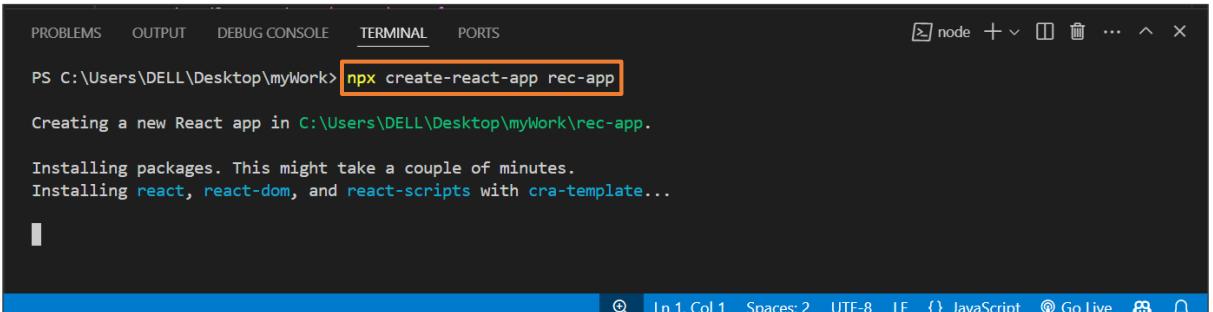


Answer Key

1. Set up the project in Visual Studio Code and GitHub utilizing npx and GitHub Copilot for code generation

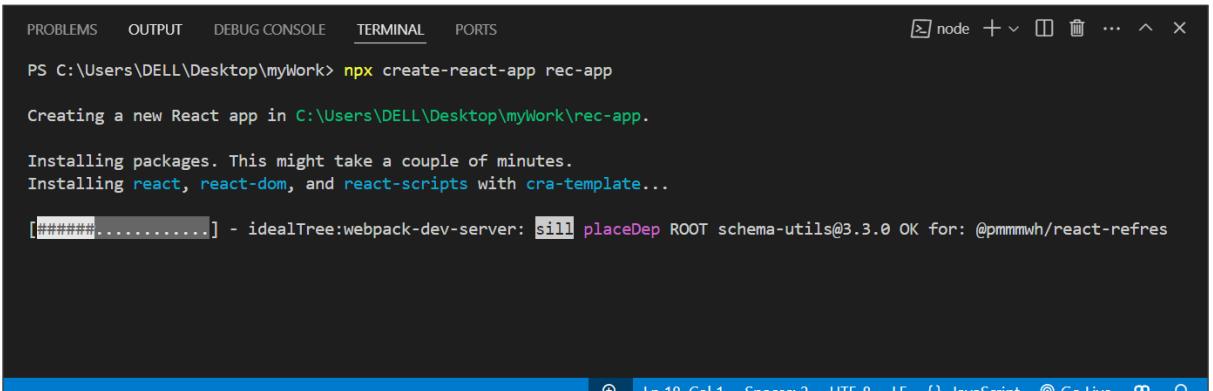
1.1 Set up a new React.js project with the default project structure and configuration using the following command:

```
npx create-react-app rec-app
```



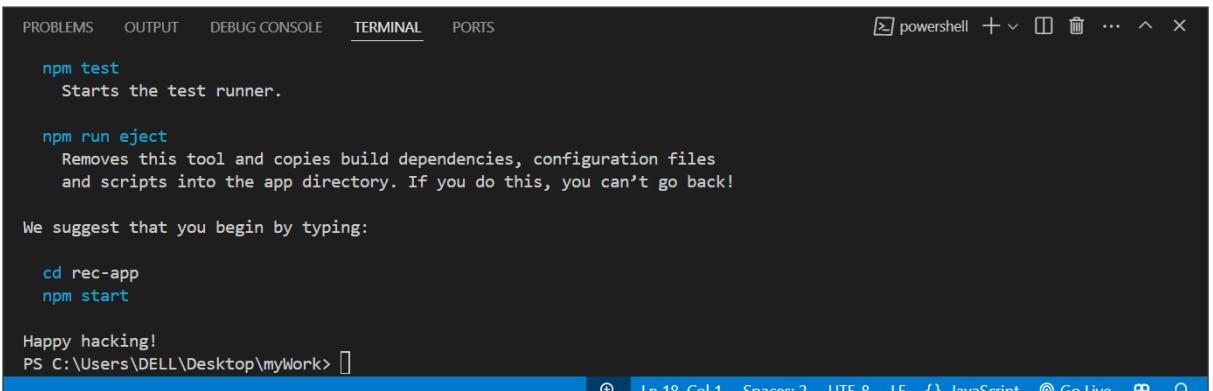
The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal window displays the command `npx create-react-app rec-app` being run. The output shows the process of creating a new React app named `rec-app`, including the installation of packages like `react`, `react-dom`, and `react-scripts`. The status bar at the bottom indicates the file is a JavaScript file.

`npx` creates the new project in the `~\Desktop\myWork\rec-app` folder as follows:



This screenshot shows a continuation of the terminal session from the previous one. It includes additional output from the `npx create-react-app rec-app` command, specifically the `silly placeDep` log message. The status bar at the bottom indicates the file is a JavaScript file.

