COPILOT HANDBOOK

For Students and Developers

EDITION 1, JULY 2024

Author

Mahesh Chand

And Copilot, ChatGPT, and Gemini

Copyright 2024 Mahesh Chand and CSharp Inc.

Table of Contents

CHAPTER 1 INTRODUCTION....4
- What is the difference between Github Copilot and Microsoft Copilot?.... 5
- What is Github Copilot?.... 6
- Github Copilot License.... 8
- What IDEs Copilot support?.... 9
- What programming language does Copilot support?....10
- What can Copilot do for students and developers?....10
- Is Copilot free?....10
- CoPilot licensing....11
- How does GitHub Copilot work?....12
- Can GitHub Copilot help with writing documentation and comments?....12
- How accurate are the code suggestions from GitHub Copilot?....12
- Is GitHub Copilot safe to use with proprietary or sensitive code?....13
- How can I improve the quality of suggestions from GitHub Copilot?....13
- Can I customize GitHub Copilot's behavior or disable it for specific files?..13
- Does Copilot share my code with others?....13
- Can I write a complete website using Copilot?....13
- Copilot challenges and limitations....14
- Copilot in Visual Studio....15
- Ask you question....16

CHAPTER 2 WRITING BOILERPLATE CODE USING COPILOT....19
- Build a simple application....19
  - Step 1. Create a new project.....19
  - Step 2: Open Copilot chat.....19
  - Step 3. Write a simple program to read a text file.....20
  - Step 4. Review code for correctness.....21
  - Step 5. Build and run.....22
- Code review....22

CHAPTER 3 AUTOCOMPLETION USING COPILOT....24
- Extend our application.....24
- Extend existing program using Copilot....26

Copyright 2024 Mahesh Chand and CSharp Inc.

CHAPTER 4 DOCUMENATION USING COPILOT 29
  Commenting code using Copilot 29
  Documentation using Copilot 34
CHAPTER 5 CODE REFACTORING 38
    Refactor a code block 38
    Refactor a program or application 41
    Update code to new version 41
CHAPTER 6 WRITE TEST CASES USING COPILOT 42
CHAPTER 7 FIX ERRORS USING COPILOT 44
  Code Review. Again! 49
CHAPER 8 COPILOT SLASH COMMANDS 52
  What are Copilot slash commands? 52
  Slash commands explained 53
    /generate: Generate code to answer your question 53
    /explain: Explain the selected code. 53
    /doc: Add a documentation comment. 55
    /fix: Propose a fix for problems in the selected code. 56
    /tests: Create unit tests for the selected code. 56
  /optimize: Improve code performance 57
  Reference: scope Copilot results to a particular file or entire solution 58
CHAPTER 9 CODE CONVERSION 60
SUMMARY 62

Copyright 2024 Mahesh Chand and CSharp Inc.

ABOUT THE AUTHOR

Mahesh Chand is a distinguished figure in the software development and technology industry, with over 25 years of experience as an award-winning Software Architect and Author. He has built a reputation for creating innovative software solutions using cutting-edge technologies.

As a serial entrepreneur, Mahesh has founded several successful companies, including CSharp Inc. and Mindcracker. He is also the founder of C# Corner, a leading online community for software developers that boasts 29.4 million annual visitors and 3 million registered members.

His contributions to the tech industry have earned him notable recognition, including being a former Microsoft Regional Director and a 14-time Microsoft MVP (Most Valuable Professional). Recently, he was inducted into the Forbes Business Council.

In addition to his entrepreneurial ventures, Mahesh is an accomplished author of several programming books and an international keynote speaker. He continues to shape the tech industry through his leadership roles in various initiatives, including CSharpTV, Web3Universe, BCrypt, and HackIndia.

Mahesh actively contributes to developer education and growth through his platforms and speaking engagements. His commitment to sharing knowledge and fostering innovation has solidified his position as a prominent figure in the software development community.

“Education is the most powerful gift one can give to people. Our mission with C# Corner is to empower its members not only become better software professionals but also become better leaders, and better human beings.”

Mahesh Chand

Founder C# Corner

Copyright 2024 Mahesh Chand and CSharp Inc.

CHAPTER 1

INTRODUCTION

Welcome to ‘COPILOT HANDBOOK for students and developers,’ a book published by CSharp.com. Copilot serves as the trusted companion for computer science students, programmers, developers, QA and test engineers, and technical writers.

If you’re using one or more of the Microsoft products such as Windows PC, Windows Operating System, Office 365, Azure, Visual Studio, or Edge, you will see Copilot almost everywhere.

There are two Copilots – Microsoft Copilot and Github Copilot. Microsoft Copilot is a general Copilot and is available through various Microsoft products, including Windows, Bing, Edge, Office 365, Teams, Power Platform, and Azure.

Github Copilot is specifically designed for programmers and developers who actively write and maintain software code. This book focuses on Github Copilot.

In this book, you’ll learn the fundamentals of Github Copilot and discover how to leverage its various features to enhance productivity, efficiency, and code quality using Copilot and Visual Studio. By the time you finish reading and practicing this book, you should be able to boost your productivity and the quality of the code you write.

What is the difference between Github Copilot and Microsoft Copilot?

Microsoft Copilot and GitHub Copilot are different, though they share similarities in their purpose and technology base.

While both GitHub Copilot and Microsoft Copilot are AI-powered tools designed to assist users, GitHub Copilot is tailored specifically for coding and development, whereas Microsoft Copilot encompasses a broader range of productivity enhancements across various Microsoft applications.

GitHub Copilot

Developed collaboratively by GitHub, OpenAI, and Microsoft.

- Specifically designed to assist developers by providing autocomplete-style code suggestions directly within their text editors or IDEs.
- It integrates with various development environments such as Visual Studio Code, Visual Studio, Vim, Neovim, the JetBrains suite of IDEs, and Azure Data Studio.

Copyright 2024 Mahesh Chand and CSharp Inc.

- It focuses primarily on enhancing the coding experience by suggesting code snippets, functions, and even entire classes based on the context of the code being written.

Microsoft Copilot

Generally, refers to a suite of AI tools integrated into Microsoft products, aimed at enhancing productivity across various applications.

- Microsoft Copilot is available as a part of Windows operating system and can be accessed from the Windows Task Bar.
- Microsoft Copilot is also available in various Microsoft products like Office (Word, Excel, and PowerPoint), Edge browser, Teams, Power Platform, and Azure.
- In these applications, Copilot assists by generating text, creating summaries, suggesting formulas, designing slides, and more.
- It leverages AI to help users perform tasks more efficiently across different Microsoft software products.

What is Github Copilot?

GitHub Copilot is an AI coding assistant that helps you write code faster without typing too much and hence improves productivity. Having a faster coding assistant, you can focus more on your business logic and problem solving.

Github Copilot was developed by Github and OpenAI to help software teams' productivity. GitHub Copilot leverages a generative AI model developed collaboratively by GitHub, OpenAI, and Microsoft.

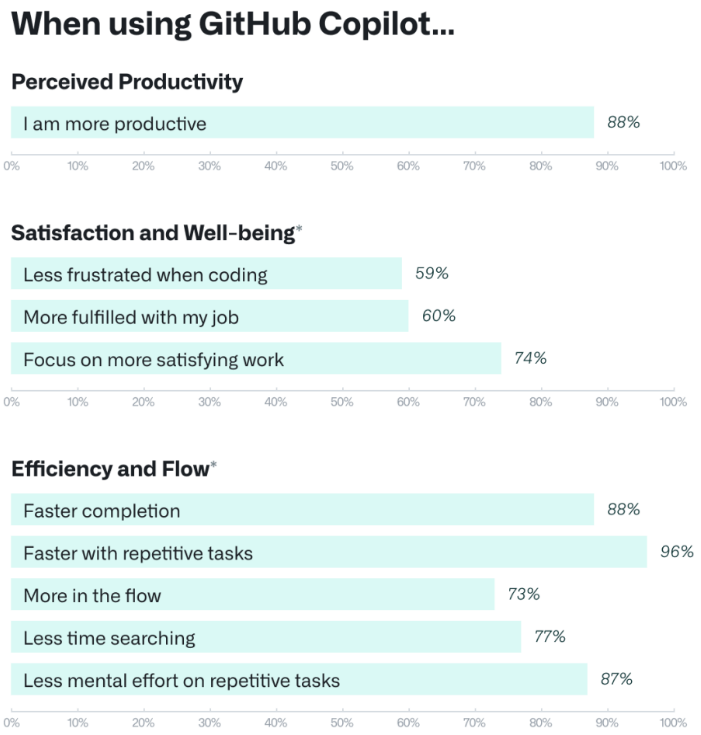
Copilot offers coding suggestions as you type: sometimes the completion of the current line, sometimes a whole new block of code. You can accept all, or part, of a suggestion, or you can ignore the suggestion and keep typing.

Using the chat feature, you can ask Copilot how best to solve a problem. Or you can ask Copilot to explain someone else's code. If your code has a bug, you can ask Copilot how to fix it.

GitHub Copilot has been proven to increase developer productivity and accelerate the pace of software development. In a survey, 88% were more productive when they used Github Copilot.

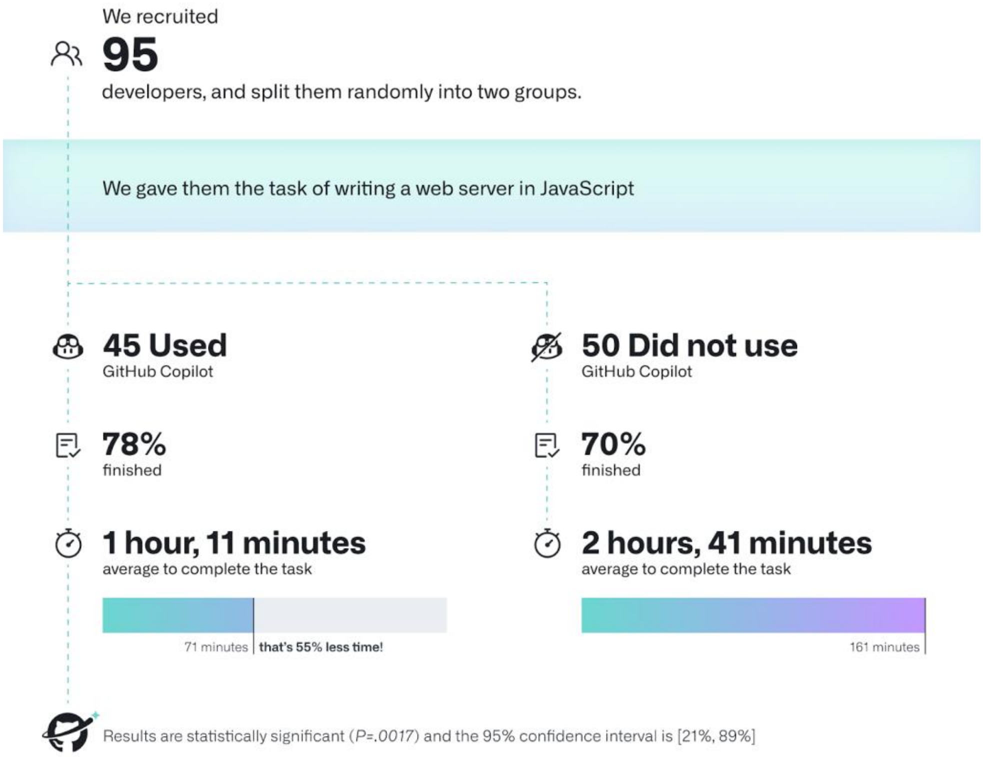
Job satisfaction is a key to developers' well-being and happiness. While using Copilot, developers were more focused on problem solving than typing code and fixing bugs.

