Lesson 05 Demo 04

Generating Component Architecture and Building a Structured React Blog Using GitHub Copilot

Objective: To generate component architecture and build a structured React blog application using GitHub Copilot for efficient development and code management

Tools required: Visual Studio Code and GitHub Copilot

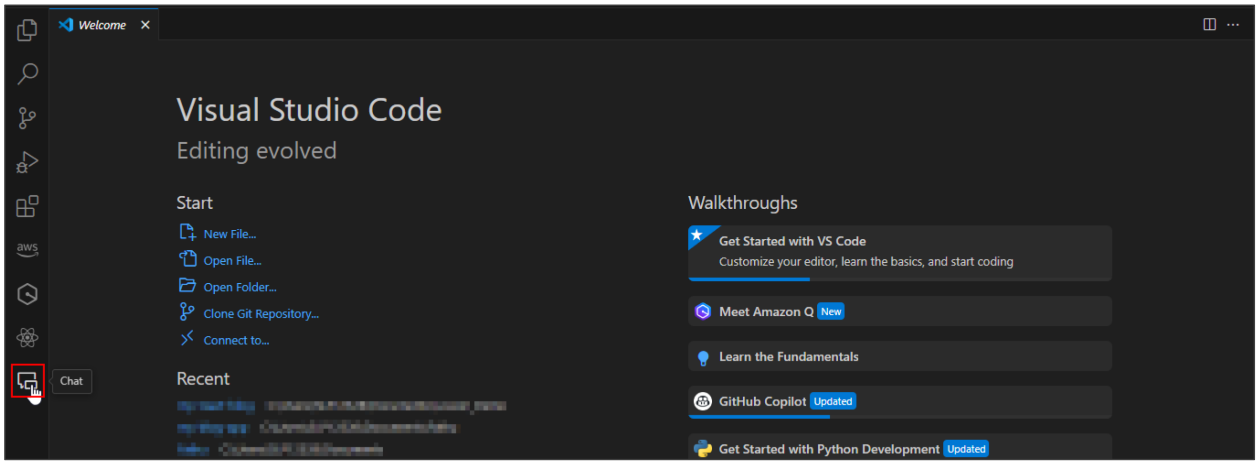
Prerequisites: None

Steps to be followed:
1. Generate component architecture and code using GitHub Copilot

Note: Please be advised that GitHub Copilot, as an artificial intelligence tool, can produce varied outputs even when presented with similar prompts.

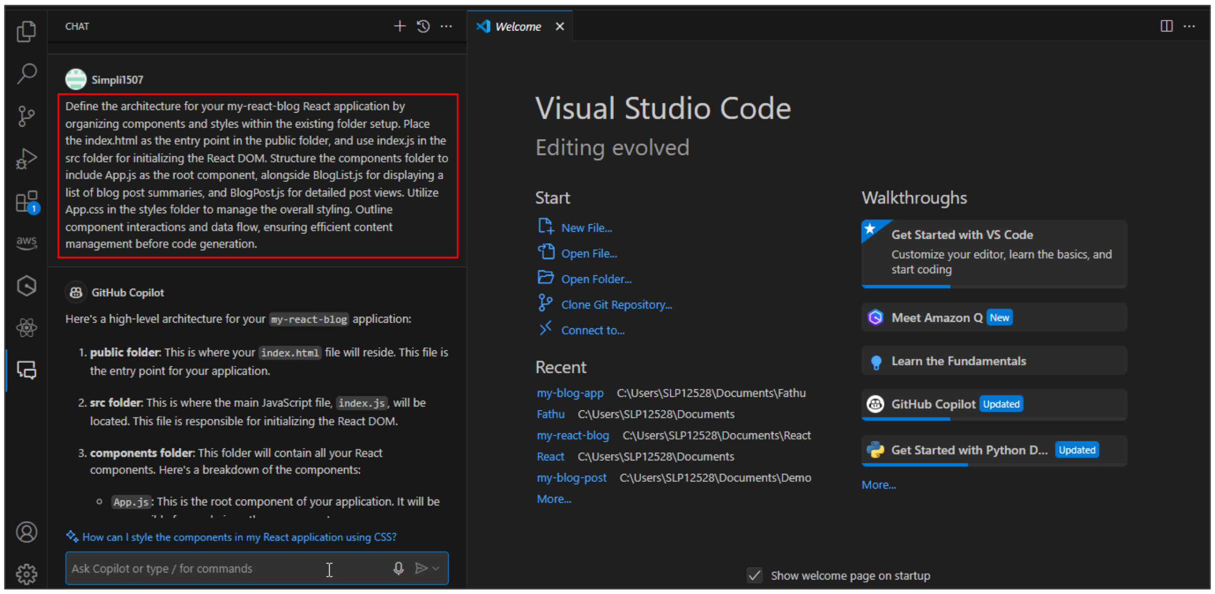
Step 1: Generate component architecture and code using GitHub Copilot

1.1 Click on the **Chat** icon, as shown in the screenshot below:



1.2 Generate the architecture using the following prompt, as shown in the screenshots below:

Define the architecture for your my-react-blog React application by organizing components and styles within the existing folder setup. Place the index.html as the entry point in the public folder, and use index.js in the src folder for initializing the React DOM. Structure the components folder to include App.js as the root component, alongside BlogList.js for displaying a list of blog post summaries, and BlogPost.js for detailed post views. Utilize App.css in the styles folder to manage the overall styling. Outline component interactions and data flow, ensuring efficient content management before code generation.