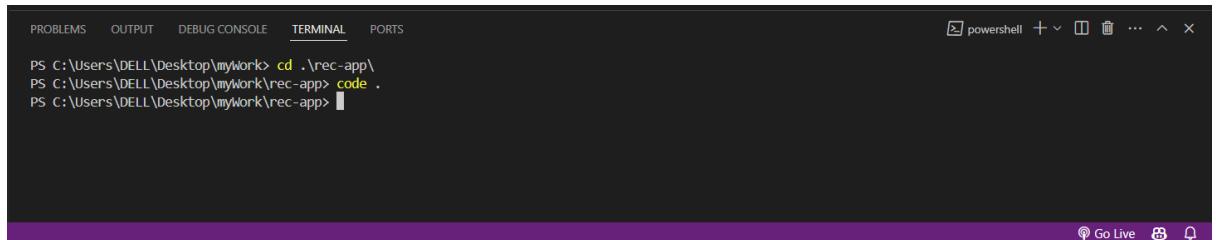
The React.js project is successfully created, showing the following after creation:



This screenshot shows the terminal displaying instructions for the newly created React app. It includes commands like `npm test` (starts the test runner), `npm run eject` (removes tooling and copies build dependencies), and suggestions for starting the app with `cd rec-app` and `npm start`. The terminal also ends with the message "Happy hacking!" and the prompt `PS C:\Users\DELL\Desktop\myWork>`. The status bar at the bottom indicates the file is a JavaScript file.

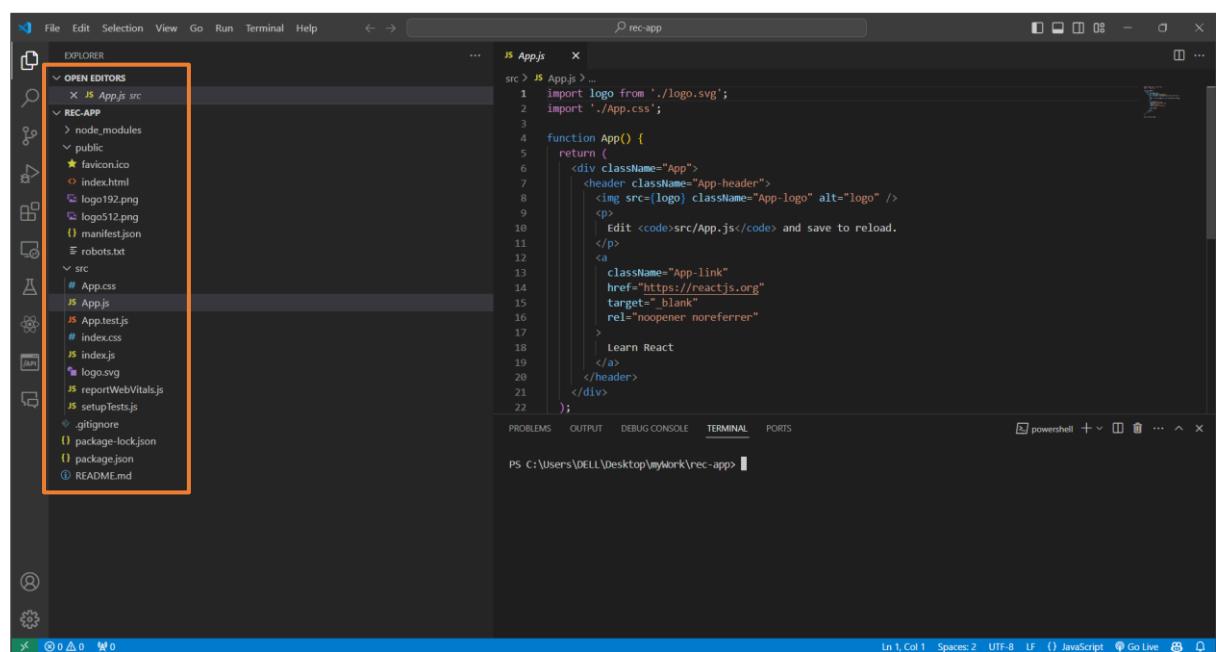
1.2 Execute the following command to navigate to the folder where it is created and to open that project in another instance of Visual Studio:

```
cd .\rec-app\  
code .
```



A screenshot of a terminal window titled "powershell". The command "cd .\rec-app\" is entered, followed by "code ." which starts the code editor. The terminal shows the path "C:\Users\DELL\Desktop\myWork\rec-app".

The basic React.js structure appears as follows:



A screenshot of the Visual Studio Code interface. The Explorer sidebar on the left shows the project structure with a red box highlighting the "src" folder. The main editor area shows the "App.js" file with some React code. The terminal at the bottom shows the command "PS C:\Users\DELL\Desktop\myWork\rec-app>".

```
src > JS App.js ...  
1 import logo from './logo.svg';  
2 import './App.css';  
3  
4 function App() {  
5   return (  
6     <div className="App">  
7       <header className="App-header">  
8         <img src={logo} className="App-logo" alt="logo" />  
9       </header>  
10      <p>Edit <code>src/App.js</code> and save to reload.  
11      </p>  
12      <a  
13        className="App-link"  
14        href="https://reactjs.org"  
15        target="_blank"  
16        rel="noopener noreferrer"  
17      >  
18        Learn React  
19      </a>  
20    </header>  
21  </div>  
22);
```

2. Generate a basic user interface for recipe data using GitHub Copilot

2.1 Enhance the project by prompting GitHub Copilot to generate React.js code for a basic UI for the recipe app

@workspace create a basic UI for recipe app

The screenshot shows the GitHub Copilot interface. On the left is a sidebar with various icons. The main area has a dark background. At the top, it says "CHAT: GITHUB COPILOT" and "and improve. Check out the [Copilot documentation](#) to learn more." Below that, a message from "aspras_microsoft" is shown: "@workspace create a basic UI for recipe app". This message is highlighted with a red box. Below the message, it says "GitHub Copilot used @workspace" and "Used 6 references >". A response follows: "Sure, here's a basic example of how you could structure a recipe app in React. This will be a simple app with a list of recipes and a form to add new recipes." Underneath, it says "In your App.js file:" followed by a code block:

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

function App() {
  const [recipes, setRecipes] = useState([]);
  const [newRecipe, setNewRecipe] = useState("");

  const addRecipe = () => {
    setRecipes([...recipes, newRecipe]);
    setNewRecipe("");
  };

  return (
    <div className="App">
      <header className="App-header">
        <h1>Recipe App</h1>
        <input
          value={newRecipe}
          onChange={e => setNewRecipe(e.target.value)}>
    
```

At the bottom, there is a question "How can I style the recipe list in the UI?" and a footer bar with "Ask Copilot or type / for commands" and a search icon.

Note: Copy the contents of the generated script and paste them into the existing `src\App.js` file

3. Generate sample data using GitHub Copilot

3.1 Prompt GitHub Copilot to create sample data for the existing recipe app @wokspace create sample data for it

The screenshot shows the GitHub Copilot interface. At the top, there is a code snippet:

```
export default App;
```

Below the code, a message says:

This will create a simple UI where you can add new recipes to a list. The new recipes are stored in the `recipes` state variable, and the input field is controlled by the `newRecipe` state variable. When you click the "Add Recipe" button, the current value of `newRecipe` is added to the `recipes` array, and `newRecipe` is reset to an empty string.

Please note that this is a very basic example and a real-world application would likely involve more complex state management, form validation, and possibly communication with a backend server to persist the recipes.

In the center, there is a GitHub Copilot logo and the text "aspras_microsoft". Below that, the prompt "@wokspace create sample data for it" is highlighted in a red box. To the left of the box is a GitHub icon. Below the box, it says "GitHub Copilot used @wokspace" and "Used 7 references >".

Further down, it says "Sure, you can create a sample data array for your recipes. Here's an example:"

```
const sampleRecipes = [
  {
    id: 1,
```

At the bottom left of the code block, there is a blue link: "How can I add pagination to the recipe list?"

This generates the sample data for the recipe application. Create a file named **recipes.js**, copy the sample data generated into this file, and save it in the **rec-app\src** folder

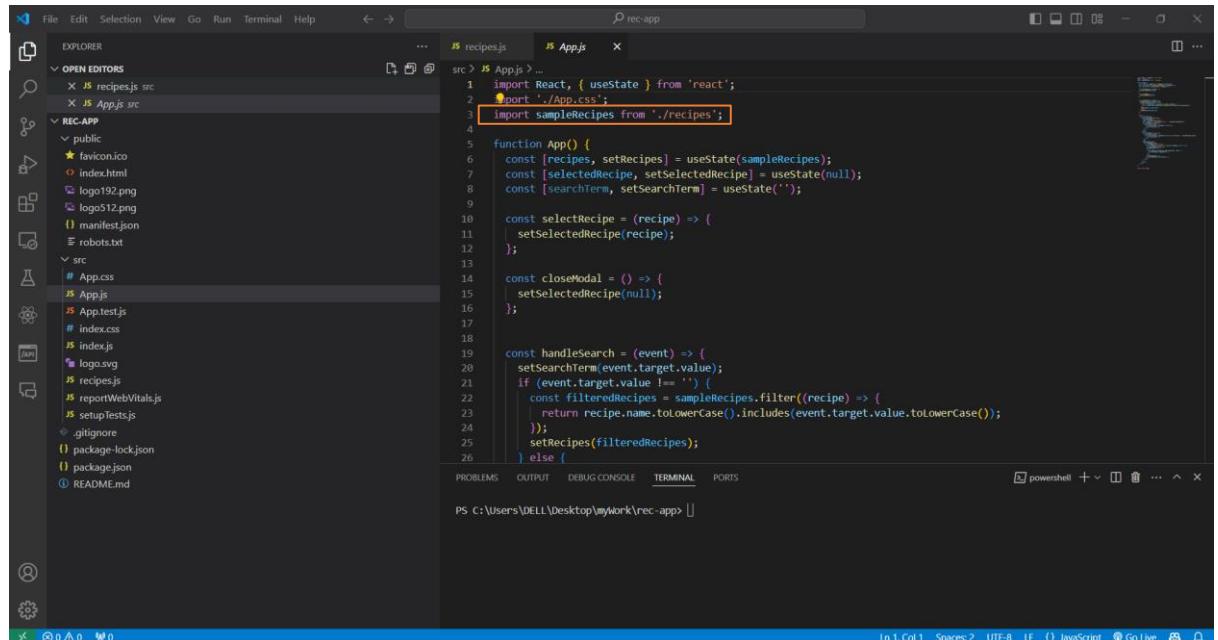
The screenshot shows the VS Code interface. The Explorer sidebar on the left shows a project structure with a file named "recipes.js" in the "src" folder under "REC-APP". The "recipes.js" file is open in the editor, displaying the following code:

```
const sampleRecipes = [
  {
    id: 1,
    name: "Spaghetti Bolognese",
    ingredients: ["Spaghetti", "Minced meat", "Tomato sauce", "Onions", "Garlic"],
    instructions: "Boil the spaghetti. Cook the minced meat in a pan. Add onions, garlic, and tomato sauce to the meat. Mix everything together and serve with the spaghetti."
  },
  {
    id: 2,
    name: "Chicken Salad",
    ingredients: ["Chicken breast", "Lettuce", "Tomatoes", "Cucumbers", "Salad dressing"],
    instructions: "Cook the chicken breast and cut it into slices. Chop the lettuce, tomatoes, and cucumbers. Mix all the ingredients together with the salad dressing and serve with the chicken salad."
  }
];
```

At the bottom of the code, there is a comment: " // Add more recipes as needed". The code ends with "export default sampleRecipes;"

The bottom status bar shows the terminal command: "PS C:\Users\DELL\Desktop\myWork\rec-app>". The status bar also indicates "Ln 18, Col 30" and "Spaces: 2".

3.2 Open **App.js** and import the newly created **recipe.js** by adding the following script:
import sampleRecipes from './recipes';

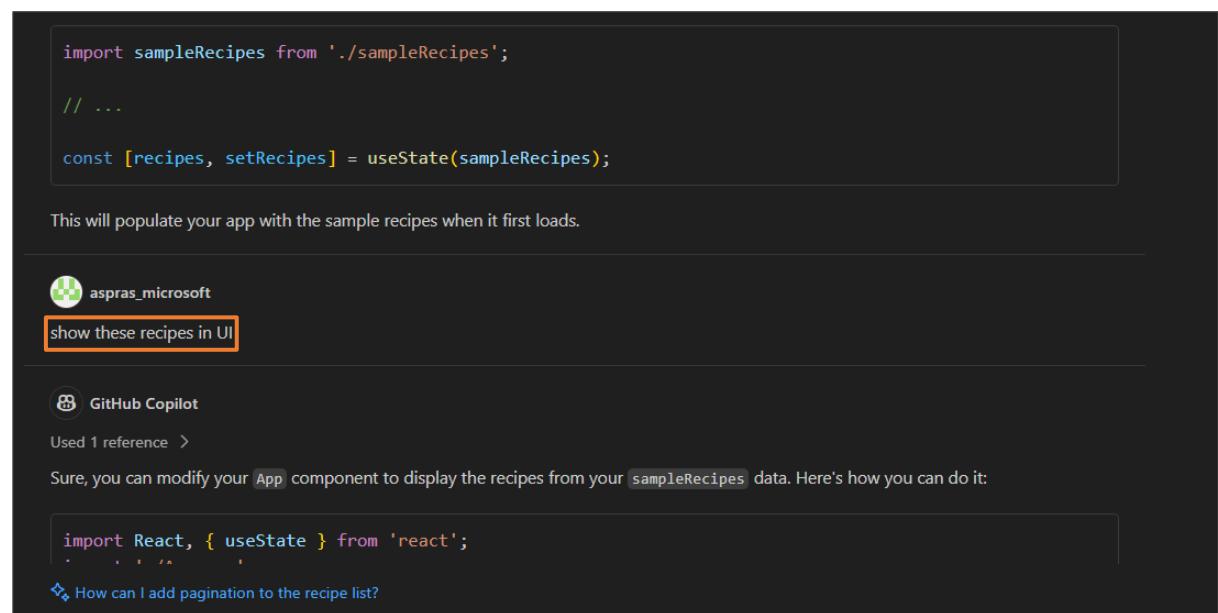


```
src > JS App.js > ...
1 import React, { useState } from 'react';
2 import './App.css';
3 import sampleRecipes from './recipes';
4
5 function App() {
6   const [recipes, setRecipes] = useState(sampleRecipes);
7   const [selectedRecipe, setSelectedRecipe] = useState(null);
8   const [searchTerm, setSearchTerm] = useState('');
9
10  const selectRecipe = (recipe) => {
11    setSelectedRecipe(recipe);
12  };
13
14  const closeModal = () => {
15    setSelectedRecipe(null);
16  };
17
18  const handleSearch = (event) => {
19    setSearchTerm(event.target.value);
20    if (event.target.value !== '') {
21      const filteredRecipes = sampleRecipes.filter(recipe => {
22        return recipe.name.toLowerCase().includes(event.target.value.toLowerCase());
23      });
24      setRecipes(filteredRecipes);
25    } else {
26    }
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\DELL\Desktop\myWork\rec-app> []
Ln 1, Col 1 Spaces: 2 UTF-8 LF {} JavaScript Go Live

```

4. Generate React code to display the recipes using the user interface

4.1 Prompt GitHub Copilot to generate code to display the recipe data using the user interface
show these recipes in UI



```
import sampleRecipes from './sampleRecipes';

// ...

const [recipes, setRecipes] = useState(sampleRecipes);
```

This will populate your app with the sample recipes when it first loads.

aspras_microsoft

show these recipes in UI

GitHub Copilot
Used 1 reference >

Sure, you can modify your `App` component to display the recipes from your `sampleRecipes` data. Here's how you can do it:

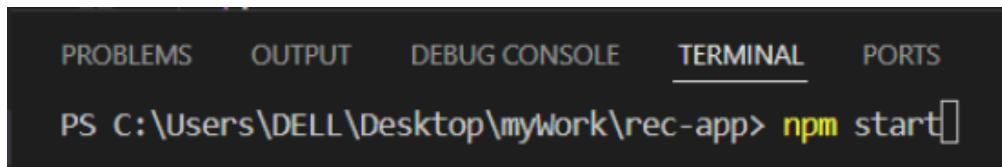
```
import React, { useState } from 'react';
.
.
.