Copyright 2024 Mahesh Chand and CSharp Inc.



Github Copilot was also able to increase the performance of development and hence delivery by 55%.

Copyright 2024 Mahesh Chand and CSharp Inc.



We recruited

95

developers, and split them randomly into two groups.

We gave them the task of writing a web server in JavaScript

45 Used
GitHub Copilot

78%
finished

1 hour, 11 minutes
average to complete the task

50 Did not use
GitHub Copilot

70%
finished

2 hours, 41 minutes
average to complete the task

Results are statistically significant (P=.0017) and the 95% confidence interval is [21%, 89%]

GitHub Copilot is available:

- In your IDE
- In GitHub Mobile, as a chat interface
- On the command line, through the GitHub CLI
- On GitHub.com, with a subscription to Copilot Enterprise

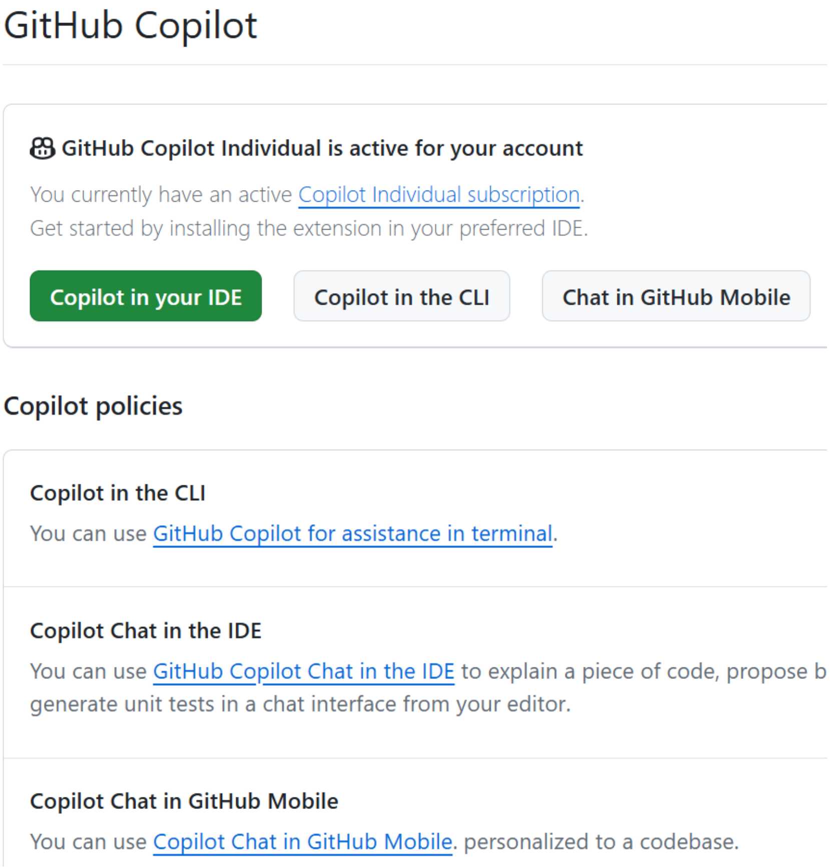
In this book, all references to Copilot are exclusively to Github Copilot.

Github Copilot License

You can buy a Copilot license via your Github.com account. Login to your Github.com account and click on Copilot menu item in the left side bar. On

Copyright 2024 Mahesh Chand and CSharp Inc.

this screen, you can buy a new or review your existing subscription.


GitHub Copilot

GitHub Copilot Individual is active for your account

You currently have an active Copilot Individual subscription.
Get started by installing the extension in your preferred IDE.

Copilot policies

Copilot in the CLI

You can use GitHub Copilot for assistance in terminal.

Copilot Chat in the IDE

You can use GitHub Copilot Chat in the IDE to explain a piece of code, propose b
generate unit tests in a chat interface from your editor.

Copilot Chat in GitHub Mobile

You can use Copilot Chat in GitHub Mobile. personalized to a codebase.

On this screen, you can also set up your subscription policies and more settings.

What IDEs Copilot support?

In this book, I’m going to use Visual Studio, but Copilot is available through the following IDEs also:

- VS Code - This is one of the first and most popular IDEs to support GitHub Copilot.
- Visual Studio - Both Visual Studio 2022 and Visual Studio 2019 are supported.
- Others - This includes a range of IDEs such as IntelliJ IDEA, PyCharm, WebStorm, Rider, CLion, and JetBrains IDEs.
- Neovim - Support for Neovim provides options for users who prefer this lightweight and highly configurable editor.
- Github Copilot is also available in Azure Data Studio.

Copyright 2024 Mahesh Chand and CSharp Inc.

What programming language does Copilot support?

Copilot supports major popular programming languages, including Python, JavaScript/TypeScript, Ruby, Go, Java, PHP, C++, C, C#, Rust, and Swift.

Copilot also supports front-end framework such as ASP.NET Core, React, and Angular.

What can Copilot do for students and developers?

The Copilot can be your best friend if you are a computer science student, programmer, or developer. What are some of the common challenges of computer science students and programmers? It takes too much time to type repetitive code again and again. Code quality could be an issue for novice programmers who do not understand the integral parts of software. Programmers dislike commenting and documenting their code. Writing test cases takes too much time.

Here is good news. Copilot can do all that for you in minutes and a lot more if you know how to use it effectively. Here is a list of few things Copilot can do for you:

- Write boilerplate code.
- Code completion and functions/methods suggestions.
- Code refactoring and implementing best practices.
- Improve code quality including error fixing.
- Explain existing project and code.
- Write code comments.
- Write test cases.
- Code translation.
- Learn programming.
- Prepare for technical interviews.

And more. You will see all these topics in the coming chapters in this book.

Is Copilot free?

While GitHub Copilot offers a paid tier with monthly or yearly subscriptions, there are ways to access it for free! Let's break down your options:

Free access for students: Verified students, teachers, and maintainers of popular open-source projects on GitHub can use Copilot Individual entirely free. For a student or teacher account, you must verify with your college email with proof.

Free trial: If you don't qualify for free access, you can still test out Copilot Individual with a 30-day free trial. After the trial, a paid subscription is required to keep using it.

Paid subscriptions: For individuals or organizations who don't qualify for free access or the trial, Copilot offers paid subscriptions. You can manage these

Copyright 2024 Mahesh Chand and CSharp Inc.

subscriptions through your personal account (Copilot Individual) or a central organization account (Copilot Business).

CoPilot licensing

Github Copilot offers three types of licenses, Individual, Business, and Enterprise. The following table summarizes the difference between these three types of licenses:

	Individual	**Business**	**Enterprise**
Pricing	$10 per month	$19/user per month	$39/user per month
Github account	Individual	Organization or Github Enterprise	Enterprise account on Github Cloud
Copilot Chat	YES	YES	YES
Copilot CLI	YES	YES	YES
Code snippet collection	YES	NO	NO
Block suggestions matching public code	YES	YES	YES
Access via editor	YES	YES	YES
Multi-line function suggestions	YES	YES	YES
Organization-wide policy management	NO	YES	YES
Exclude specific files	NO	YES	YES
Audit logs	NO	YES	YES
Copilot chat in Github.com	NO	NO	YES
Chat skills in VS Code	NO	NO	YES
Chat skills in Visual Studio	NO	NO	YES
Chat in Github Mobile	YES	YES	YES
Pull request summaries	NO	NO	YES
Copilot knowledge base	NO	NO	YES

Copyright 2024 Mahesh Chand and CSharp Inc.

How does GitHub Copilot work?

GitHub Copilot developed by Microsoft uses LLMs trained by OpenAI on vast amount of open-source code including millions of code repositories from Github, Microsoft, and online communities.

Copilot uses machine learning to predict and suggest code based on the context of what the developer is currently writing.

Copilot can also learn from your local project in your IDE and suggest code based on the project's style and

Can GitHub Copilot help with writing documentation and comments?

Yes, GitHub Copilot can be used to comment on your code and write detailed documentation. You can even write a blog or technical article about your code. You will see how to add comments to your code and write detailed documentation in the following sections.

How accurate are the code suggestions from GitHub Copilot?

Github Copilot uses OpenAI's LLM that is trained on billions of lines of public open-source code. As a result, the training set for Copilot may contain insecure coding patterns, bugs, or references to outdated APIs or idioms. When GitHub Copilot produces suggestions based on this training data, those suggestions may also contain undesirable patterns.

The quality of the code suggestions varies depending on the volume and diversity of the training data for each language. For instance, JavaScript and C#, being widely represented in public repositories, are two of the best-supported languages by GitHub Copilot.

While Copilot provides highly relevant suggestions, you should always review and test the generated code. If you are developing code with a newer version of a framework, language, or library, Copilot may not generate code that matches your latest version.

From my personal experience, I've noticed that Copilot does not have the latest updates of some programming languages that update frequently. You will have to make sure that you do not fully rely on Copilot. This may change in the future as Copilot and LLMs evolve over time.

Copyright 2024 Mahesh Chand and CSharp Inc.

Is GitHub Copilot safe to use with proprietary or sensitive code?

GitHub Copilot is designed to prioritize user privacy and security. However, you should be cautious when using it with proprietary or sensitive code and always review suggestions for potential security risks.

If you are building a product that may be sold to others, you may want to be careful with the code generated by Copilot. While you can generate the code bit, I recommend you understand it, learn from it, and rewrite your own code.

How can I improve the quality of suggestions from GitHub Copilot?

Copilot is a generative AI tool that uses prompts to understand a question, its relevancy, and the context. More clear and descriptive with the required output you provide, the better results you will get.

You can also provide Copilot your feedback and tell it to learn from it before the next suggestion.

Here is a detailed article on how to write prompts for generative AI:

Craft Most Effective AI Prompts To Get The Most Out Of ChatGPT

Can I customize GitHub Copilot's behavior or disable it for specific files?

Yes, you can customize GitHub Copilot's behavior through settings in your code editor. You can also disable it for specific files or projects.

Does Copilot share my code with others?

While Copilot reads and processes the code you write in your Visual Studio or other editor, analyzes the context and provides code suggestions, completions, and even entire code blocks based on the patterns it has learned from publicly available code repositories.

However, Copilot does not share your code with anyone else. It operates locally within your IDE, ensuring your privacy and security.

Can I write a complete website using Copilot?

Certainly! Copilot can assist you in writing code for various parts of a website, but it’s important to understand its capabilities and limitations.

Here’s how you can use Copilot to create a website:

Frontend and UI

Copyright 2024 Mahesh Chand and CSharp Inc.

Copilot can help you write HTML and CSS code for structuring your web pages and styling them. For example, you can use Copilot to generate HTML tags, create divs, add classes, and style elements.

Copilot supports JavaScript and can help you write functions, event handlers, and other client-side code.

You can use it to create interactive features like form validation, animations, and dynamic content.

Frameworks and Libraries

Copilot can assist with popular web frameworks and libraries like React, Angular, and Vue.js.

It provides code snippets, suggests best practices, and helps you write components.

Backend Development

While Copilot doesn't directly handle backend development, it can generate code snippets for APIs, database connections, and server-side logic.

You'll need to choose a backend language (such as C#, Node.js, Python, or Java) and implement it separately.

Responsive Design

Copilot can help you create responsive layouts by suggesting media queries and responsive CSS rules.

Remember to test your website on different devices and screen sizes.

SEO and Accessibility

Copilot can guide you in writing SEO-friendly meta tags, alt attributes for images, and accessible HTML.

However, understanding SEO and accessibility principles is essential.

Deployment

Copilot won't deploy your website, but it can assist with code related to deployment scripts or configuration files.

Copilot challenges and limitations

While Copilot is your best buddy to help write your code, it has its own limitations. It is very important for you to understand these challenges and limitations.

1. Copilot is trained on public data.

Copilot uses LLMs that are trained on public data. That means, the code it generates may not be unique. Copilot data comes from Github repositories and online websites. You must keep this in mind when use Copilot.

Copilot code may also be biased towards certain styles and design and may not follow your style. However, you can train it to use your coding styles and patterns.

Copyright 2024 Mahesh Chand and CSharp Inc.

2. Copilot code may not be accurate.

Copilot heavily relies on public LLMs and hence may not be accurate. Prompt engineering is the key to getting the most out of generative AI assistants like Copilot. Writing effective generative AI prompts is a key skill you may want to learn before using Copilot.

CAUTION

Remember that while Copilot is a powerful tool, it's not a substitute for understanding web development concepts. Learning the basics of HTML, CSS, and JavaScript is crucial for building a complete and well-functioning website.

Copilot can also make mistakes and may use older version of the code. You need to make sure to review and test the code generated by Copilot.

3. Copilot code may not have an updated version.

Software programming language and frameworks get updates almost every month. The code generated by Copilot is most likely not the latest version of the language and framework since Copilot is usually trained on older data. For example, if C# version 15 is just announced and you are building your application in this version, Copilot will have no idea of the new features and obsolete features since it's probably using an older version of C# code.

You may want to update your code by yourself to have the latest features in your program.

4. Code security and performance.

Copilot does not check the code it generates for security and performance. If there is a popular public code repository that has some security flaws or performance issues, Copilot may not know that.

It is your responsibility to review the code for security and performance.

5. Intellectual property concerns.

Copilot may not follow national and international intellectual property guidelines. If you write code for your company or to sell a product, you may want to pay close attention to the intellectual property guidelines of Copilot generated code.

Copilot in Visual Studio

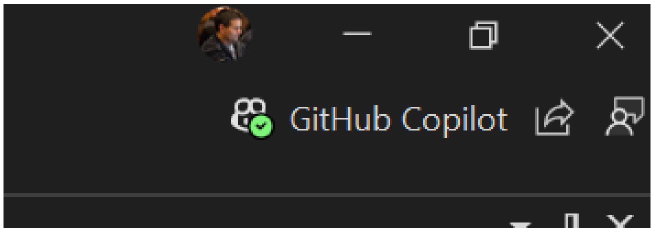
Copilot is available through Visual Studio and VS Code is an AI programming assistant that is strictly limited to software development topics only. It can assist you with coding-related questions, provide explanations, and help you with programming problems only.

Note: Before you can use Github Copilot in Visual Studio, you must have a Github Copilot license. You can buy

---

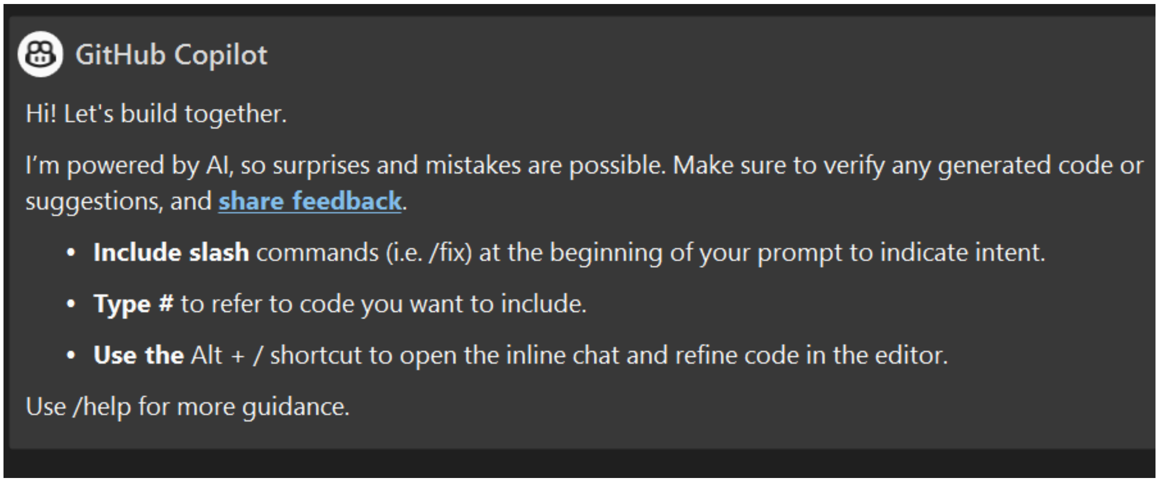
Launch CoPilot Chat

If you have Visual Studio's latest version, CoPilot is already integrated in it at the top right corner of the IDE.



Click on the Open Chat Windows will open Copilot chat where you can start asking your questions or paste your code for Copilot to review or rewrite.

The first GitHub Copilot screen you will see has the following message:



GitHub Copilot

Hi! Let's build together.

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and share feedback.

- **Include slash** commands (i.e. /fix) at the beginning of your prompt to indicate intent.
- **Type #** to refer to code you want to include.
- **Use the** Alt + / shortcut to open the inline chat and refine code in the editor.

Use /help for more guidance.

Here you will see three instructions:

1. Include slash commands at the beginning of your prompt to include intent.
2. Type # to refer to code you want to include.
3. Use the Alt + / shortcut to open the inline chat and refine code in the editor.

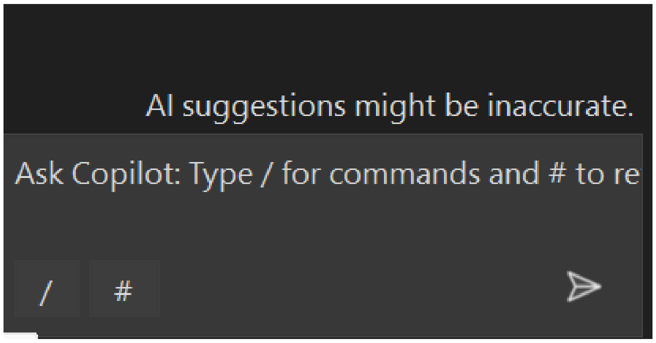
You can always use /help for more guidance.

Ask you question

We will see the above options in the coming sections but first let's get going. I want Copilot to do a few simple things for me first.

You can type your question or instruction in this chat window.

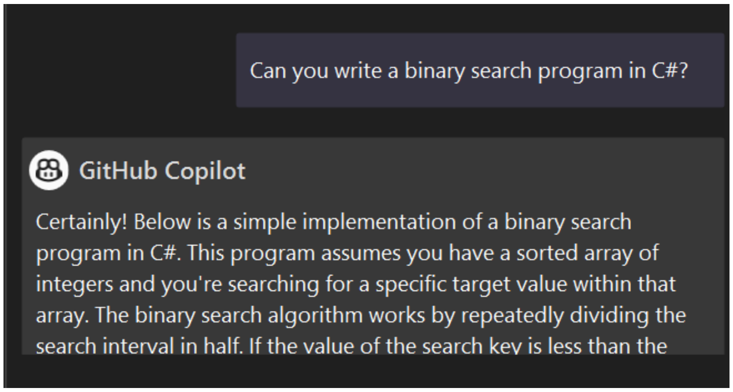
Copyright 2024 Mahesh Chand and CSharp Inc.



AI suggestions might be inaccurate.

Ask Copilot: Type / for commands and # to re

I type the following question in the chat window and this is the response I get:



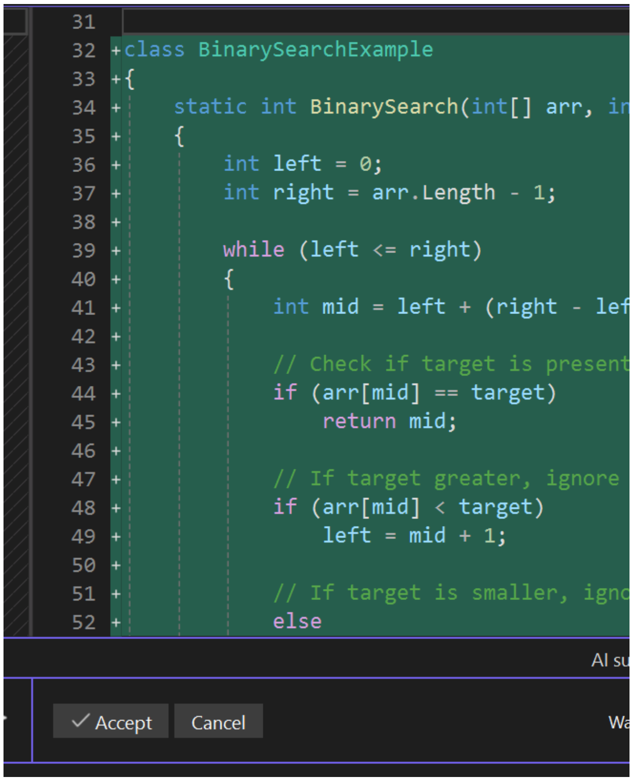
Can you write a binary search program in C#?

GitHub Copilot

Certainly! Below is a simple implementation of a binary search program in C#. This program assumes you have a sorted array of integers and you're searching for a specific target value within that array. The binary search algorithm works by repeatedly dividing the search interval in half. If the value of the search key is less than the

Below this is the complete C# program. At the bottom of the chat window, there is an option to create a new file and have a Preview.

The Preview windows allow us to Accept the new code in the current file.



```
class BinarySearchExample
{
    static int BinarySearch(int[] arr, in
    {
        int left = 0;
        int right = arr.Length - 1;

        while (left <= right)
        {
            int mid = left + (right - lef

            // Check if target is present
            if (arr[mid] == target)
                return mid;

            // If target greater, ignore
            if (arr[mid] < target)
                left = mid + 1;

            // If target is smaller, igno
            else
```

Copyright 2024 Mahesh Chand and CSharp Inc.

Note: You can also success Copilot in Visual Studio Editor using Alt + / shortcut or Right click in the Editor and select Ask Copilot menu option.

We will see this working in the following sections.

Copyright 2024 Mahesh Chand and CSharp Inc.

CHAPTER 2

WRITING BOILERPLATE CODE USING COPILOT

Almost every programmer and developer end up writing boilerplate code, especially at the beginning of a new project. Copilot is great at generating boilerplate code quickly. This is actually the most popular use of Copilot to write code for young developers and students to learn programming.

Build a simple application.

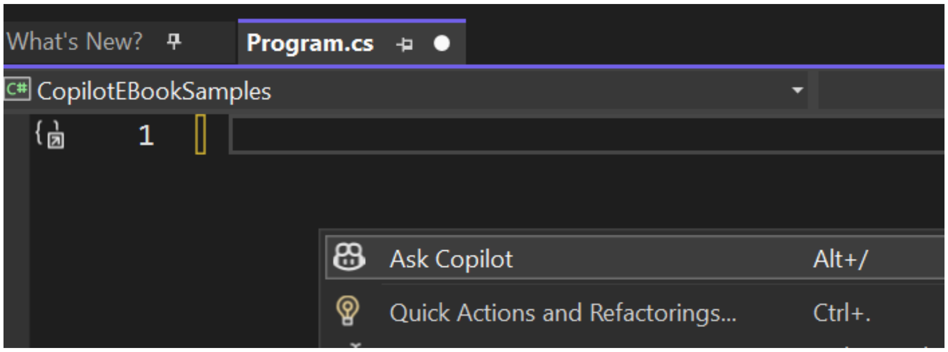
Let's create a simple yet complete C# application using Copilot without typing a single line of code.

Step 1. Create a new project.

Create a new C# Console app project in Visual Studio and delete the automatically generated code.

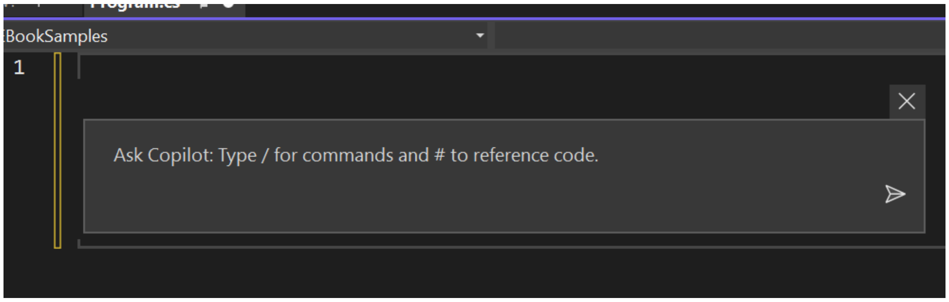
Step 2: Open Copilot chat.

Right click in the editor and select Ask Copilot or simply use Alt + /.



It will open Copilot chat window where you can ask a question or paste your existing code.

Copyright 2024 Mahesh Chand and CSharp Inc.



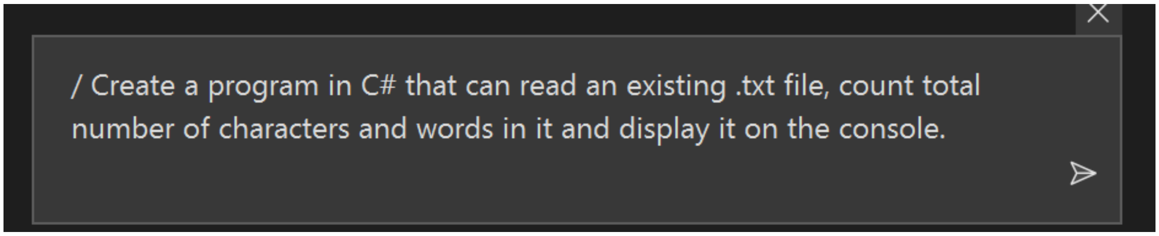
Ask Copilot: Type / for commands and # to reference code.

Note: If you ask non programming related questions, Copilot will nicely tell you that it doesn't support non programming related questions.

Step 3. Write a simple program to read a text file.

Now, let's ask Copilot to create a simple program in C# that can read a text file, count the number of characters and words in it and display the results on the console.

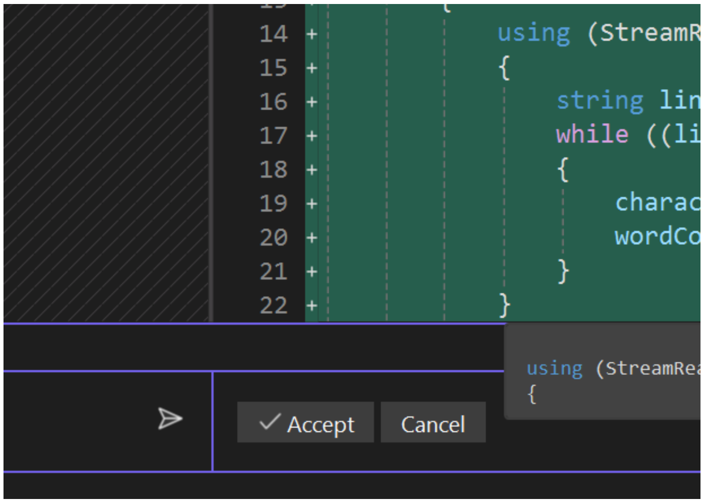
Open Copilot chat and type the following prompt and hit ENTER.



/ Create a program in C# that can read an existing .txt file, count total number of characters and words in it and display it on the console.

Wait and see the magic happens.

Copilot goes around the data it has, finds the code and writes the following code for you and ask you to review and accept it.



```
        using (StreamR
        {
            string lin
            while ((li
            {
                charac
                wordCo
            }
        }
```

```
using (StreamRea
{
```

Copyright 2024 Mahesh Chand and CSharp Inc.

Hit the Accept button.

This is what my complete program looks like:

```
using System;
using System.IO;

class Program
{
    static void Main()
    {
        string filePath = "path/to/your/file.txt";
        int characterCount = 0;
        int wordCount = 0;

        try
        {
            using (StreamReader sr = new StreamReader(filePath))
            {
                string line;
                while ((line = sr.ReadLine()) != null)
                {
                    characterCount += line.Length;
                    wordCount += line.Split(new char[] { ' ', '\t' },
StringSplitOptions.RemoveEmptyEntries).Length;
                }
            }

            Console.WriteLine("Total number of characters: " + characterCount);
            Console.WriteLine("Total number of words: " + wordCount);
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine("File not found.");
        }
        catch (Exception ex)
        {
            Console.WriteLine("An error occurred: " + ex.Message);
        }
    }
}
```

Listing 1.

Step 4. Review code for correctness.

Keep in mind, Copilot only writes code based on the data it has. You must review the code to make sure it works the way it should be and it does what it was supposed to.

To run this program, I need to change the following line to the correct text file with its full path:

`string filePath = "path/to/your/file.txt";`

I want to read an existing file named CSharpCorner.txt. I update the code with the following:

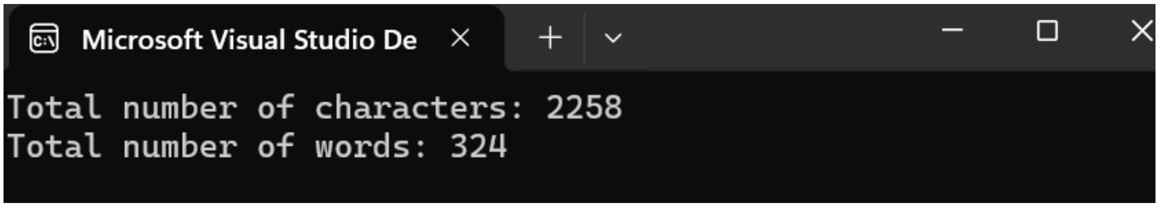
`string filePath = "C:/temp/CSharpCorner.txt";`

Copyright 2024 Mahesh Chand and CSharp Inc.

Step 5. Build and run.

Now, build and run the project.

The output looks like the following and displays results on the console:



```
Total number of characters: 2258
Total number of words: 324
```

Code review

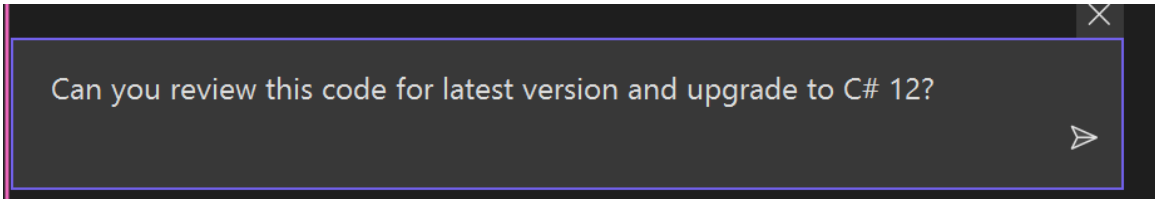
Remember, Copilot is your AI programming assistant. You're still in the driver's seat. You must ensure that the code is written by Copilot, follows your standards, quality, and works the way it is supposed to.

Copilot is also likely to make mistakes since it just writes based on what data it has. It is your responsibility to do a code review.

Copilot may also use an older version of a programming language or framework. For example, if the current version of C# is 12.0, Copilot may write code in C# 7.0 or 8.0.

You want to make sure the code is written in the latest version of C#.

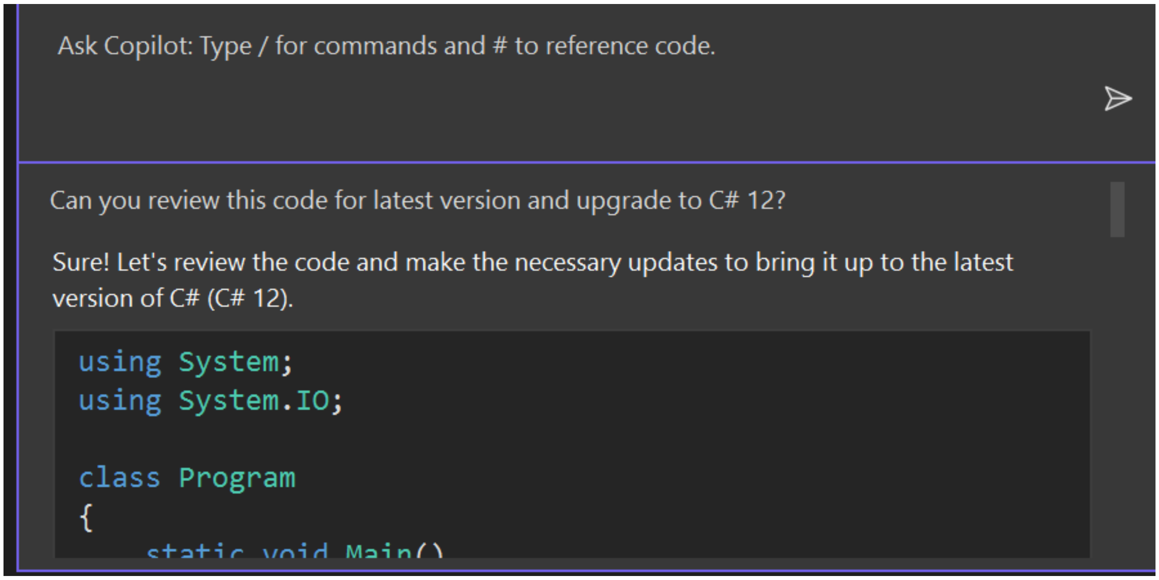
I asked Copilot to review the code and update it to C# 12.



Can you review this code for latest version and upgrade to C# 12?

Copilot replies the following:

Copyright 2024 Mahesh Chand and CSharp Inc.



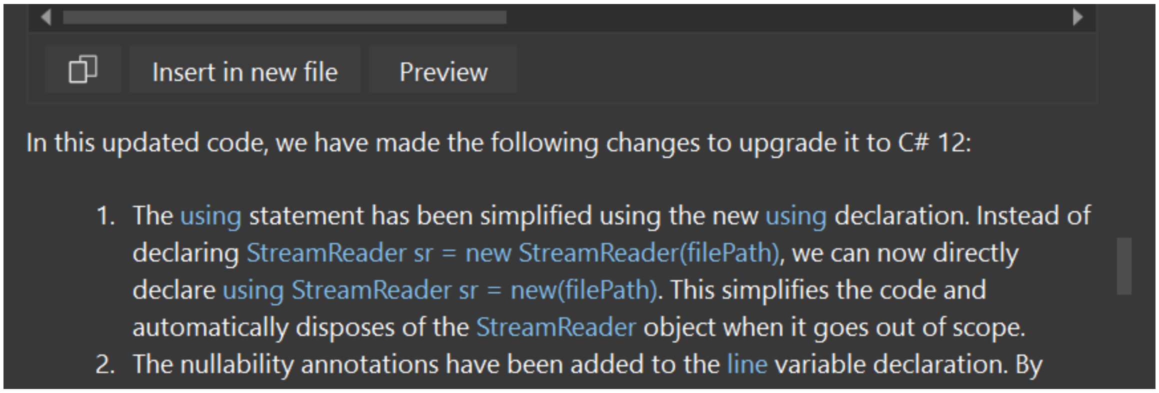
Ask Copilot: Type / for commands and # to reference code.

Can you review this code for latest version and upgrade to C# 12?

Sure! Let's review the code and make the necessary updates to bring it up to the latest version of C# (C# 12).

```
using System;
using System.IO;

class Program
{
    static void Main()
```

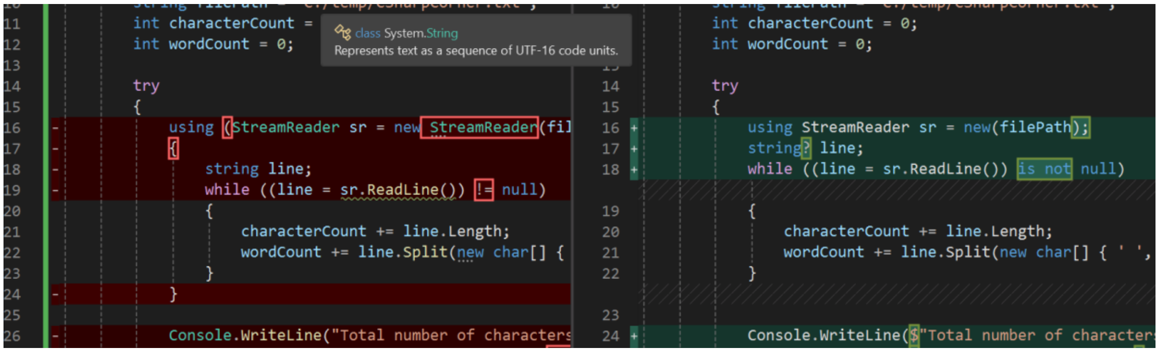
The explanation of the code is below the code:



In this updated code, we have made the following changes to upgrade it to C# 12:

1. The using statement has been simplified using the new using declaration. Instead of declaring StreamReader sr = new StreamReader(filePath), we can now directly declare using StreamReader sr = new(filePath). This simplifies the code and automatically disposes of the StreamReader object when it goes out of scope.
2. The nullability annotations have been added to the line variable declaration. By

The Preview option will show the updated code and changes line by line.



class System.String
Represents text as a sequence of UTF-16 code units.

```
        int characterCount =
        int wordCount = 0;

        try
        {
-           using (StreamReader sr = new StreamReader(fil
-           {
-               string line;
-               while ((line = sr.ReadLine()) != null)
                {
                    characterCount += line.Length;
                    wordCount += line.Split(new char[] {
                }
-           }

-           Console.WriteLine("Total number of characters
```

```
        int characterCount = 0;
        int wordCount = 0;

        try
        {
+           using StreamReader sr = new(filePath);
+           string? line;
+           while ((line = sr.ReadLine()) is not null)
            {
                characterCount += line.Length;
                wordCount += line.Split(new char[] { ' ',
            }

+           Console.WriteLine($"Total number of character
```

To accept the changes, click on the Accept button.

The changes will be reflected in your code file.

Build and run the project. You will get the same results.

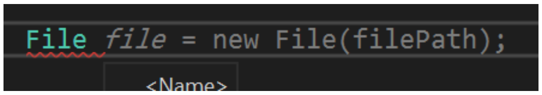
Copyright 2024 Mahesh Chand and CSharp Inc.

CHAPTER 3 AUTOCOMPLETION USING COPILOT

Code completion or extending existing code is a common feature Copilot is used for. Most of the editors are smart enough to auto complete code these days and suggest code snippet and code blocks as soon as you type something.

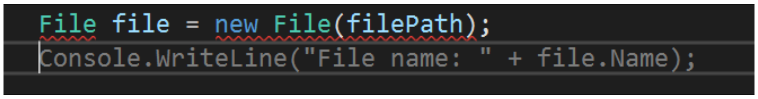
Let’s try this.

I just type File in code editor, you will see the entire line of code is already there. To accept this suggested code, all I must do is hit TAB.



```
File file = new File(filePath);
```

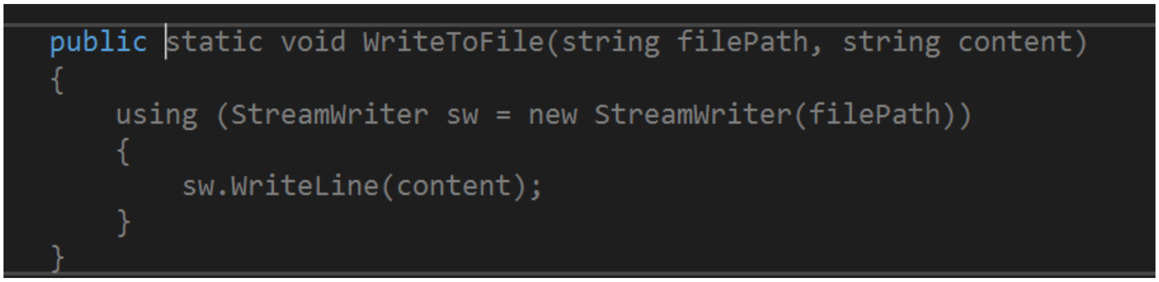
After I hit TAB key, the next line of the code is already suggested to me.



```
File file = new File(filePath);
Console.WriteLine("File name: " + file.Name);
```

Extend our application.

Let’s try to add a method to our existing class we created earlier in Listing 1. As soon as I type ‘public’, you will see Copilot suggest the following method WriteToFile.

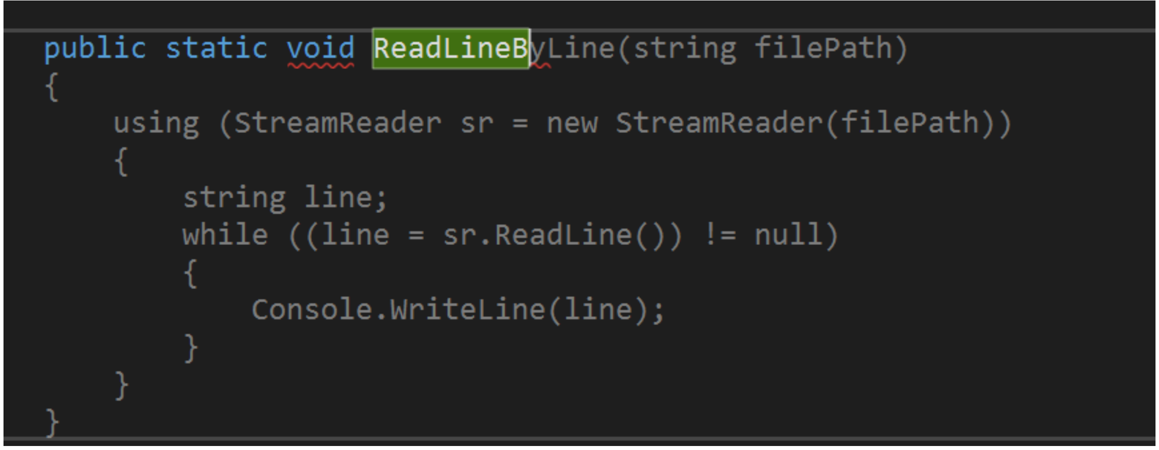


```
public static void WriteToFile(string filePath, string content)
{
    using (StreamWriter sw = new StreamWriter(filePath))
    {
        sw.WriteLine(content);
    }
}
```

But I really don’t want to write the above method. But I want to create a method that reads a text file line by line.

Copyright 2024 Mahesh Chand and CSharp Inc.

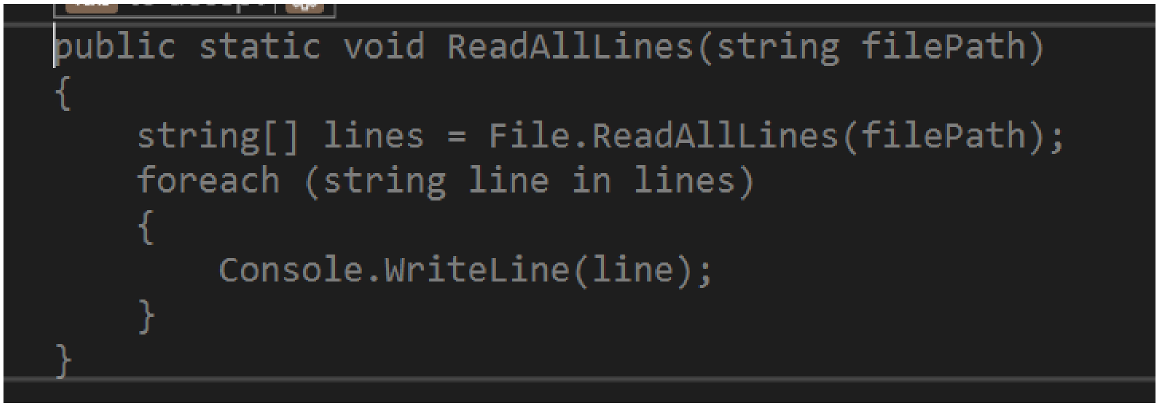
As soon as I continue the method signature and type ReadLineB.. Copilot automatically completes my intentions with the method ReadLineByLine as you can see below.



```
public static void ReadLineByLine(string filePath)
{
    using (StreamReader sr = new StreamReader(filePath))
    {
        string line;
        while ((line = sr.ReadLine()) != null)
        {
            Console.WriteLine(line);
        }
    }
}
```

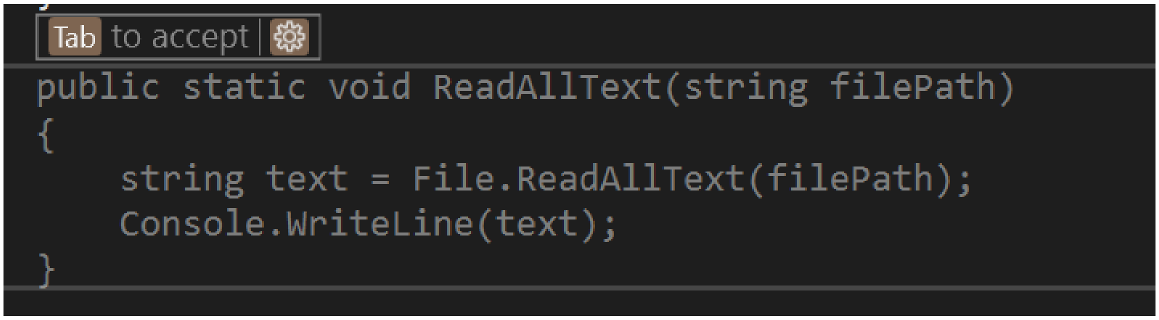
Hit TAB to accept the new method,

I hit ENTER below that and the new method suggested by Copilot is the following:



```
public static void ReadAllLines(string filePath)
{
    string[] lines = File.ReadAllLines(filePath);
    foreach (string line in lines)
    {
        Console.WriteLine(line);
    }
}
```

I hit ENTER again and the next suggested method is the following:



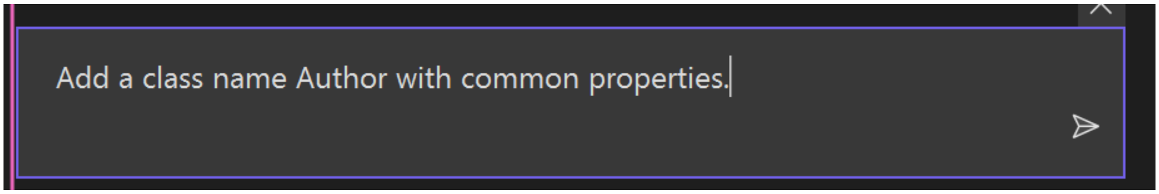
```
public static void ReadAllText(string filePath)
{
    string text = File.ReadAllText(filePath);
    Console.WriteLine(text);
}
```

In similar fashion, you may continue to use Copilot to autocomplete code.

Copyright 2024 Mahesh Chand and CSharp Inc.

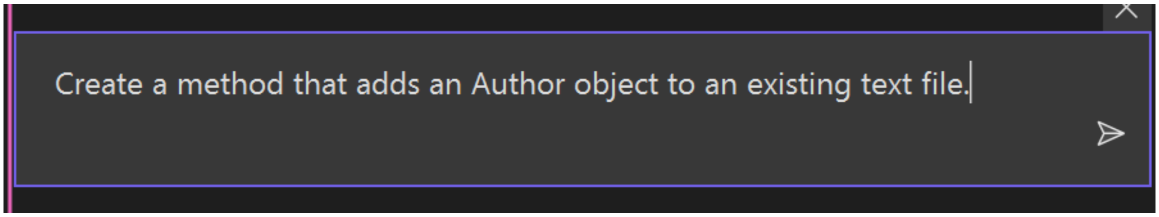
Extend existing program using Copilot

Now, let’s extend our program. We are going to add a new class in the program and then add some objects to the text file.



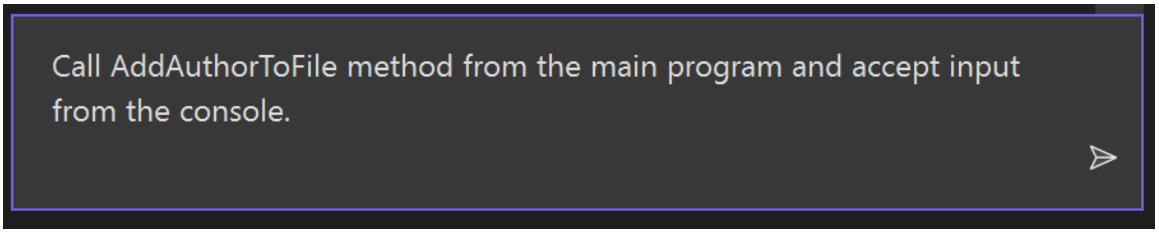
Add a class name Author with common properties.

```
class Author
{
    public string Name { get; set; }
    public string Email { get; set; }
    public string Country { get; set; }
}
```



Create a method that adds an Author object to an existing text file.

```
public static void AddAuthorToFile(Author author, string filePath)
{
    using (StreamWriter sw = new StreamWriter(filePath, true))
    {
        sw.WriteLine($"Name: {author.Name}");
        sw.WriteLine($"Email: {author.Email}");
        sw.WriteLine($"Country: {author.Country}");
        sw.WriteLine();
    }
}
```



Call AddAuthorToFile method from the main program and accept input from the console.

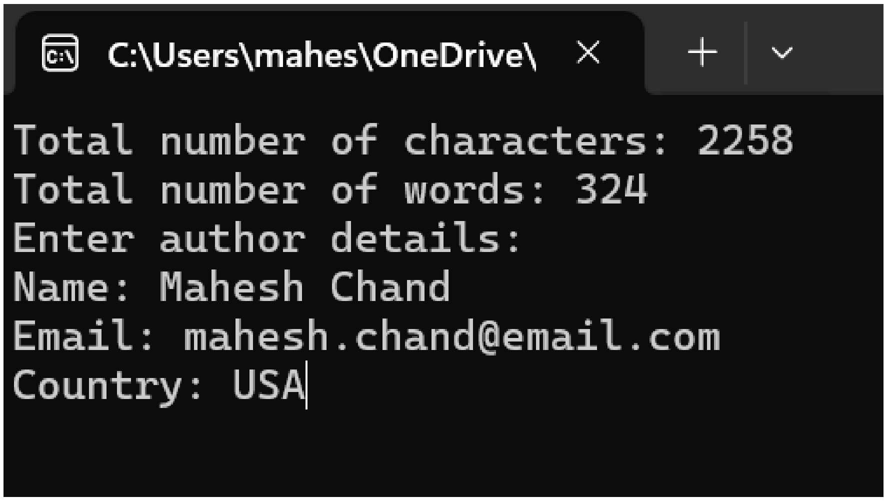
```
// Accept input from the console
Console.WriteLine("Enter author details:");
Console.Write("Name: ");
string name = Console.ReadLine();
Console.Write("Email: ");
string email = Console.ReadLine();
```

Copyright 2024 Mahesh Chand and CSharp Inc.

```
Console.Write("Country: ");
string country = Console.ReadLine();

// Create an instance of the Author class
Author author = new Author
{
    Name = name,
    Email = email,
    Country = country
};

// Call the AddAuthorToFile method
AddAuthorToFile(filePath, author);
```

Now build and run the application.

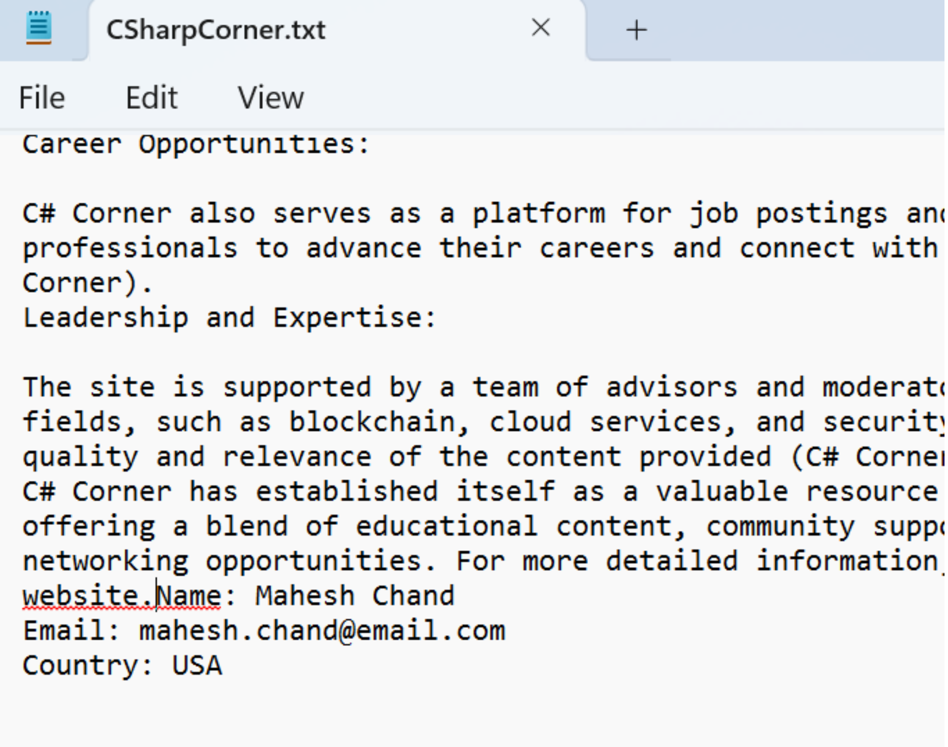
The console will ask you to enter Name, Email, and Country input values. Provide these values and hit ENTER.



```
Total number of characters: 2258
Total number of words: 324
Enter author details:
Name: Mahesh Chand
Email: mahesh.chand@email.com
Country: USA
```

Now check the .txt file and you will see the new values are added to the text file.

Copyright 2024 Mahesh Chand and CSharp Inc.



```
Career Opportunities:

C# Corner also serves as a platform for job postings an
professionals to advance their careers and connect with
Corner).
Leadership and Expertise:

The site is supported by a team of advisors and moderat
fields, such as blockchain, cloud services, and securit
quality and relevance of the content provided (C# Corne
C# Corner has established itself as a valuable resource
offering a blend of educational content, community supp
networking opportunities. For more detailed information
website.Name: Mahesh Chand
Email: mahesh.chand@email.com
Country: USA
```

Copyright 2024 Mahesh Chand and CSharp Inc.

CHAPTER 4

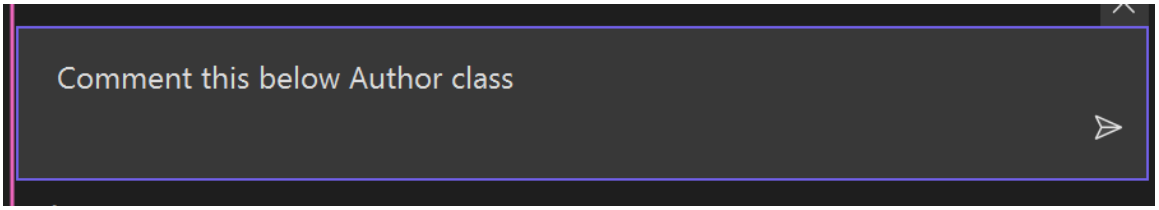
DOCUMENATION USING COPILOT

Code commenting and documentation is not something developers enjoy. Code comments are very useful to make your code readable and maintainable. Documentation plays a vital role in teams and helps new developers getting started with the project.

Commenting code using Copilot

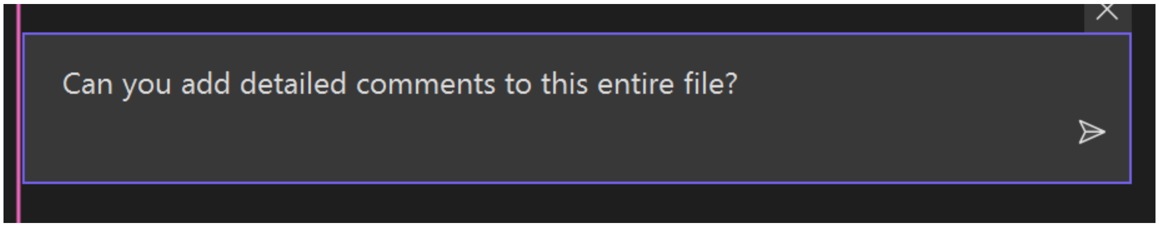
You can use Copilot to add comments to your code by simply asking Copilot.

The following prompt adds comments to the Author class.



Comment this below Author class

The following prompt comments entire program.



Can you add detailed comments to this entire file?

Now the updated program is listed in Listing 2.

```
// Represents an author with properties for name, email, and country
class Author
{
    public string Name { get; set; }
```

Copyright 2024 Mahesh Chand and CSharp Inc.

```
    public string Email { get; set; }
    public string Country { get; set; }
}


class Program
{
    static void Main()
    {
        // Specify the file path
        string filePath = "C:/temp/CSharpCorner.txt";

        // Initialize character and word count variables
        int characterCount = 0;
        int wordCount = 0;

        try
        {
            // Read the file line by line
            using (StreamReader sr = new StreamReader(filePath))
            {
                string line;
                while ((line = sr.ReadLine()) != null)
                {
                    // Update character count
                    characterCount += line.Length;

                    // Split the line into words and update word count
                    wordCount += line.Split(new char[] { ' ', '\t' },
StringSplitOptions.RemoveEmptyEntries).Length;
                }
            }

            // Display the total number of characters and words
            Console.WriteLine("Total number of characters: " +
characterCount);
            Console.WriteLine("Total number of words: " + wordCount);
        }
        catch (FileNotFoundException)
        {
            // Handle file not found exception
            Console.WriteLine("File not found.");
        }
        catch (Exception ex)
        {
            // Handle other exceptions
            Console.WriteLine("An error occurred: " + ex.Message);
        }

        // Accept input from the console
        Console.WriteLine("Enter author details:");
        Console.Write("Name: ");
        string name = Console.ReadLine();
        Console.Write("Email: ");
        string email = Console.ReadLine();
        Console.Write("Country: ");
        string country = Console.ReadLine();
```

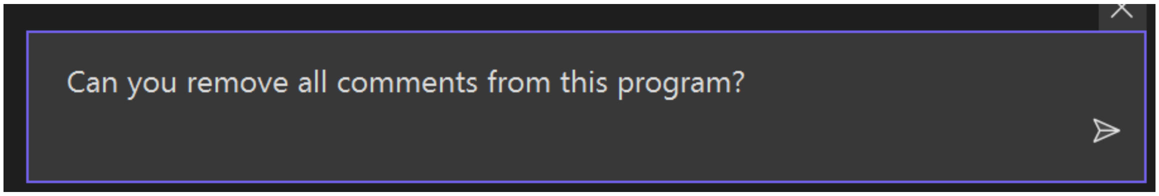
Copyright 2024 Mahesh Chand and CSharp Inc.

```
        // Create an instance of the Author class
        Author author = new Author
        {
            Name = name,
            Email = email,
            Country = country
        };

        // Call the AddAuthorToFile method to append author details to the
file
        AddAuthorToFile(filePath, author);
    }

    // Method to read the file line by line and display each line
    public static void ReadLineByLine(string filePath)
    {
        using (StreamReader sr = new StreamReader(filePath))
        {
            string line;
            while ((line = sr.ReadLine()) != null)
            {
                Console.WriteLine(line);
            }
        }
    }

    // Method to read all lines from the file and display each line
    public static void ReadAllLines(string filePath)
    {
        string[] lines = File.ReadAllLines(filePath);
        foreach (string line in lines)
        {
            Console.WriteLine(line);
        }
    }

    // Method to append author details to the file
    public static void AddAuthorToFile(string filePath, Author author)
    {
        using (StreamWriter sw = new StreamWriter(filePath, true))
        {
            sw.WriteLine($"Name: {author.Name}");
            sw.WriteLine($"Email: {author.Email}");
            sw.WriteLine($"Country: {author.Country}");
            sw.WriteLine();
        }
    }
}
```

Listing 2.

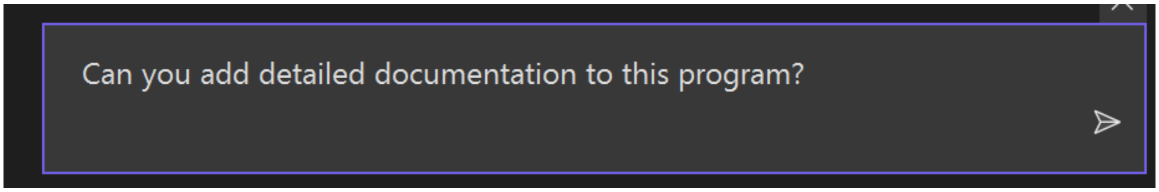
Now only you can comment your existing code but can also ask to remove existing comments.

The following prompt removes all existing comments from the program.


Can you remove all comments from this program?

Once you accept the changes, all comments from the code will be removed.

You can also add detailed documentation to the program by using the following prompt.


Can you add detailed documentation to this program?

> *You can also go to a specific class, method, and code line by typing a prompt specifically about that code block.*

The final code

```
// Represents an author with properties for name, email, and country
class Author
{
    public string Name { get; set; }
    public string Email { get; set; }
    public string Country { get; set; }
}


class Program
{
    /// <summary>
    /// Entry point of the program.
    /// </summary>
    static void Main()
    {
        // Specify the file path
        string filePath = "C:/temp/CSharpCorner.txt";

        // Initialize character and word count variables
        int characterCount = 0;
        int wordCount = 0;

        try
        {
```

Copyright 2024 Mahesh Chand and CSharp Inc.

```
            // Read the file using a StreamReader
            using (StreamReader sr = new StreamReader(filePath))
            {
                string line;
                while ((line = sr.ReadLine()) != null)
                {
                    // Update character count
                    characterCount += line.Length;

                    // Split the line into words and update word count
                    wordCount += line.Split(new char[] { ' ', '\t' },
StringSplitOptions.RemoveEmptyEntries).Length;
                }
            }

            // Print the total number of characters and words
            Console.WriteLine("Total number of characters: " +
characterCount);
            Console.WriteLine("Total number of words: " + wordCount);
        }
        catch (FileNotFoundException)
        {
            // Handle file not found exception
            Console.WriteLine("File not found.");
        }
        catch (Exception ex)
        {
            // Handle other exceptions
            Console.WriteLine("An error occurred: " + ex.Message);
        }

        // Prompt the user to enter author details
        Console.WriteLine("Enter author details:");
        Console.Write("Name: ");
        string name = Console.ReadLine();
        Console.Write("Email: ");
        string email = Console.ReadLine();
        Console.Write("Country: ");
        string country = Console.ReadLine();

        // Create an Author object with the entered details
        Author author = new Author
        {
            Name = name,
            Email = email,
            Country = country
        };

        // Add the author details to the file
        AddAuthorToFile(filePath, author);
    }

    /// <summary>
    /// Reads the file line by line and prints each line to the console.
    /// </summary>
    /// <param name="filePath">The path of the file to read.</param>
    public static void ReadLineByLine(string filePath)
    {
```