CHAT
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
my-blog-app C:\Users\SLP12528\Documents\Fathu
Fathu C:\Users\SLP12528\Documents
my-react-blog C:\Users\SLP12528\Documents\React
React C:\Users\SLP12528\Documents
my-blog-post C:\Users\SLP12528\Documents\Demo
More...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Meet Amazon Q New
Learn the Fundamentals
GitHub Copilot Updated
Get Started with Python D... Updated
More...
Show welcome page on startup
Ask Copilot or type / for commands

Simpli1507

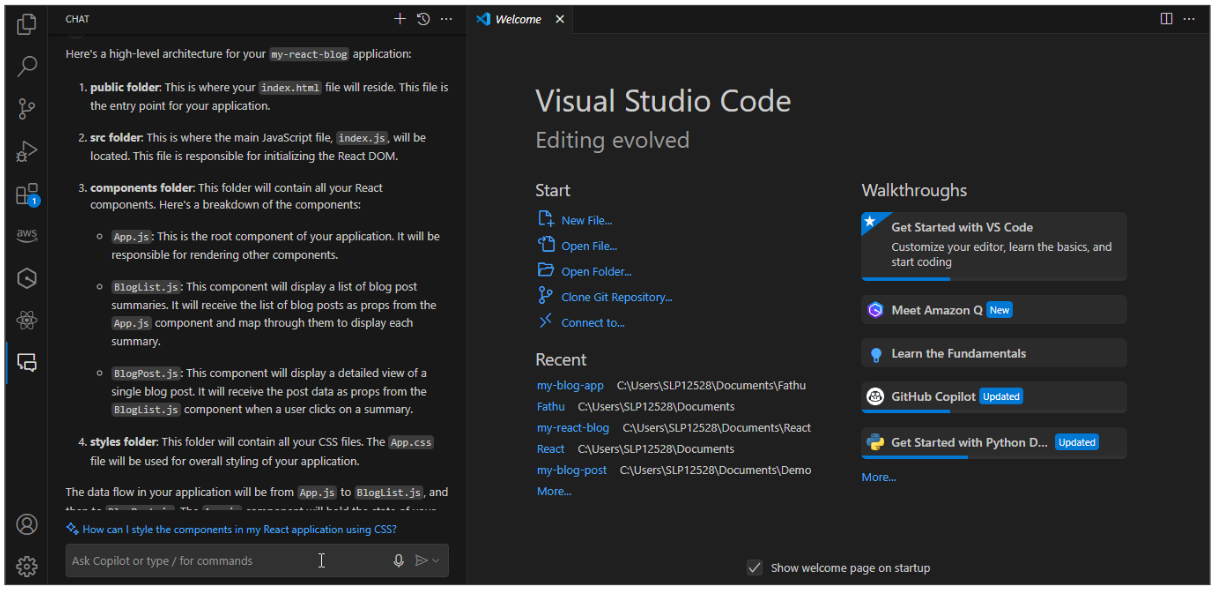
Define the architecture for your my-react-blog React application by organizing components and styles within the existing folder setup. Place the index.html as the entry point in the public folder, and use index.js in the src folder for initializing the React DOM. Structure the components folder to include App.js as the root component, alongside BlogList.js for displaying a list of blog post summaries, and BlogPost.js for detailed post views. Utilize App.css in the styles folder to manage the overall styling. Outline component interactions and data flow, ensuring efficient content management before code generation.

GitHub Copilot

Here's a high-level architecture for your `my-react-blog` application:

1. **public folder**: This is where your `index.html` file will reside. This file is the entry point for your application.
2. **src folder**: This is where the main JavaScript file, `index.js`, will be located. This file is responsible for initializing the React DOM.
3. **components folder**: This folder will contain all your React components. Here's a breakdown of the components:
   - `App.js`: This is the root component of your application. It will be

How can I style the components in my React application using CSS?

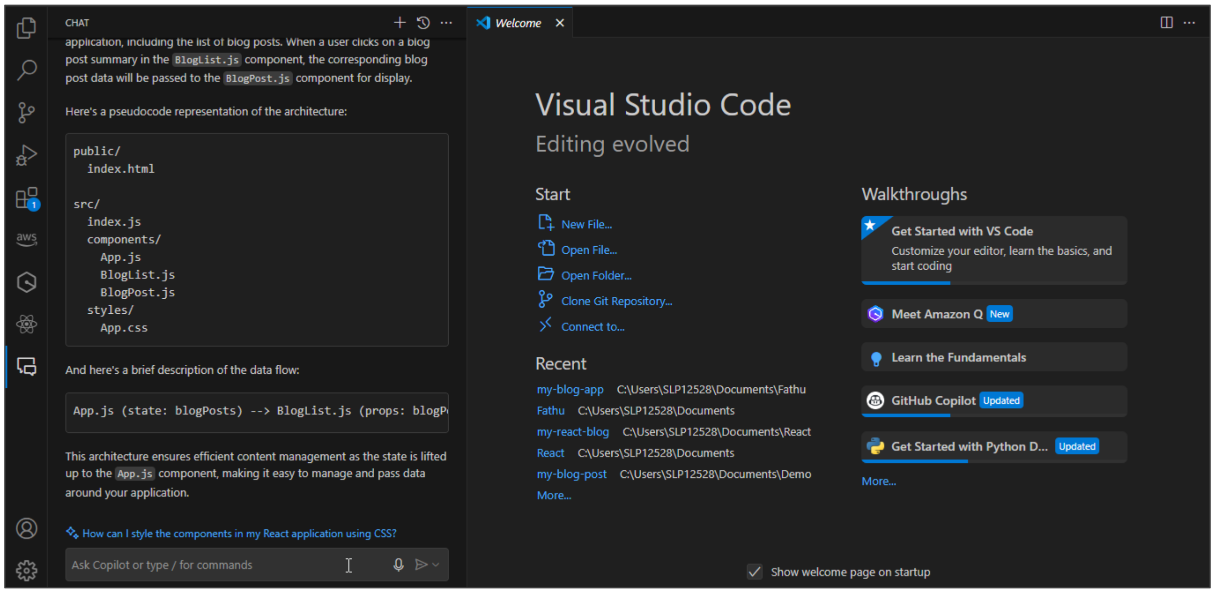

CHAT
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
my-blog-app C:\Users\SLP12528\Documents\Fathu
Fathu C:\Users\SLP12528\Documents
my-react-blog C:\Users\SLP12528\Documents\React
React C:\Users\SLP12528\Documents
my-blog-post C:\Users\SLP12528\Documents\Demo
More...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Meet Amazon Q New
Learn the Fundamentals
GitHub Copilot Updated
Get Started with Python D... Updated
More...
Show welcome page on startup
Ask Copilot or type / for commands

Here's a high-level architecture for your `my-react-blog` application:

1. **public folder**: This is where your `index.html` file will reside. This file is the entry point for your application.
2. **src folder**: This is where the main JavaScript file, `index.js`, will be located. This file is responsible for initializing the React DOM.
3. **components folder**: This folder will contain all your React components. Here's a breakdown of the components:
   - `App.js`: This is the root component of your application. It will be responsible for rendering other components.
   - `BlogList.js`: This component will display a list of blog post summaries. It will receive the list of blog posts as props from the `App.js` component and map through them to display each summary.
   - `BlogPost.js`: This component will display a detailed view of a single blog post. It will receive the post data as props from the `BlogList.js` component when a user clicks on a summary.
4. **styles folder**: This folder will contain all your CSS files. The `App.css` file will be used for overall styling of your application.

The data flow in your application will be from `App.js` to `BlogList.js`, and

How can I style the components in my React application using CSS?



CHAT

application, including the list of blog posts. When a user clicks on a blog post summary in the `BlogList.js` component, the corresponding blog post data will be passed to the `BlogPost.js` component for display.

Here's a pseudocode representation of the architecture:

```
public/
  index.html

src/
  index.js
  components/
    App.js
    BlogList.js
    BlogPost.js
  styles/
    App.css
```

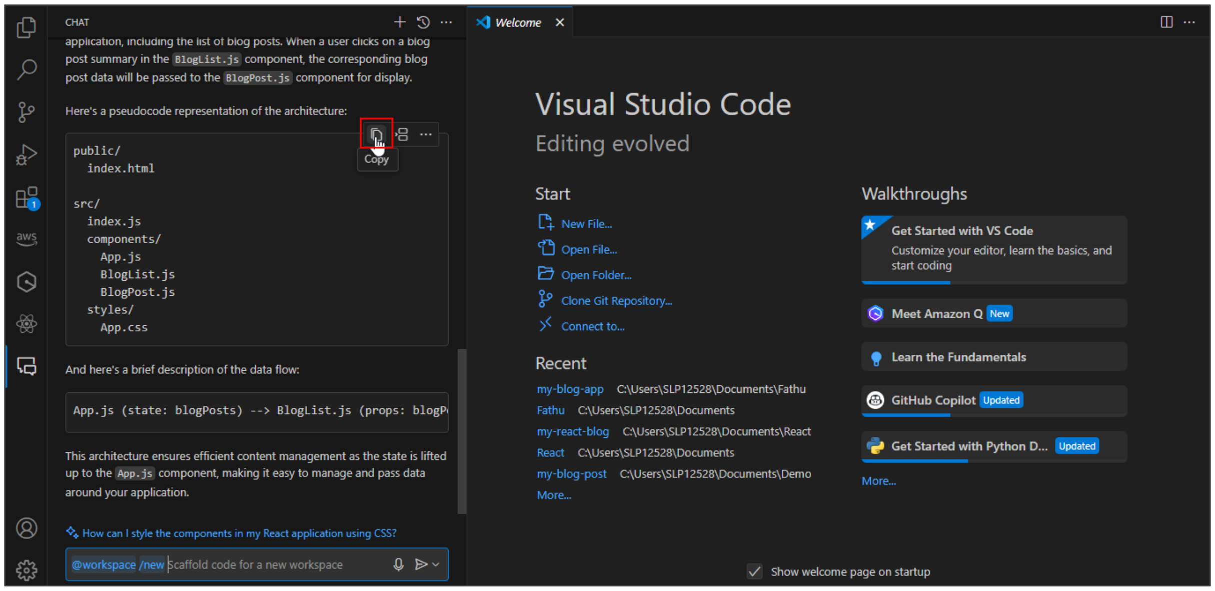
And here's a brief description of the data flow:

```
App.js (state: blogPosts) --> BlogList.js (props: blogP
```

This architecture ensures efficient content management as the state is lifted up to the `App.js` component, making it easy to manage and pass data around your application.

How can I style the components in my React application using CSS?
Ask Copilot or type / for commands
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
my-blog-app C:\Users\SLP12528\Documents\Fathu
Fathu C:\Users\SLP12528\Documents
my-react-blog C:\Users\SLP12528\Documents\React
React C:\Users\SLP12528\Documents
my-blog-post C:\Users\SLP12528\Documents\Demo
More...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Meet Amazon Q New
Learn the Fundamentals
GitHub Copilot Updated
Get Started with Python D... Updated
More...
Show welcome page on startup

1.3 Click on the **Copy** icon and save the pseudocode representation of the architecture, as shown in the screenshot below:



CHAT

application, including the list of blog posts. When a user clicks on a blog post summary in the `BlogList.js` component, the corresponding blog post data will be passed to the `BlogPost.js` component for display.

Here's a pseudocode representation of the architecture:

```
public/
  index.html

src/
  index.js
  components/
    App.js
    BlogList.js
    BlogPost.js
  styles/
    App.css
```

And here's a brief description of the data flow:

```
App.js (state: blogPosts) --> BlogList.js (props: blogP
```

This architecture ensures efficient content management as the state is lifted up to the `App.js` component, making it easy to manage and pass data around your application.

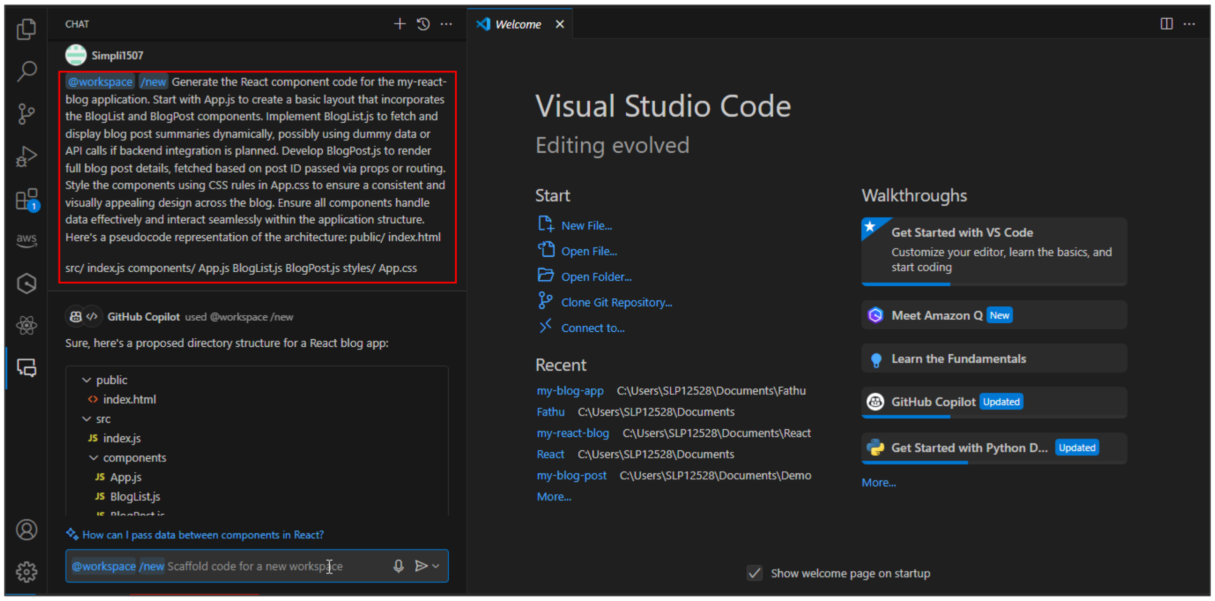
Copy
How can I style the components in my React application using CSS?
@workspace /new Scaffold code for a new workspace
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
my-blog-app C:\Users\SLP12528\Documents\Fathu
Fathu C:\Users\SLP12528\Documents
my-react-blog C:\Users\SLP12528\Documents\React
React C:\Users\SLP12528\Documents
my-blog-post C:\Users\SLP12528\Documents\Demo
More...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Meet Amazon Q New
Learn the Fundamentals
GitHub Copilot Updated
Get Started with Python D... Updated
More...
Show welcome page on startup

1.4 Use the following prompt along with the pseudocode representation of the architecture, as shown in the screenshot below:

Generate the React component code for the my-react-blog application. Start with App.js to create a basic layout that incorporates the BlogList and BlogPost components. Implement BlogList.js to fetch and display blog post summaries dynamically, possibly using dummy data or API calls if backend integration is planned. Develop BlogPost.js to render full blog post details, fetched based on post ID passed via props or routing. Style the components using CSS rules in App.css to ensure a consistent and visually appealing design across the blog. Ensure all components handle data effectively and interact seamlessly within the application structure. Here's a pseudocode representation of the architecture:

```
public/
  index.html

src/
  index.js
  components/
    App.js
    BlogList.js
    BlogPost.js
  styles/
    App.css
```



Simpli1507

@workspace /new Generate the React component code for the my-react-blog application. Start with App.js to create a basic layout that incorporates the BlogList and BlogPost components. Implement BlogList.js to fetch and display blog post summaries dynamically, possibly using dummy data or API calls if backend integration is planned. Develop BlogPost.js to render full blog post details, fetched based on post ID passed via props or routing. Style the components using CSS rules in App.css to ensure a consistent and visually appealing design across the blog. Ensure all components handle data effectively and interact seamlessly within the application structure. Here's a pseudocode representation of the architecture: public/ index.html

src/ index.js components/ App.js BlogList.js BlogPost.js styles/ App.css

GitHub Copilot used @workspace /new

Sure, here's a proposed directory structure for a React blog app:

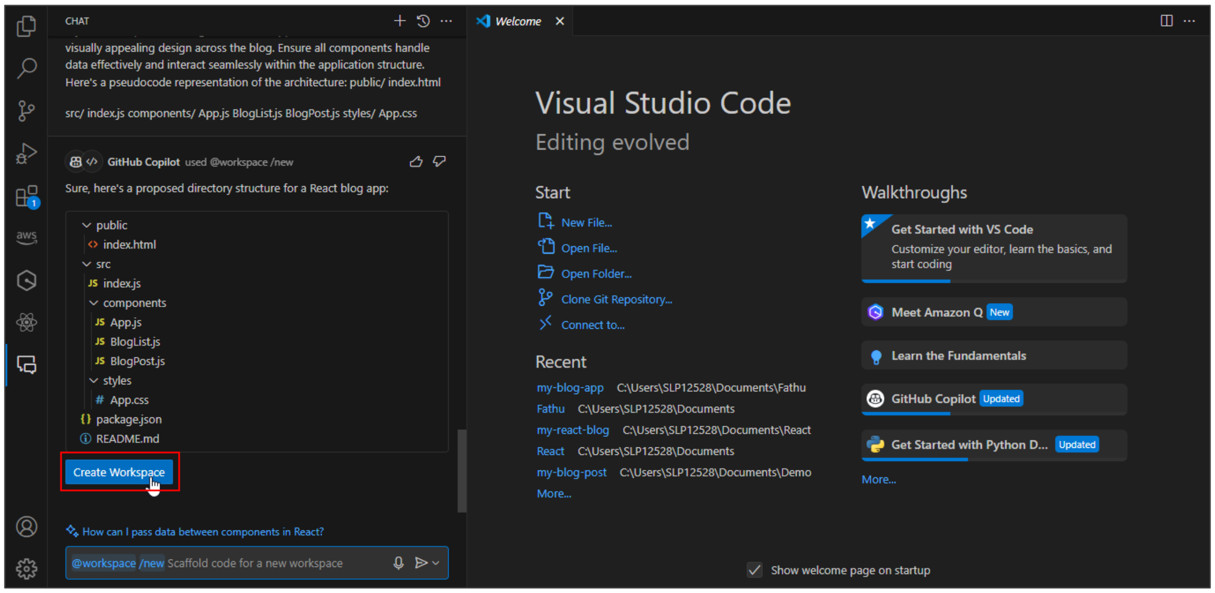
```
public
  index.html
src
  index.js
  components
    App.js
    BlogList.js
```

How can I pass data between components in React?

@workspace /new Scaffold code for a new workspace

CHAT
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
my-blog-app C:\Users\SLP12528\Documents\Fathu
Fathu C:\Users\SLP12528\Documents
my-react-blog C:\Users\SLP12528\Documents\React
React C:\Users\SLP12528\Documents
my-blog-post C:\Users\SLP12528\Documents\Demo
More...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Meet Amazon Q New
Learn the Fundamentals
GitHub Copilot Updated
Get Started with Python D... Updated
More...
Show welcome page on startup

1.5 Click on the **Create Workspace** button, as shown in the screenshot below:


visually appealing design across the blog. Ensure all components handle data effectively and interact seamlessly within the application structure. Here's a pseudocode representation of the architecture: public/ index.html

src/ index.js components/ App.js BlogList.js BlogPost.js styles/ App.css

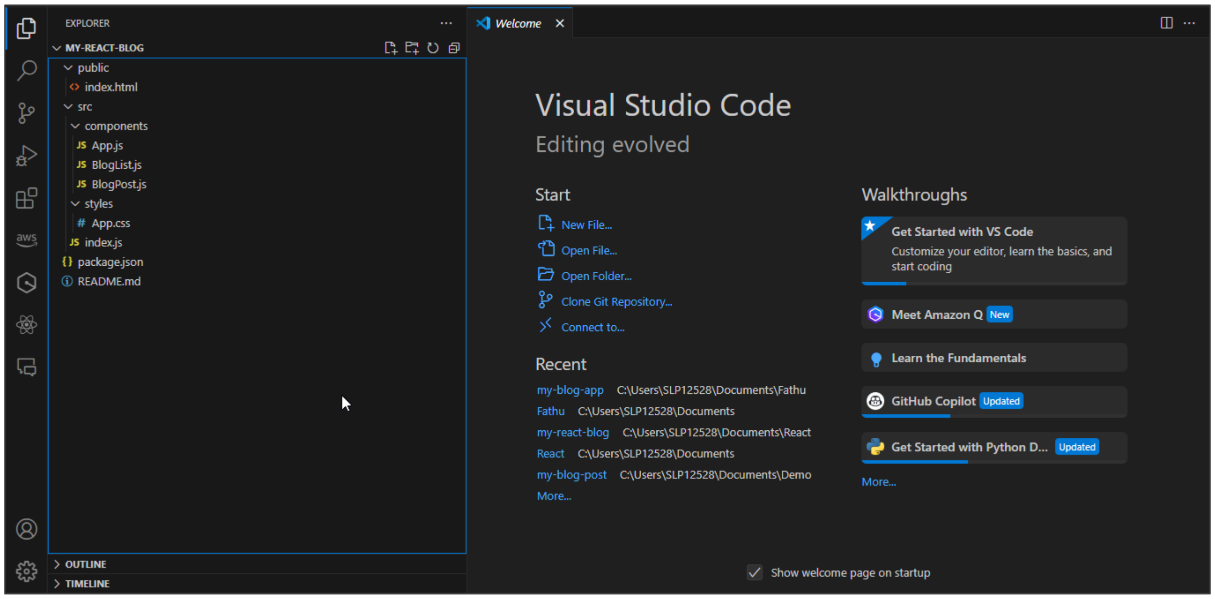
GitHub Copilot used @workspace /new

Sure, here's a proposed directory structure for a React blog app:

```
public
  index.html
src
  index.js
  components
    App.js
    BlogList.js
    BlogPost.js
  styles
    App.css
package.json
README.md
```

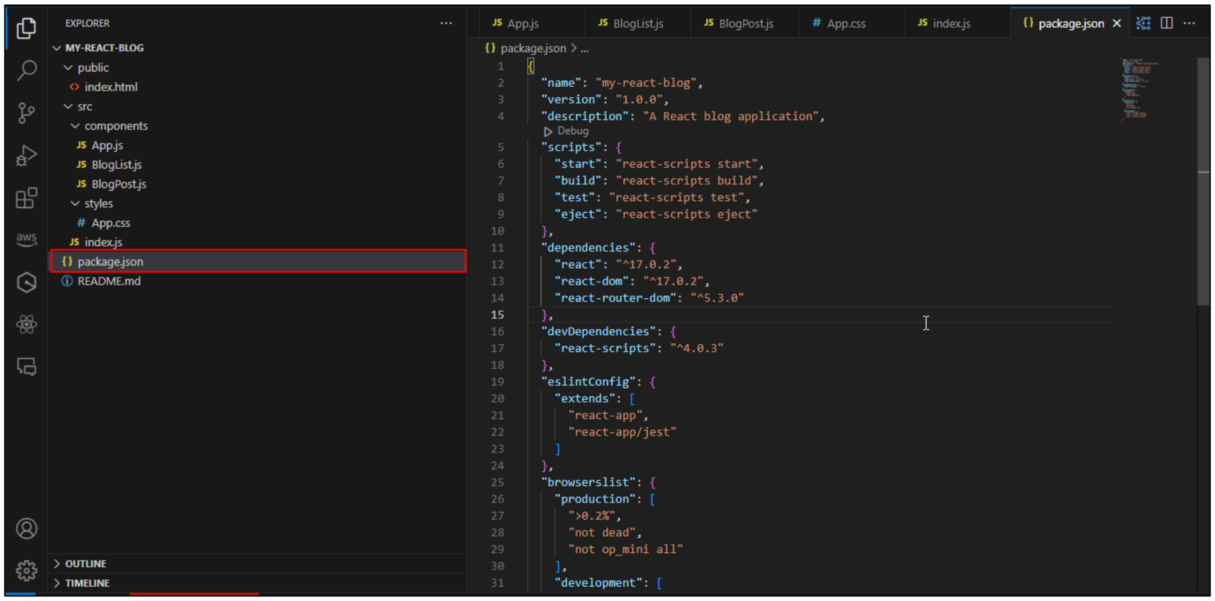
CHAT
Create Workspace
How can I pass data between components in React?
@workspace /new Scaffold code for a new workspace
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
my-blog-app C:\Users\SLP12528\Documents\Fathu
Fathu C:\Users\SLP12528\Documents
my-react-blog C:\Users\SLP12528\Documents\React
React C:\Users\SLP12528\Documents
my-blog-post C:\Users\SLP12528\Documents\Demo
More...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Meet Amazon Q New
Learn the Fundamentals
GitHub Copilot Updated
Get Started with Python D... Updated
More...
Show welcome page on startup

You will see the following interface:


```
MY-REACT-BLOG
  public
    index.html
  src
    components
      App.js
      BlogList.js
      BlogPost.js
    styles
      App.css
    index.js
  package.json
  README.md
```

EXPLORER
OUTLINE
TIMELINE
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
my-blog-app C:\Users\SLP12528\Documents\Fathu
Fathu C:\Users\SLP12528\Documents
my-react-blog C:\Users\SLP12528\Documents\React
React C:\Users\SLP12528\Documents
my-blog-post C:\Users\SLP12528\Documents\Demo
More...
Walkthroughs
Get Started with VS Code
Customize your editor, learn the basics, and start coding
Meet Amazon Q New
Learn the Fundamentals
GitHub Copilot Updated
Get Started with Python D... Updated
More...
Show welcome page on startup

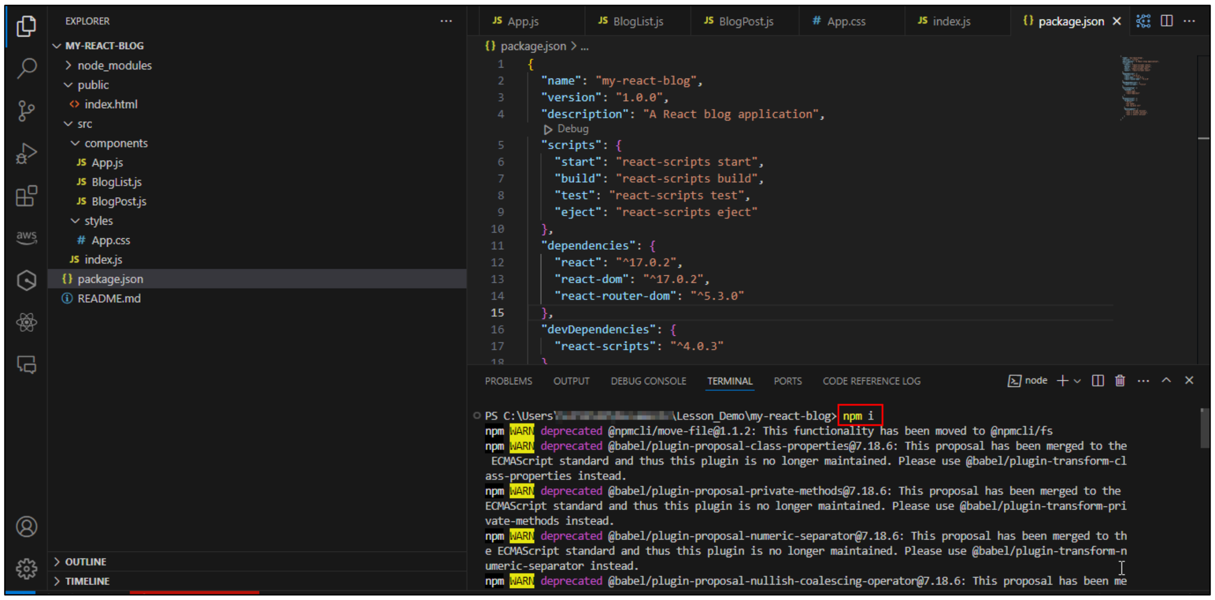
1.6 Click on the **package.json** file, as shown in the screenshot below:



```json
{
  "name": "my-react-blog",
  "version": "1.0.0",
  "description": "A React blog application",
  ▷ Debug
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "dependencies": {
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-router-dom": "^5.3.0"
  },
  "devDependencies": {
    "react-scripts": "^4.0.3"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
```

Note: View all the files for your reference and make the changes per the requirements

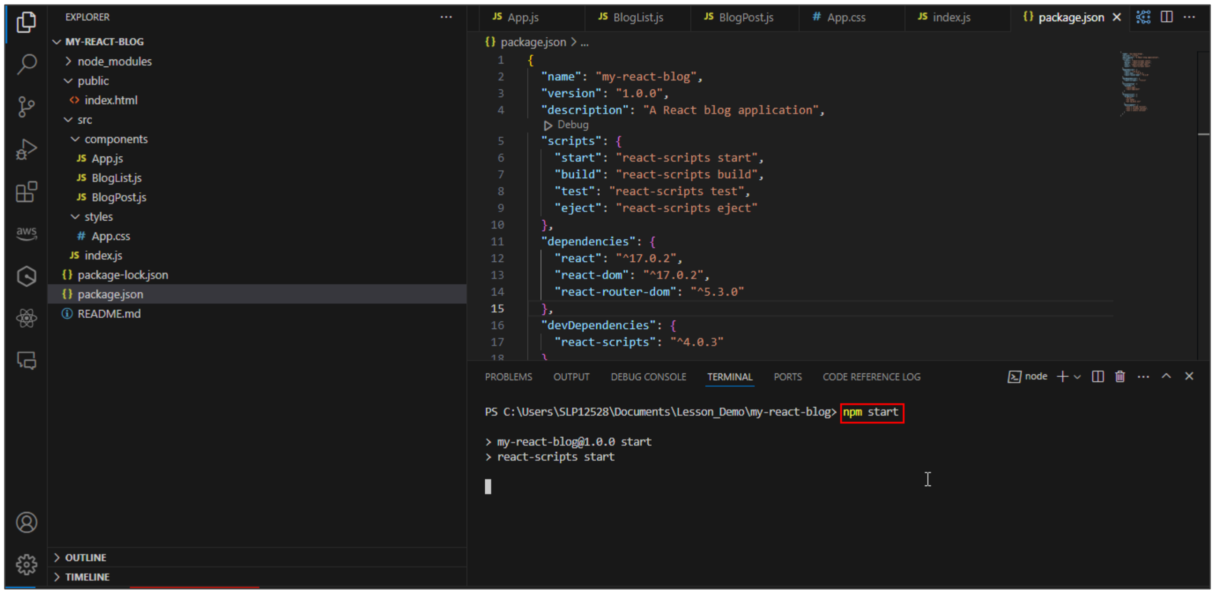
1.7 Open a new terminal and execute the following command to download and install all project dependencies listed in the **package.json** file into the **node_modules**:
npm i



```json
{
  "name": "my-react-blog",
  "version": "1.0.0",
  "description": "A React blog application",
  ▷ Debug
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "dependencies": {
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-router-dom": "^5.3.0"
  },
  "devDependencies": {
    "react-scripts": "^4.0.3"
```

```
PS C:\Users\[redacted]\Lesson_Demo\my-react-blog> npm i
npm WARN deprecated @npmcli/move-file@1.1.2: This functionality has been moved to @npmcli/fs
npm WARN deprecated @babel/plugin-proposal-class-properties@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-cl
ass-properties instead.
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-pri
vate-methods instead.
npm WARN deprecated @babel/plugin-proposal-numeric-separator@7.18.6: This proposal has been merged to th
e ECMAScript standard and thus this plugin is no longer maintained. Please use @babel/plugin-transform-n
umeric-separator instead.
npm WARN deprecated @babel/plugin-proposal-nullish-coalescing-operator@7.18.6: This proposal has been me
```

1.8 Execute the following to launch the application, as shown in the screenshot below:
npm start

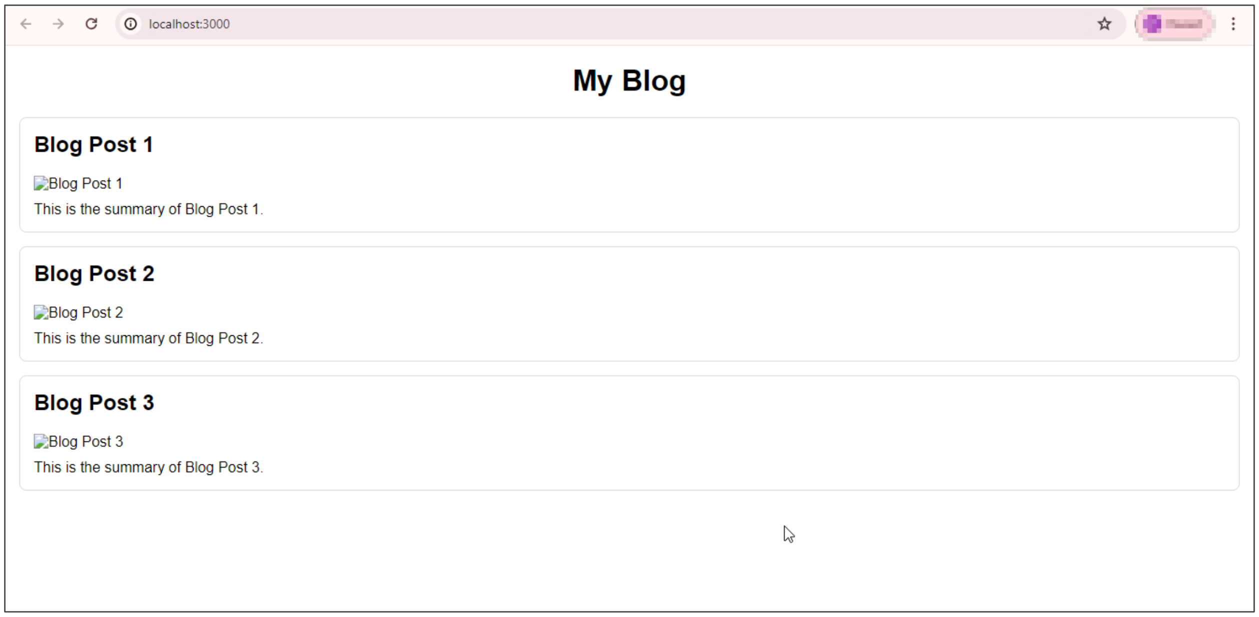


```json
{
  "name": "my-react-blog",
  "version": "1.0.0",
  "description": "A React blog application",
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "dependencies": {
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-router-dom": "^5.3.0"
  },
  "devDependencies": {
    "react-scripts": "^4.0.3"
```

```
PS C:\Users\SLP12528\Documents\Lesson_Demo\my-react-blog> npm start

> my-react-blog@1.0.0 start
> react-scripts start
```

You will see the following interface:



My Blog

Blog Post 1

Blog Post 1

This is the summary of Blog Post 1.

Blog Post 2

Blog Post 2

This is the summary of Blog Post 2.

Blog Post 3

Blog Post 3

This is the summary of Blog Post 3.

By following these steps, you have successfully generated a component architecture and built a structured React blog application using GitHub Copilot for efficient development and code management.