❖ How can I add pagination to the recipe list?
```

Note: Copy the contents of the generated script and paste them into the existing `src\App.js` file

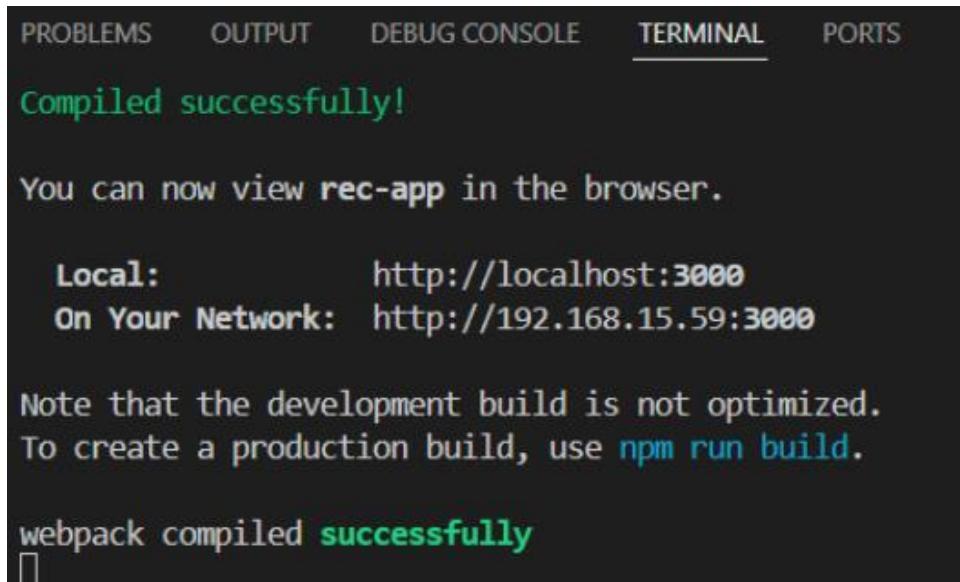
4.2 Run the react project after updating App.js

`npm start`



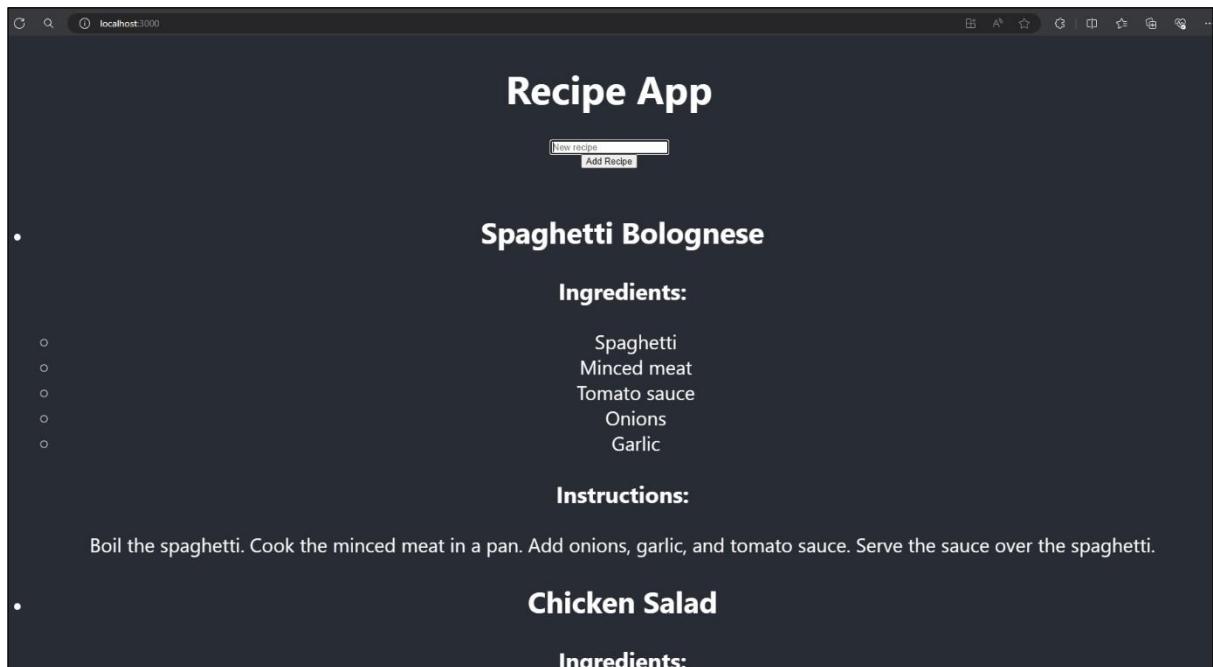
A screenshot of a terminal window. The tab bar at the top has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The terminal area shows the command `PS C:\Users\DELL\Desktop\myWork\rec-app> npm start`.

The following message will be displayed after successful execution:



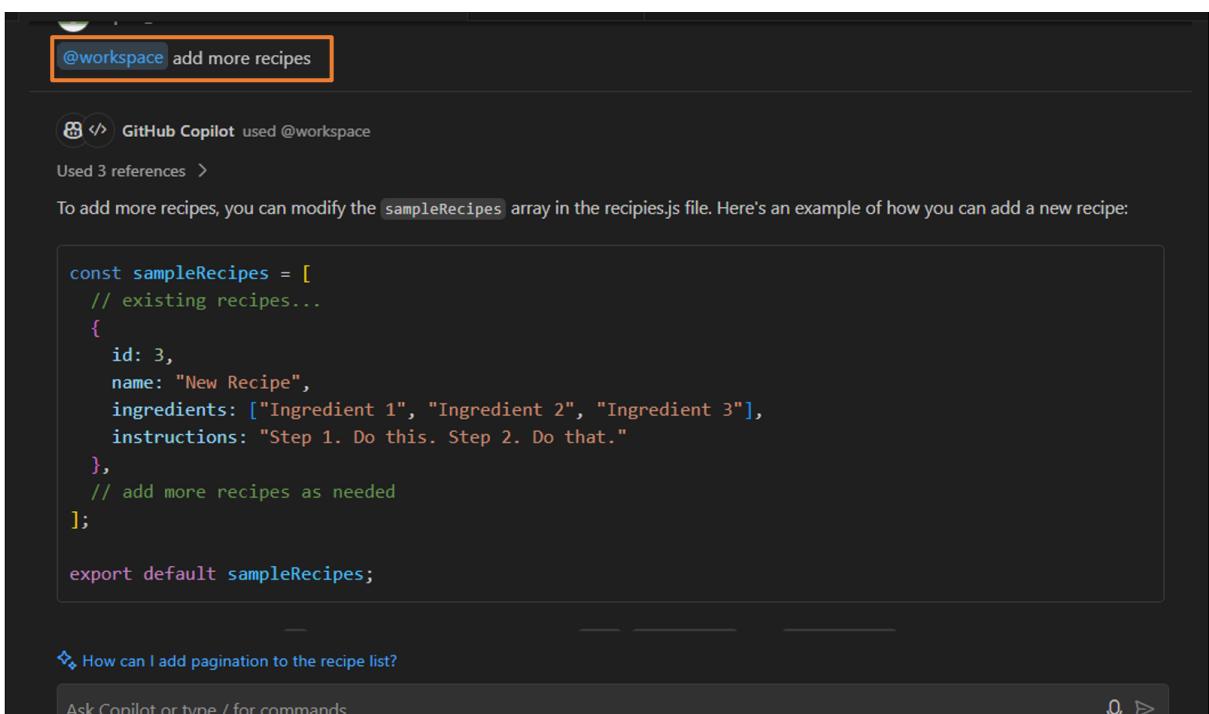
A screenshot of a terminal window. The tab bar at the top has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The terminal area displays the following text:
Compiled successfully!
You can now view `rec-app` in the browser.
Local: `http://localhost:3000`
On Your Network: `http://192.168.15.59:3000`
Note that the development build is not optimized.
To create a production build, use `npm run build`.
webpack compiled **successfully**

The output will appear as follows:



5. Generate a React code to add more recipes to the recipe app

5.1 Improve the recipe app by adding more recipes using GitHub Copilot prompts
@workspace add more recipes



The screenshot shows the GitHub Copilot interface with the following text in the input field:

```
@workspace add more recipes
```

Below the input field, the Copilot summary states: "GitHub Copilot used @workspace". It also mentions "Used 3 references >".

The main content area provides instructions: "To add more recipes, you can modify the `sampleRecipes` array in the `recipes.js` file. Here's an example of how you can add a new recipe:"

```
const sampleRecipes = [
  // existing recipes...
  {
    id: 3,
    name: "New Recipe",
    ingredients: ["Ingredient 1", "Ingredient 2", "Ingredient 3"],
    instructions: "Step 1. Do this. Step 2. Do that."
  },
  // add more recipes as needed
];
export default sampleRecipes;
```

At the bottom, there's a question from Copilot: "How can I add pagination to the recipe list?" and a footer bar with the text "Ask Copilot or type / for commands" and a "0" icon.

The following React.js code will be generated by GitHub Copilot:

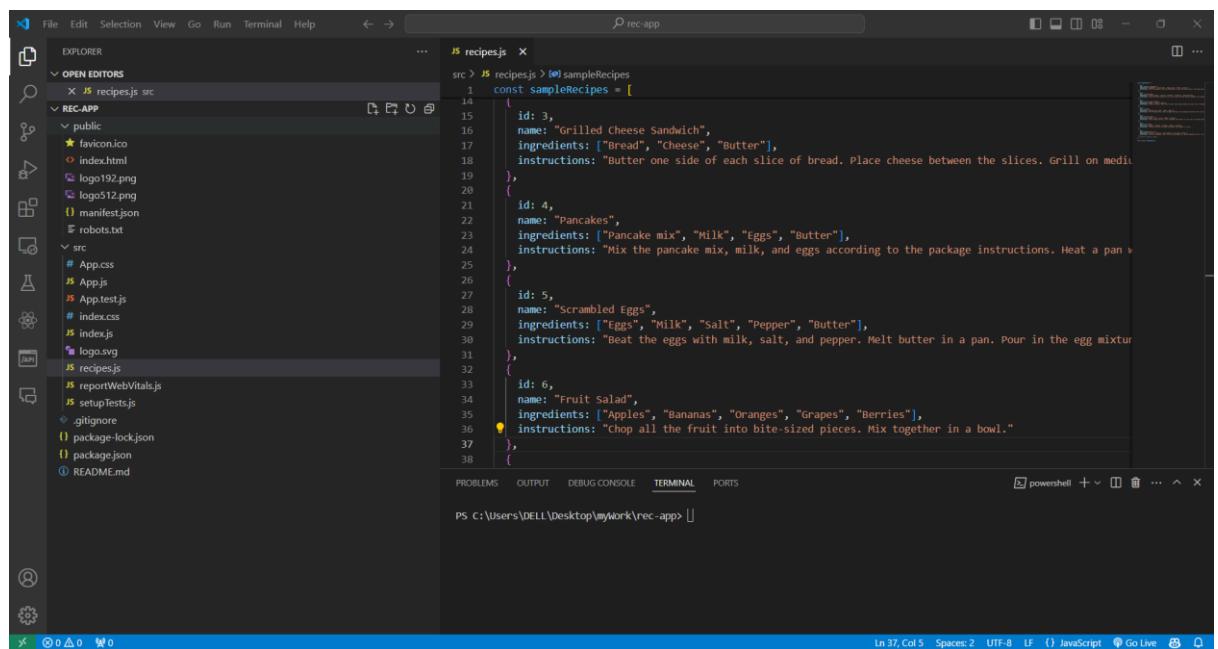
 GitHub Copilot

Used 1 reference >

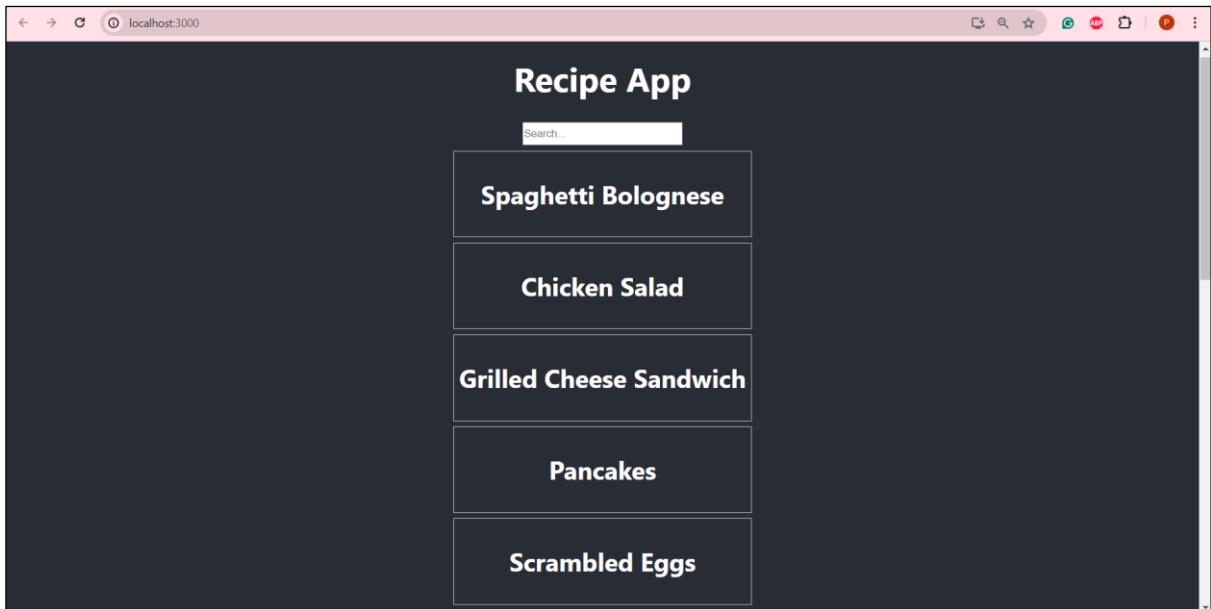
Sure, here are five more recipes you can add to your `sampleRecipes` array:

```
const sampleRecipes = [
  // existing recipes...
  {
    id: 3,
    name: "Grilled Cheese Sandwich",
    ingredients: ["Bread", "Cheese", "Butter"],
    instructions: "Butter one side of each slice of bread. Place cheese between the slices. Grill on medium heat until golden brown."
  },
  {
    id: 4,
    name: "Pancakes",
    ingredients: ["Pancake mix", "Milk", "Eggs", "Butter"],
    instructions: "Mix the pancake mix, milk, and eggs according to the package instructions. Heat a pan over medium heat and cook until golden brown on both sides."
  },
  {
    id: 5,
    name: "Omelette"
  }
]
```

Update the file **recipes.js** by copying all the generated data into it.



Execute the React project using **npm start** after updating the recipe.js file. The output will appear as follows:



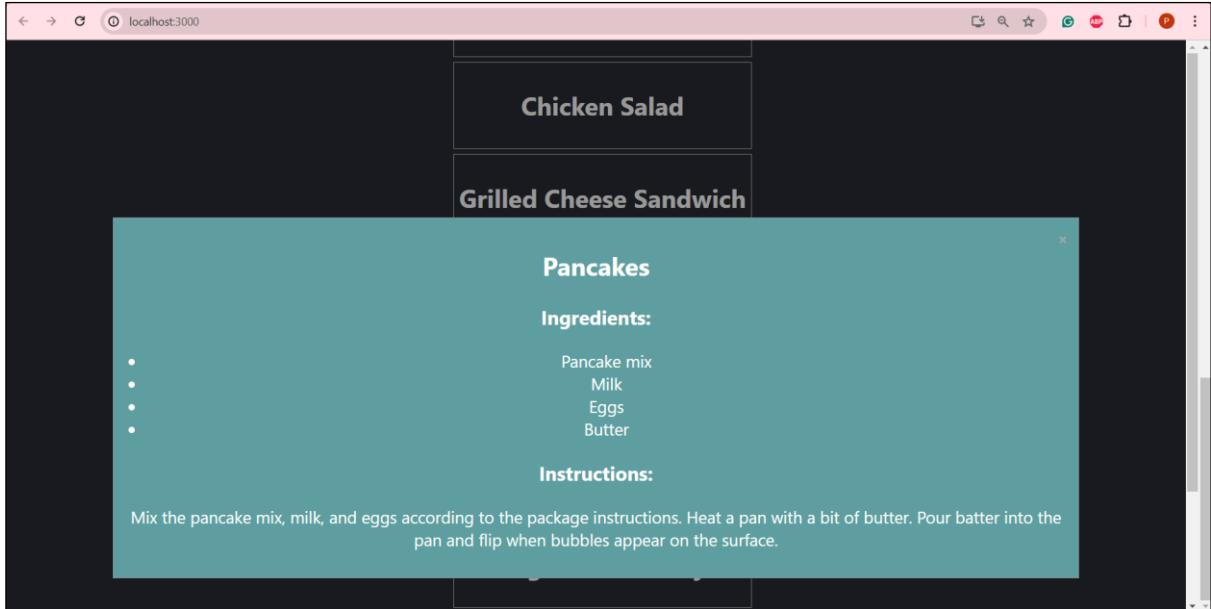
6. Create the Recipe Detail Component to display detailed information about each recipe

6.1 Prompt GitHub Copilot to generate a React code for displaying details of each recipe
show it in card format in UI, on click of card it should load the details

A screenshot of the GitHub Copilot AI interface. At the top, there is a user input field containing the text "show it in card format in UI, on click of card it should load the details". Below this, the GitHub Copilot logo and name are shown. The response section starts with "Used 1 reference >". The text reads: "Sure, you can use a card layout to display the recipes and show the details when a card is clicked. Here's how you can modify your App component to do this:". Below this is a code editor window showing a portion of a React component. The code imports React, useState, and sampleRecipes from './recipes'. It defines an App function that uses useState to manage the recipes and selectedRecipe. It also includes a selectRecipe function that updates the selectedRecipe state. At the bottom of the code editor, there is a question: "How can I add pagination to the recipe list?".

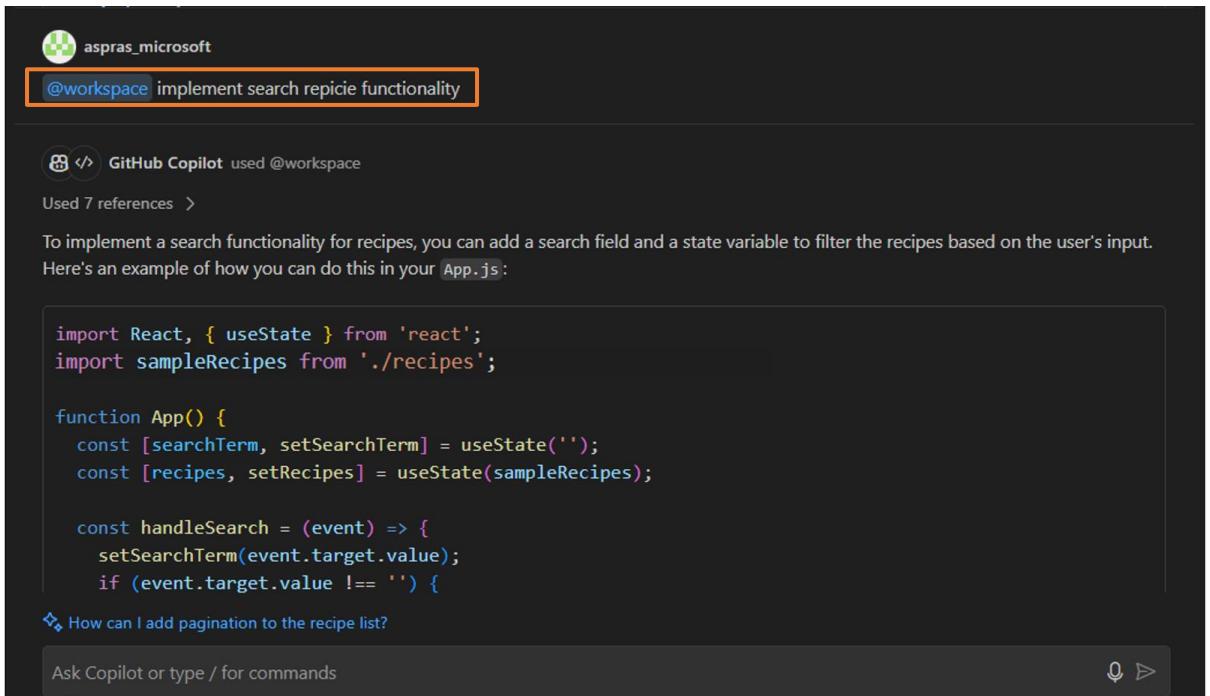
Note: Copy the contents of the generated script and paste them into the existing `src\App.js` file

After updating App.js file, execute the React project using `npm start`. The output will appear as follows:



7. Implement search functionality to enable users to search for specific recipes

7.1 Design a search feature by prompting GitHub Copilot @workspace implement search recipe functionality



The screenshot shows the GitHub Copilot interface. At the top, there's a user profile icon for 'aspras_microsoft'. Below it, a search bar contains the text '@workspace implement search recipe functionality'. A blue button labeled 'Ask Copilot' is visible. The main area displays a snippet of code for a React application:

```
import React, { useState } from 'react';
import sampleRecipes from './recipes';

function App() {
  const [searchTerm, setSearchTerm] = useState('');
  const [recipes, setRecipes] = useState(sampleRecipes);

  const handleSearch = (event) => {
    setSearchTerm(event.target.value);
    if (event.target.value !== '') {
```

Below the code, a note says 'How can I add pagination to the recipe list?'. At the bottom, there's a text input field with 'Ask Copilot or type / for commands' and a button with a question mark icon.

Note: Copy the contents of the generated script and paste them into the existing `src\App.js` file

The output will appear as follows:

