Please use @babel/plugin-transform-optional-chaining instead.
npm WARN deprecated rollup-plugin-terser@7.0.2: This package has been deprecated and is no longer maintained. Please use @rollup/plu
gin-terser
npm WARN deprecated domexception@2.0.1: Use your platform's native DOMException instead
npm WARN deprecated sourcemap-codec@1.4.8: Please use @jridgewell/sourcemap-codec instead
npm WARN deprecated workbox-cacheable-response@6.6.0: workbox-background-sync@6.6.0
```

2.5 Run the project using the following command:

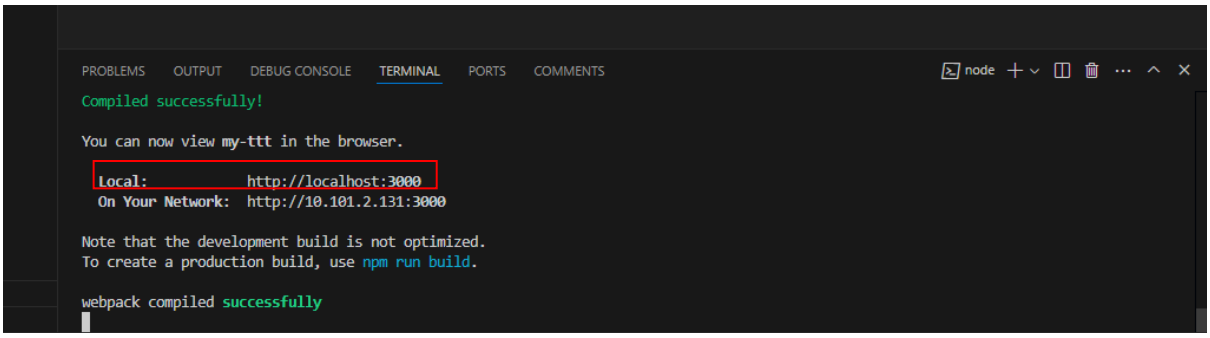
npm run start



```json
{
  "name": "my-react-app",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts --openssl-legacy-provider start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
```

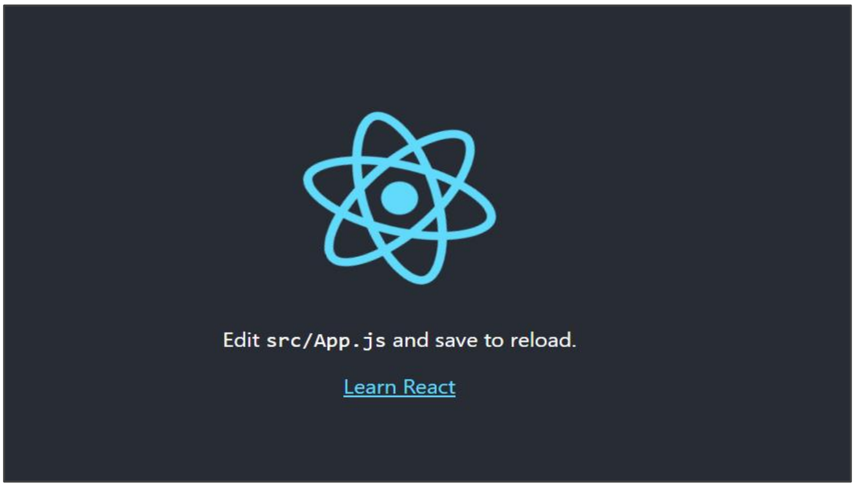
```
PS C:\Users\DELL\Desktop\MyWorkspace\my-react-app2> npm run start

> my-react-app@0.1.0 start
> react-scripts --openssl-legacy-provider start
```

2.6 Once the project runs successfully and displays the success message, click on the Local link, as shown in the screenshot below:



```
Compiled successfully!

You can now view my-ttt in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://10.101.2.131:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

The output is as follows:



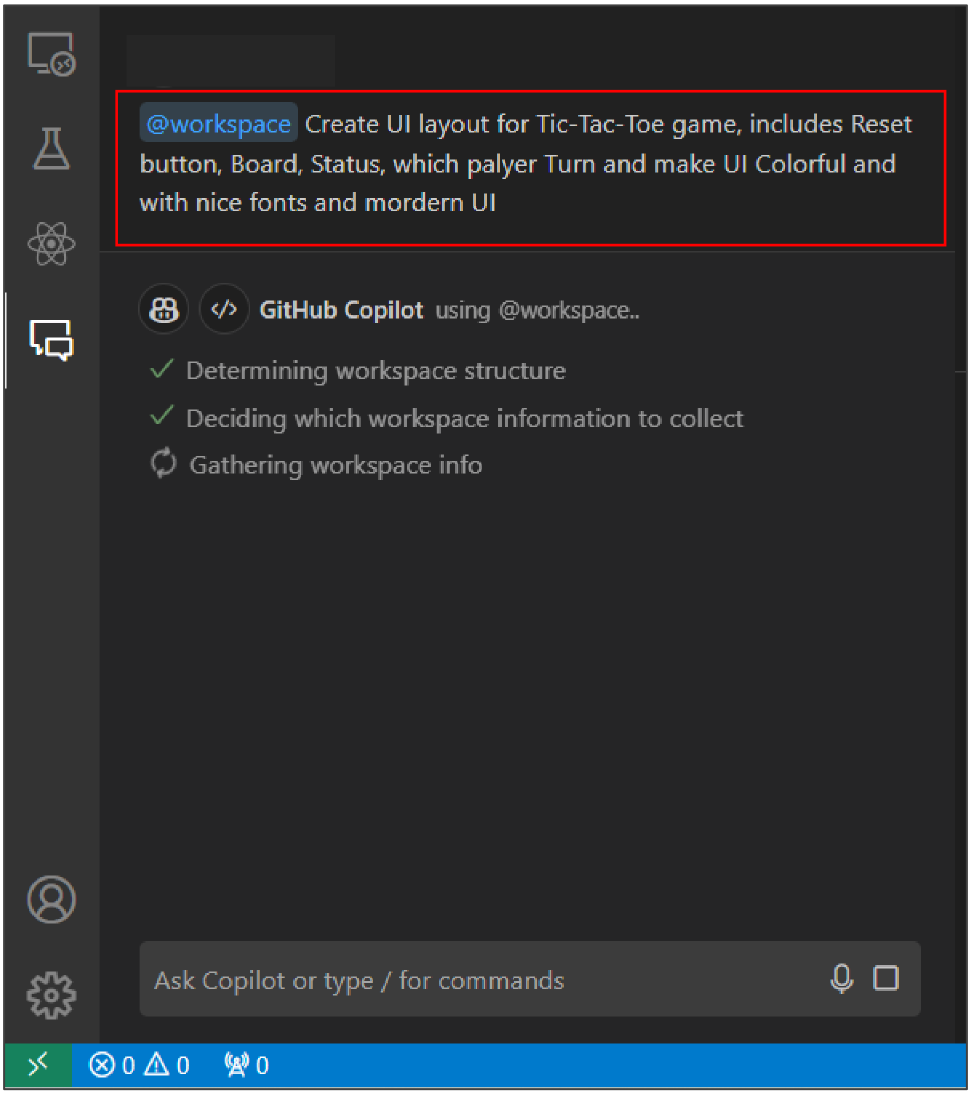
Edit `src/App.js` and save to reload.

Learn React

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

Step 1: Use prompts to create the basic user interface

1.1 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

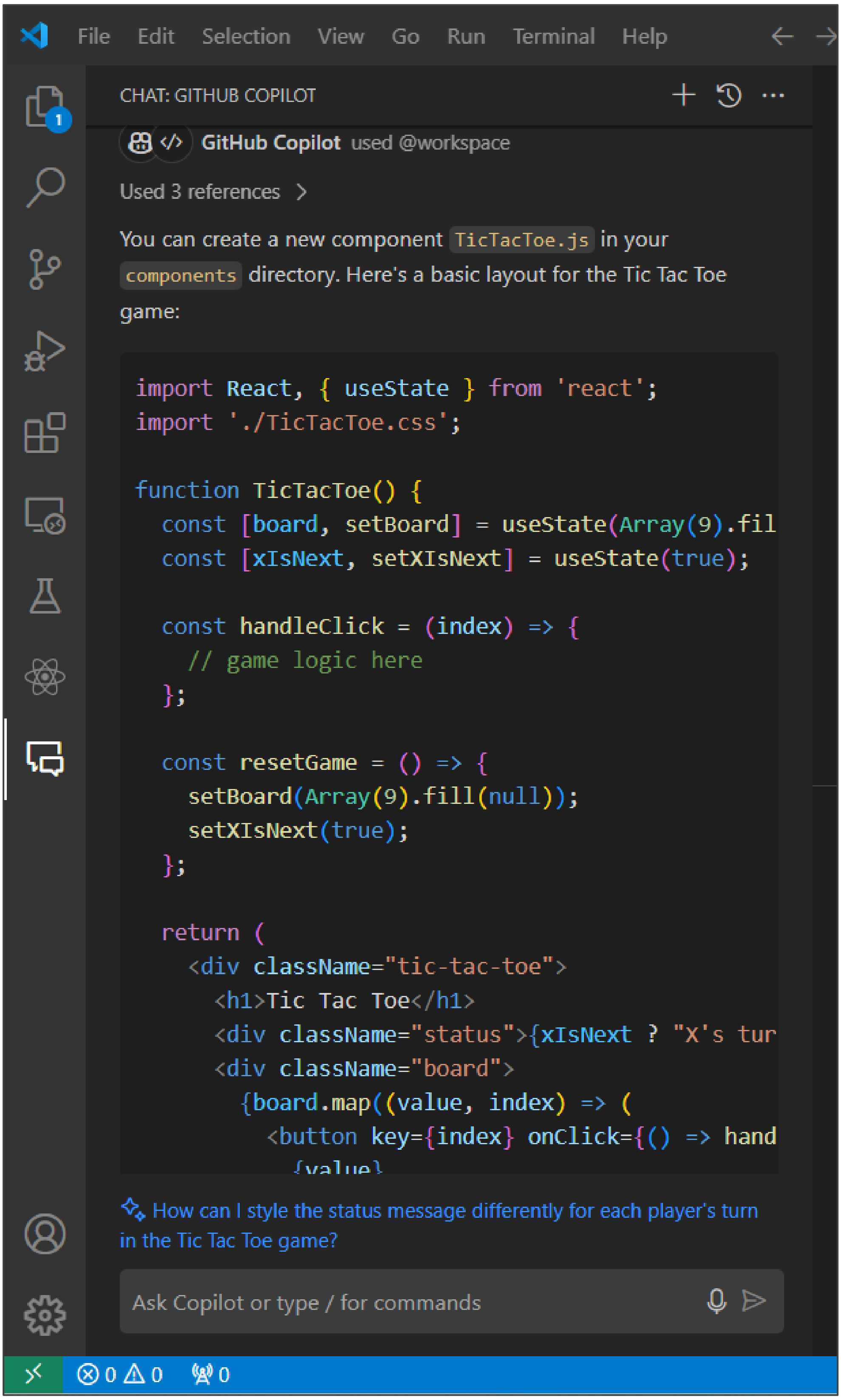


@workspace Create UI layout for Tic-Tac-Toe game, includes Reset button, Board, Status, which palyer Turn and make UI Colorful and with nice fonts and mordern UI

GitHub Copilot using @workspace..

✓ Determining workspace structure
✓ Deciding which workspace information to collect
Gathering workspace info

1.2 Place the generated script inside the src folder



GitHub Copilot used @workspace

Used 3 references >

You can create a new component `TicTacToe.js` in your `components` directory. Here's a basic layout for the Tic Tac Toe game:

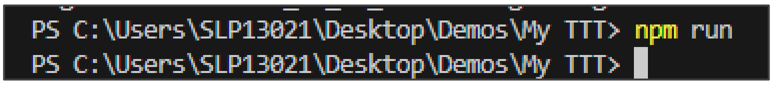
```
import React, { useState } from 'react';
import './TicTacToe.css';

function TicTacToe() {
  const [board, setBoard] = useState(Array(9).fil
  const [xIsNext, setXIsNext] = useState(true);

  const handleClick = (index) => {
    // game logic here
  };

  const resetGame = () => {
    setBoard(Array(9).fill(null));
    setXIsNext(true);
  };

  return (
    <div className="tic-tac-toe">
      <h1>Tic Tac Toe</h1>
      <div className="status">{xIsNext ? "X's tur
      <div className="board">
        {board.map((value, index) => (
          <button key={index} onClick={() => hand
            {value}
```

How can I style the status message differently for each player's turn in the Tic Tac Toe game?

Ask Copilot or type / for commands

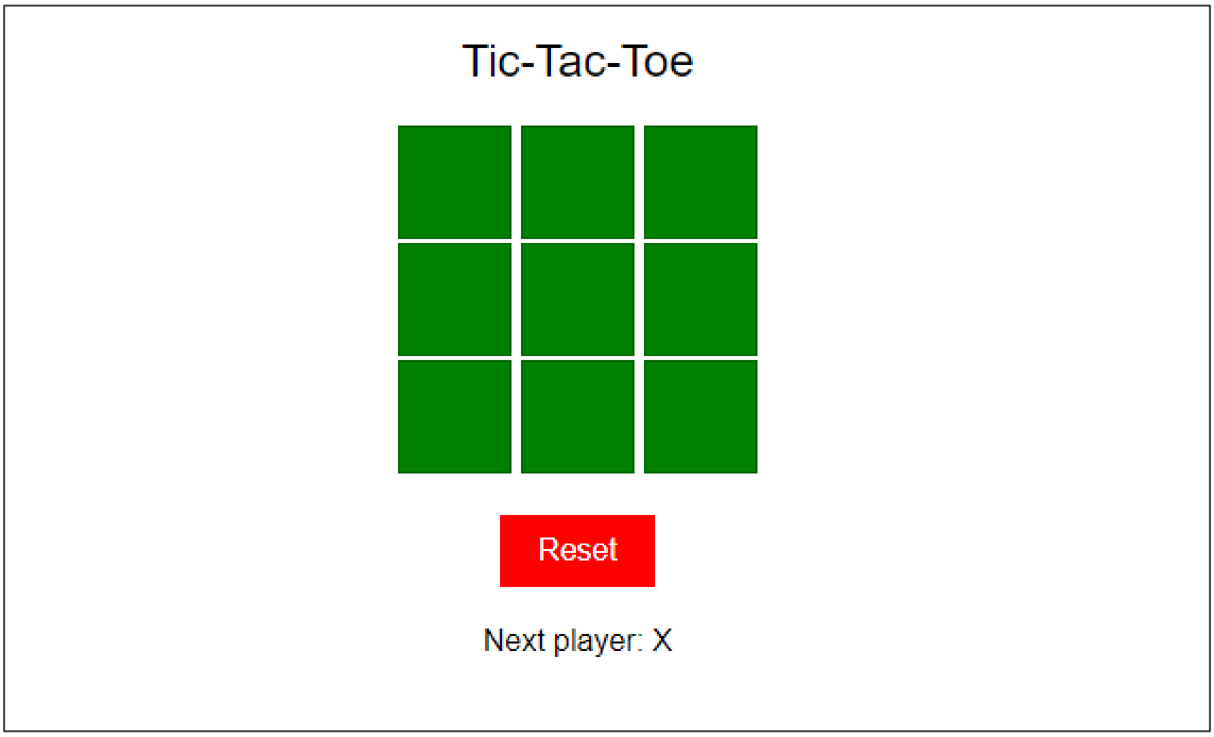
1.3 Run the following command to execute scripts:

npm run



```
PS C:\Users\SLP13021\Desktop\Demos\My TTT> npm run
PS C:\Users\SLP13021\Desktop\Demos\My TTT> 
```

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

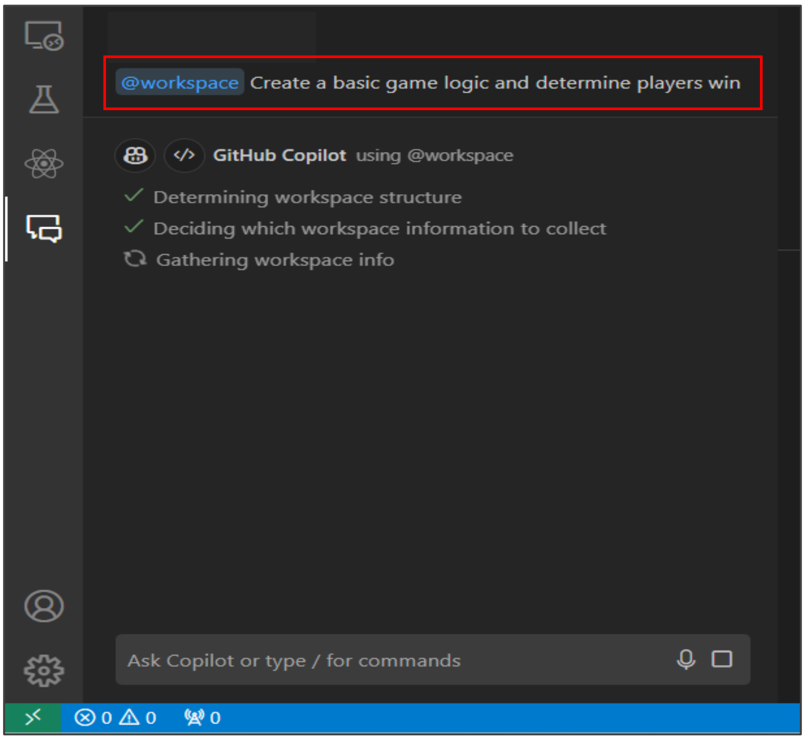


Next player: X

Step 2: Use prompts to develop the basic game logic and determine when a player wins

2.1 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



@workspace Create a basic game logic and determine players win

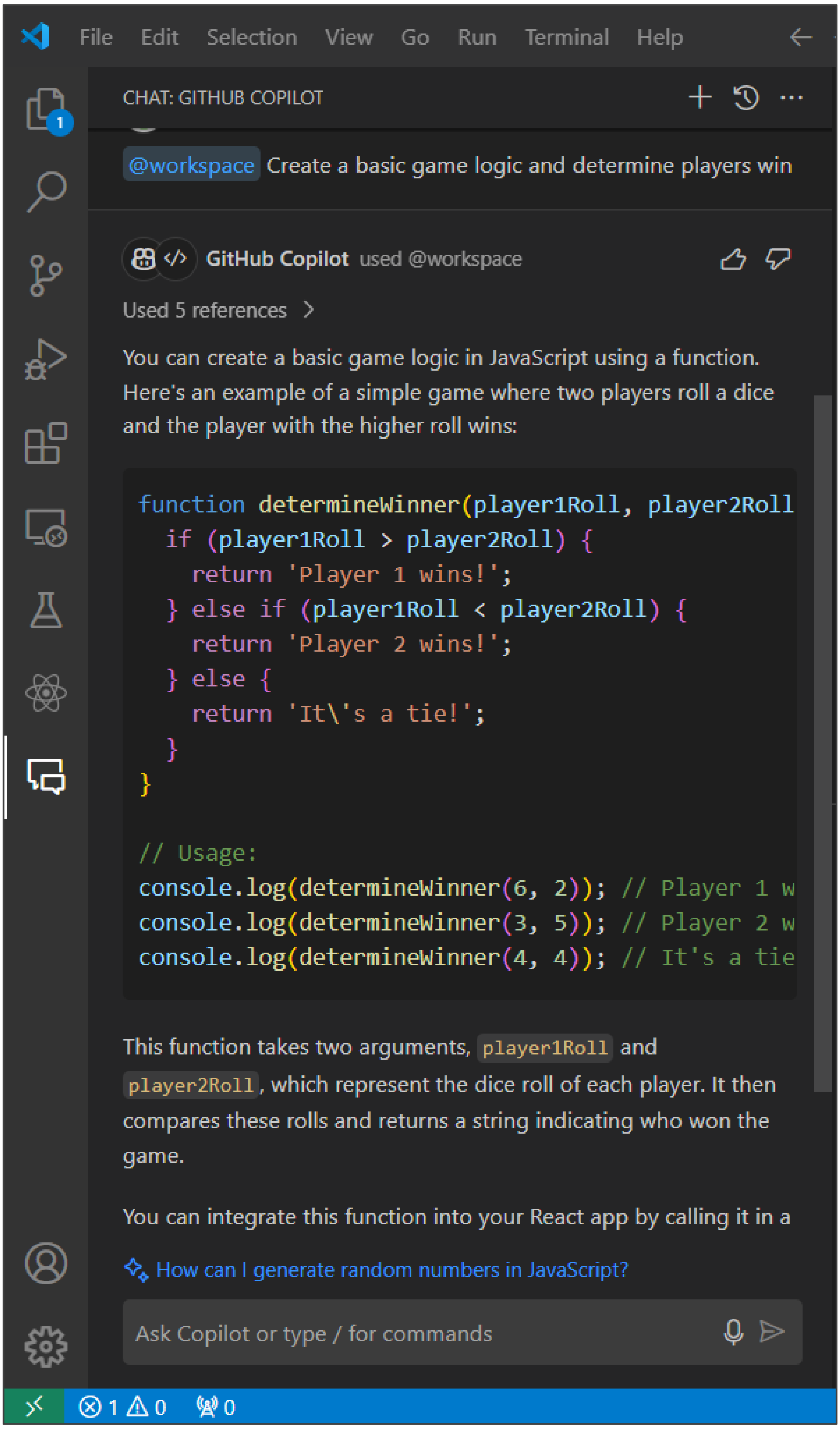
GitHub Copilot using @workspace

✓ Determining workspace structure

✓ Deciding which workspace information to collect

Gathering workspace info

The response from GitHub Copilot is as follows:



@workspace Create a basic game logic and determine players win

GitHub Copilot used @workspace

Used 5 references

You can create a basic game logic in JavaScript using a function. Here's an example of a simple game where two players roll a dice and the player with the higher roll wins:

```
function determineWinner(player1Roll, player2Roll
  if (player1Roll > player2Roll) {
    return 'Player 1 wins!';
  } else if (player1Roll < player2Roll) {
    return 'Player 2 wins!';
  } else {
    return 'It\'s a tie!';
  }
}

// Usage:
console.log(determineWinner(6, 2)); // Player 1 w
console.log(determineWinner(3, 5)); // Player 2 w
console.log(determineWinner(4, 4)); // It's a tie
```

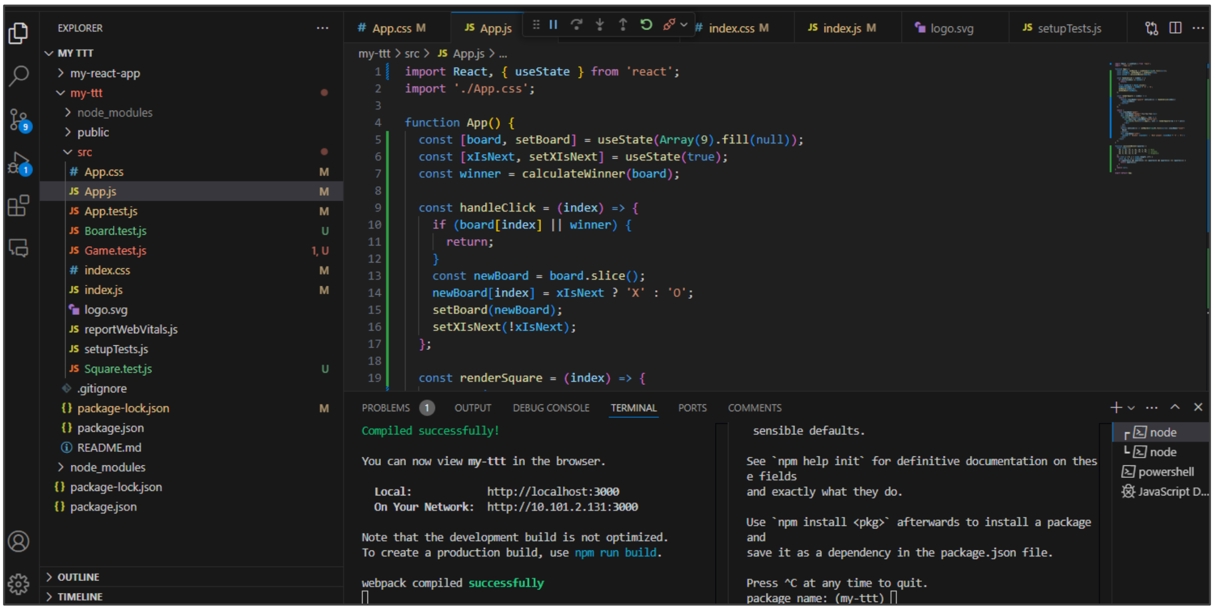
This function takes two arguments, `player1Roll` and `player2Roll`, which represent the dice roll of each player. It then compares these rolls and returns a string indicating who won the game.

You can integrate this function into your React app by calling it in a

How can I generate random numbers in JavaScript?

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

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

function App() {
  const [board, setBoard] = useState(Array(9).fill(null));
  const [xIsNext, setXIsNext] = useState(true);
  const winner = calculateWinner(board);

  const handleClick = (index) => {
    if (board[index] || winner) {
      return;
    }
    const newBoard = board.slice();
    newBoard[index] = xIsNext ? 'X' : 'O';
    setBoard(newBoard);
    setXIsNext(!xIsNext);
  };

  const renderSquare = (index) => {
    return (
      <button className="square" onClick={() => handleClick(index)}>
        {board[index]}
      </button>
    );
  };

  return (
    <div className="game">
      <div className="header">Tic-Tac-Toe</div>
      <div className="board">
        {Array(3).fill(null).map((_, row) => (
          <div key={row} className="board-row">
            {Array(3).fill(null).map((_, col) => renderSquare(row * 3 + col))}
          </div>
        ))}
      </div>
      <button onClick={() => setBoard(Array(9).fill(null))} className="reset">
        Reset
      </button>
      <div className="info">
        {winner ? `Winner: ${winner}` : `Next player: ${xIsNext ? 'X' : 'O'}`}
      </div>
    </div>
  );
}

function calculateWinner(squares) {
  const lines = [
    [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 i = 0; i < lines.length; i++) {
    const [a, b, c] = lines[i];
    if (squares[a] && squares[a] === squares[b] && squares[a] === squares[c]) {
      return squares[a];
    }
  }
  return null;
}

export default App;
```



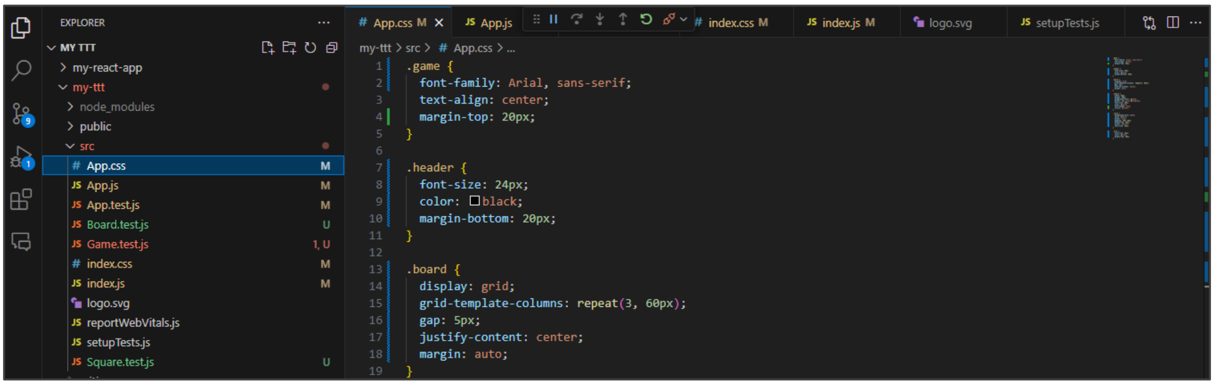
```
import React, { useState } from 'react';
import './App.css';

function App() {
  const [board, setBoard] = useState(Array(9).fill(null));
  const [xIsNext, setXIsNext] = useState(true);
  const winner = calculateWinner(board);

  const handleClick = (index) => {
    if (board[index] || winner) {
      return;
    }
    const newBoard = board.slice();
    newBoard[index] = xIsNext ? 'X' : 'O';
    setBoard(newBoard);
    setXIsNext(!xIsNext);
  };

  const renderSquare = (index) => {
```

```
Compiled successfully!

You can now view my-ttt in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://10.101.2.131:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

```
 sensible defaults.

See `npm help init` for definitive documentation on thes
e fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package
and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (my-ttt)
```

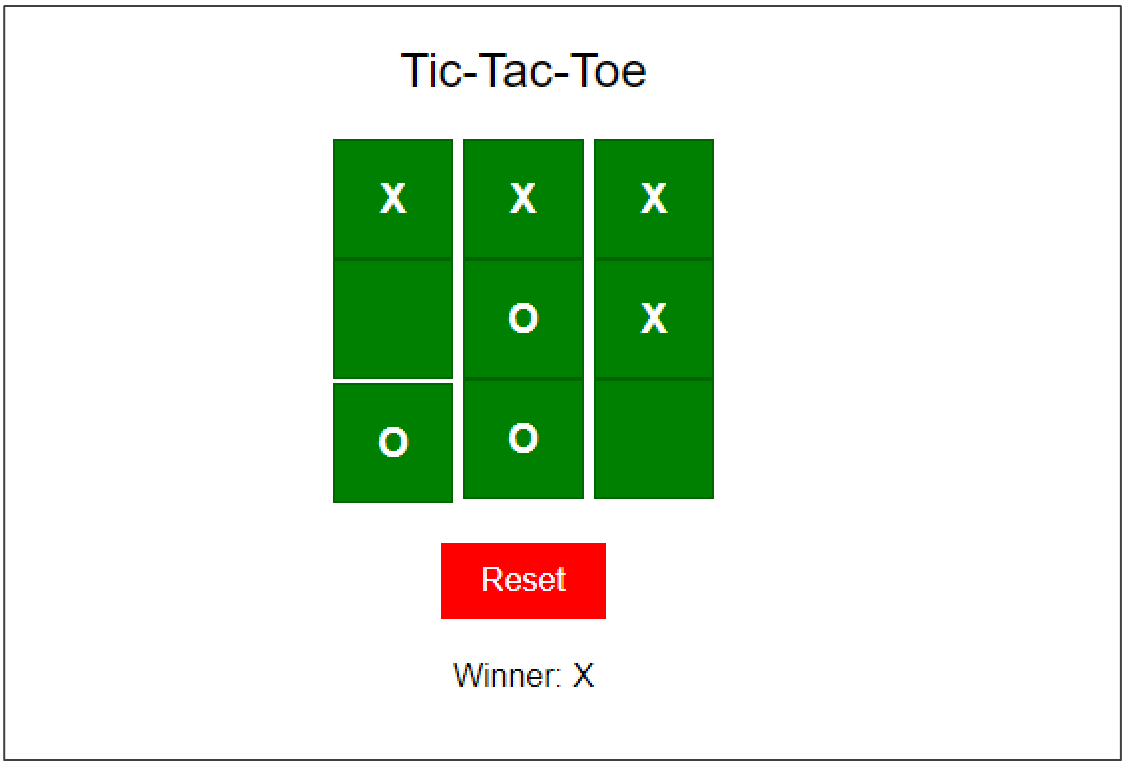
2.3 Add the following code inside App.css to enhance the visual appearance:

```
.game {
  font-family: Arial, sans-serif;
  text-align: center;
  margin-top: 20px;
}

.header {
  font-size: 24px;
  color: black;
  margin-bottom: 20px;
}

.board {
  display: grid;
  grid-template-columns: repeat(3, 60px);
  gap: 5px;
  justify-content: center;
  margin: auto;
}

.square {
  width: 60px;
  height: 60px;
  background-color: green;
  border: 1px solid darkgreen;
  font-size: 20px;
  color: white;
  font-weight: bold;
  cursor: pointer;
}

.reset {
  background-color: red;
  color: white;
  border: none;
  padding: 10px 20px;
  margin-top: 20px;
  cursor: pointer;
  font-size: 16px;
}

.info {
  font-size: 16px;
  color: black;
  margin-top: 20px;
}
```



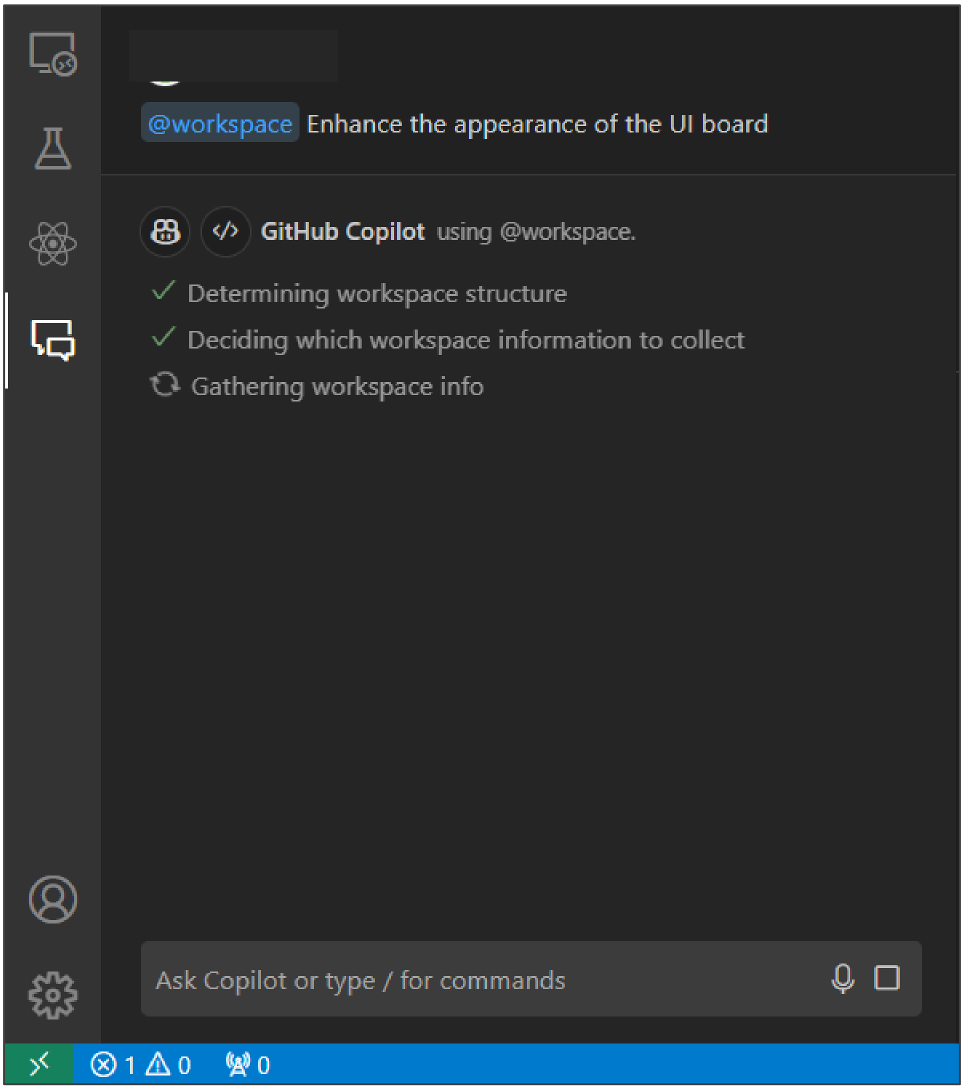
```css
.game {
  font-family: Arial, sans-serif;
  text-align: center;
  margin-top: 20px;
}

.header {
  font-size: 24px;
  color: black;
  margin-bottom: 20px;
}

.board {
  display: grid;
  grid-template-columns: repeat(3, 60px);
  gap: 5px;
  justify-content: center;
  margin: auto;
}
```

The output after implementing the game logic is as follows:



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

3.1 Use the following prompt to enhance the appearance of the UI board:
@workspace Enhance the appearance of the UI board



@workspace Enhance the appearance of the UI board

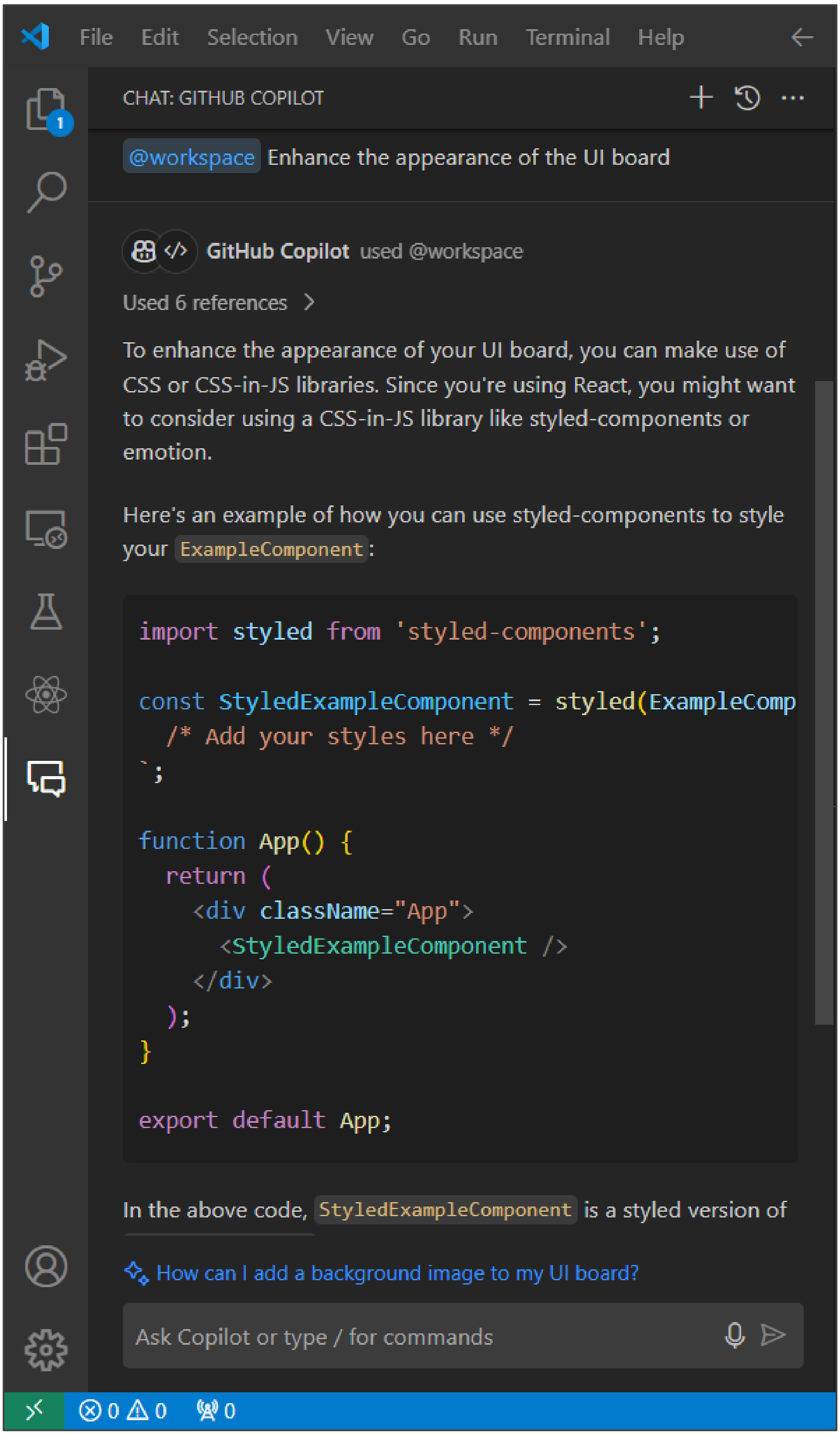
GitHub Copilot using @workspace.

✓ Determining workspace structure

✓ Deciding which workspace information to collect

Gathering workspace info

The generated script for enhancing the board is as follows:



@workspace Enhance the appearance of the UI board

GitHub Copilot used @workspace

Used 6 references

To enhance the appearance of your UI board, you can make use of CSS or CSS-in-JS libraries. Since you're using React, you might want to consider using a CSS-in-JS library like styled-components or emotion.

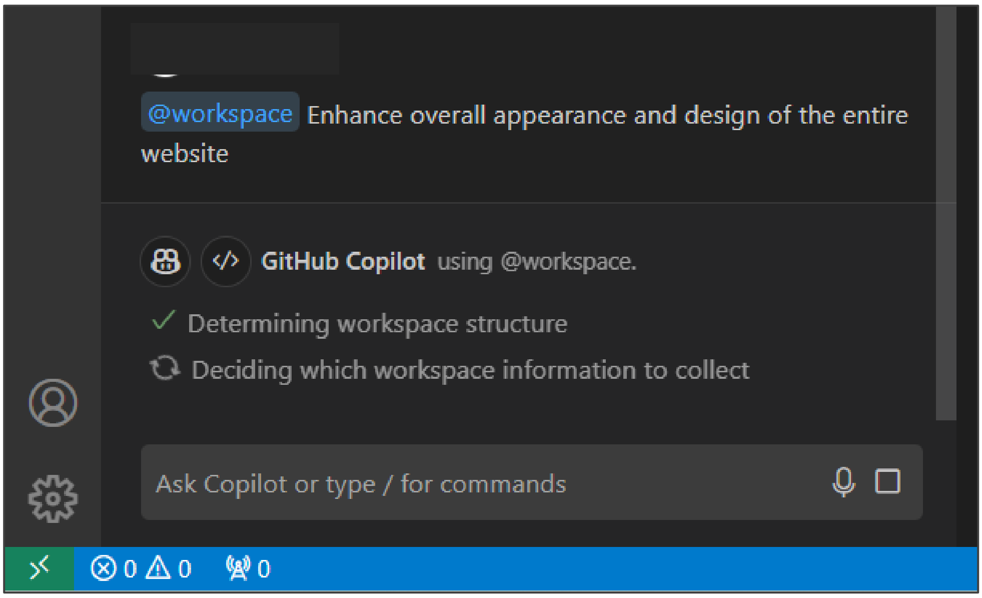
Here's an example of how you can use styled-components to style your `ExampleComponent`:

```
import styled from 'styled-components';

const StyledExampleComponent = styled(ExampleComp
  /* Add your styles here */
`;

function App() {
  return (
    <div className="App">
      <StyledExampleComponent />
    </div>
  );
}

export default App;
```

In the above code, `StyledExampleComponent` is a styled version of

How can I add a background image to my UI board?

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



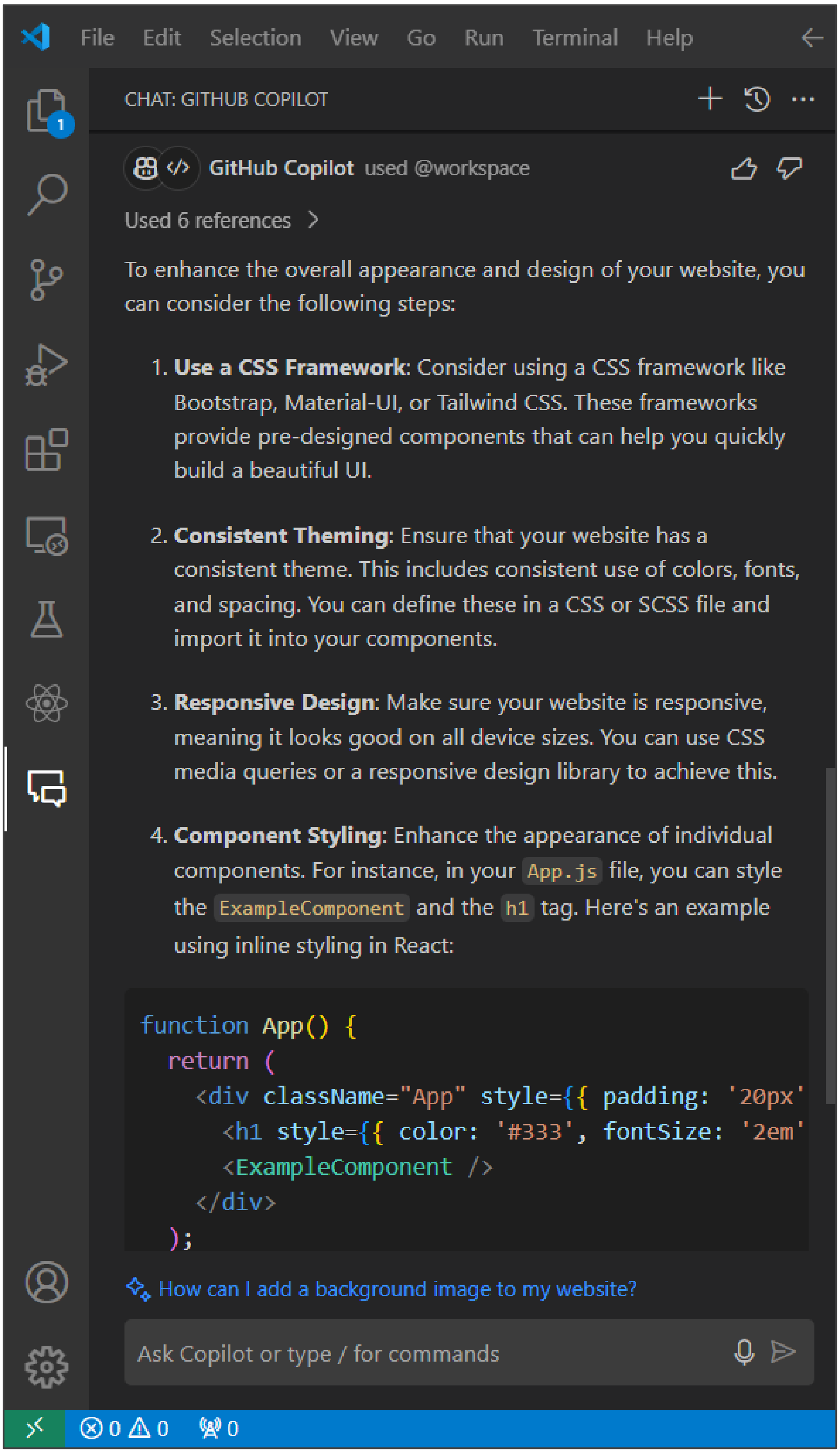
@workspace Enhance overall appearance and design of the entire website

GitHub Copilot using @workspace.

✓ Determining workspace structure

Deciding which workspace information to collect

The generated script to enhance the overall appearance and design is as follows:



To enhance the overall appearance and design of your website, you can consider the following steps:

1. **Use a CSS Framework**: Consider using a CSS framework like Bootstrap, Material-UI, or Tailwind CSS. These frameworks provide pre-designed components that can help you quickly build a beautiful UI.

2. **Consistent Theming**: Ensure that your website has a consistent theme. This includes consistent use of colors, fonts, and spacing. You can define these in a CSS or SCSS file and import it into your components.

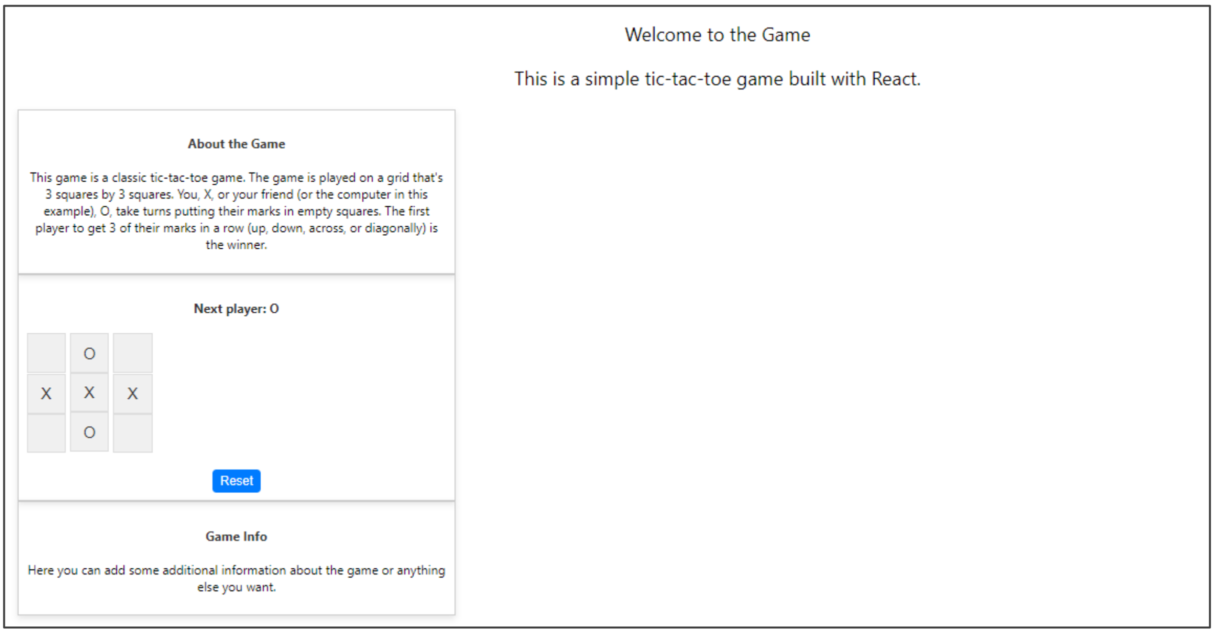
3. **Responsive Design**: Make sure your website is responsive, meaning it looks good on all device sizes. You can use CSS media queries or a responsive design library to achieve this.

4. **Component Styling**: Enhance the appearance of individual components. For instance, in your `App.js` file, you can style the `ExampleComponent` and the `h1` tag. Here's an example using inline styling in React:

```
function App() {
  return (
    <div className="App" style={{ padding: '20px'
      <h1 style={{ color: '#333', fontSize: '2em'
      <ExampleComponent />
    </div>
  );
```

How can I add a background image to my website?
Ask Copilot or type / for commands
0 0
0

The overall output of the game is:



Welcome to the Game

This is a simple tic-tac-toe game built with React.

About the Game

This game is a classic tic-tac-toe game. The game is played on a grid that's 3 squares by 3 squares. You, X, or your friend (or the computer in this example), O, take turns putting their marks in empty squares. The first player to get 3 of their marks in a row (up, down, across, or diagonally) is the winner.

Next player: O

	O	
X	X	X
	O	

Game Info

Here you can add some additional information about the game or anything else you want.