Copyright 2024 Mahesh Chand and CSharp Inc.

```
        // Read the file using a StreamReader
        using (StreamReader sr = new StreamReader(filePath))
        {
            string line;
            while ((line = sr.ReadLine()) != null)
            {
                // Print each line to the console
                Console.WriteLine(line);
            }
        }
    }

    /// <summary>
    /// Reads all lines from the file and prints each line to the console.
    /// </summary>
    /// <param name="filePath">The path of the file to read.</param>
    public static void ReadAllLines(string filePath)
    {
        // Read all lines from the file
        string[] lines = File.ReadAllLines(filePath);

        // Print each line to the console
        foreach (string line in lines)
        {
            Console.WriteLine(line);
        }
    }

    /// <summary>
    /// Adds the author details to the file.
    /// </summary>
    /// <param name="filePath">The path of the file to write.</param>
    /// <param name="author">The author object containing the
details.</param>
    public static void AddAuthorToFile(string filePath, Author author)
    {
        // Write the author details to the file using a StreamWriter
        using (StreamWriter sw = new StreamWriter(filePath, true))
        {
            sw.WriteLine($"Name: {author.Name}");
            sw.WriteLine($"Email: {author.Email}");
            sw.WriteLine($"Country: {author.Country}");
            sw.WriteLine();
        }
    }
}
```

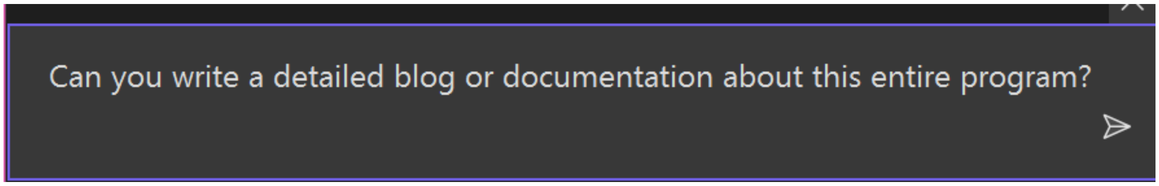
Documentation using Copilot

Copilot is very helpful in explaining code and writing a blog or documentation about existing code. This is very useful when you are a new member of an existing project and want to understand the project. Copilot can help you explain the code in no time.

Copyright 2024 Mahesh Chand and CSharp Inc.

If you write blogs and share articles, you can ask Copilot to write for you. However, I highly recommend you get the Copilot content and change it to your own style and voice.

Here is the prompt that can ask Copilot to write a blog for your program.



Can you write a detailed blog or documentation about this entire program?

And this is what I got in the reply for our existing program.

Certainly! Here's a detailed blog about the program:

Introduction

In this blog, we will explore a C# program that performs various file operations, such as reading and writing to a text file. The program also demonstrates exception handling and object-oriented programming concepts. Let's dive into the code and understand its functionality.

Program Structure

The program consists of a single class called Program. This class contains the Main method, which serves as the entry point for the program. Additionally, it includes several helper methods for file operations.

Reading File Contents

The program starts by defining a file path to a text file using the filePath variable. It then initializes two variables, characterCount and wordCount, to keep track of the number of characters and words in the file, respectively.

Next, the program attempts to read the file using a StreamReader object wrapped in a using statement. The StreamReader allows us to read the file line by line. Inside a while loop, the program reads each line and updates the characterCount by adding the length of the line. It also updates the wordCount by splitting the line into words using the Split method and counting the non-empty entries.

After reading the entire file, the program displays the total number of characters and words on the console using the Console.WriteLine method.

Exception Handling

To handle potential errors, the program includes a try-catch block. If the file specified by filePath is not found, a FileNotFoundException is thrown. The

Copyright 2024 Mahesh Chand and CSharp Inc.

program catches this exception and displays a user-friendly error message using the Console.WriteLine method.

If any other exception occurs during file reading, it is caught by the generic Exception catch block. The program then displays the error message provided by the exception using the ex.Message property.

Accepting User Input

After reading the file, the program prompts the user to enter author details. It uses the Console.ReadLine method to read the user's input for the name, email, and country. These inputs are stored in the respective variables name, email, and country.

Object-Oriented Programming

The program demonstrates object-oriented programming principles by creating an instance of the Author class. The Author class is defined separately and contains properties for the author's name, email, and country.

Using the user-provided inputs, the program initializes a new Author object and assigns the values to its properties.

Writing to File

The program calls the AddAuthorToFile method, passing the filePath and author objects as arguments. This method appends the author's details to the end of the file using a StreamWriter object wrapped in a using statement. The StreamWriter allows us to write text to a file.

Inside the using block, the program writes the author's name, email, and country to the file using the WriteLine method of the StreamWriter. It also adds an empty line for better readability.

Additional File Operations

The Program class includes two more helper methods: ReadLineByLine and ReadAllLines. These methods demonstrate different ways to read the contents of a file.

The ReadLineByLine method reads the file line by line using a StreamReader object and displays each line on the console using the Console.WriteLine method.

The ReadAllLines method reads all the lines of the file into an array of strings using the File.ReadAllLines method. It then iterates over the array and displays each line on the console.

Conclusion

In this blog, we explored a C# program that performs file operations, exception handling, and object-oriented programming concepts. We learned how to read the contents of a file, handle exceptions, accept user input, create objects, and write to a file. Understanding these concepts is crucial for building robust and interactive applications.

Copyright 2024 Mahesh Chand and CSharp Inc.

While the above content may be accurate but make sure you read through it and change it as needed before publishing it anywhere.

Copyright 2024 Mahesh Chand and CSharp Inc.

CHAPTER 5

CODE REFACTORING

Today, many software applications running worldwide are outdated and require upgrading. Software frameworks, libraries, and programming language versions evolve approximately every six months to a year. Frequent updates are often driven by security enhancements and performance optimizations.

To create better software, it's crucial to keep our code up to date whenever possible. However, updating code can be time-consuming and resource intensive. Fortunately, Copilot provides a powerful solution. With Copilot's assistance, we can efficiently refactor our code, saving valuable time and resources.

Code refactoring is the process of rewriting existing code without changing its external behavior to make it modern, maintainable, readable, fast, and secure. Refactoring is a key practice in maintaining a healthy codebase, especially in agile development environments where the codebase frequently evolves.

Here are some common practices involved in refactoring:

- Improving Code Readability: Making the code easier to understand by renaming variables, breaking large functions into smaller ones, and adding comments where necessary.
- Reducing Complexity: Simplifying complex logic, for instance by breaking complex conditions into simpler ones or using design patterns.
- Eliminating Redundant Code: Removing unnecessary or duplicate code to streamline and optimize performance.
- Enhancing Performance: Optimizing the code to improve performance, which might involve improving algorithms or data structures.
- Improving Design: Adapting the code to make future changes easier and safer, often by applying solid design principles.
- Upgrade to the latest versions: Programming languages, frameworks, and libraries get frequent updates and patches. Code refactoring helps us to bring new updates in the code and remove obsolete code.

Code refactoring is needed when you suspect some code needs to be updated or reorganized.

Using Copilot, we can refactor a code block, entire file, or even full application.

Refactor a code block

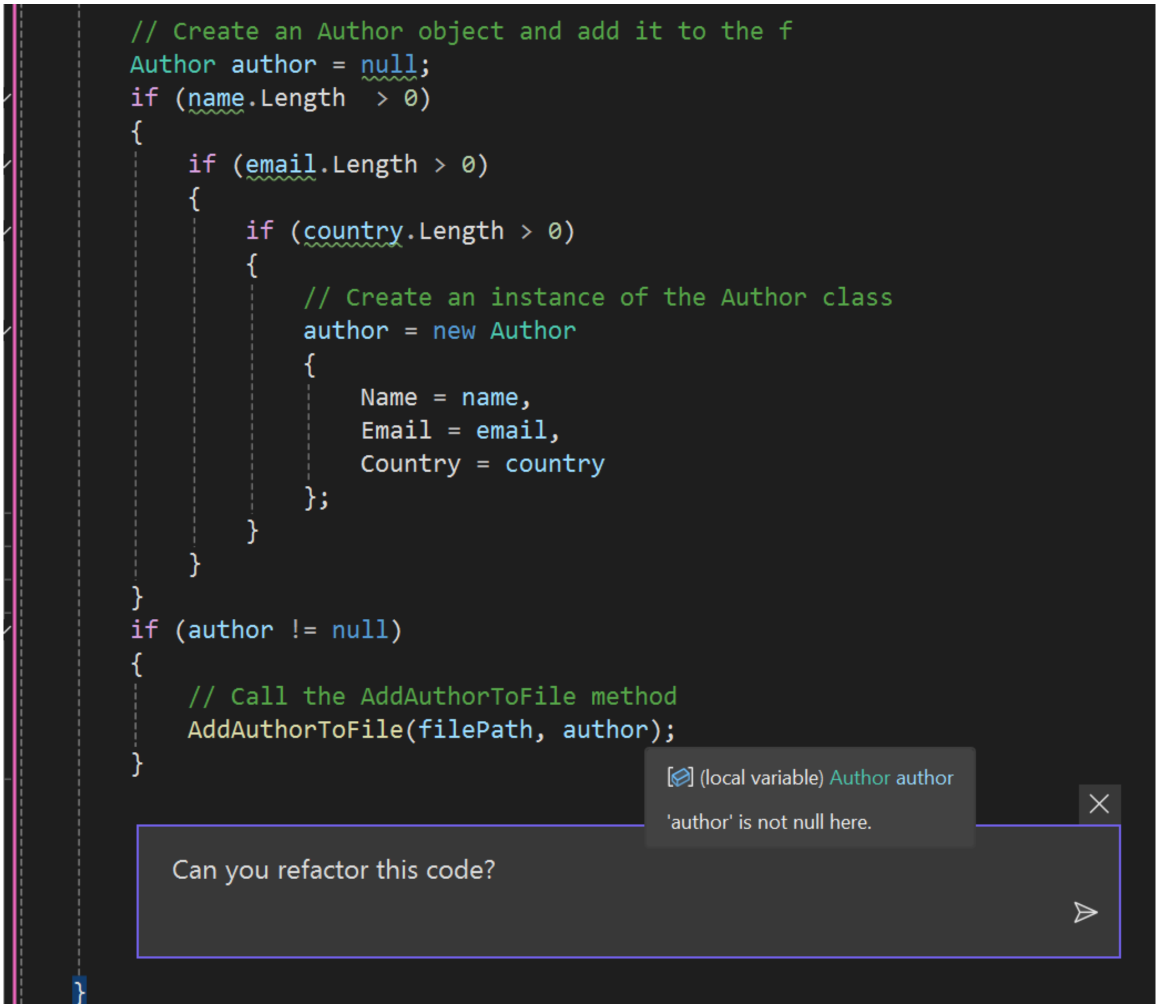
For the testing purposes, I updated the code that adds an Author object to the text file. I purposely wrote the code like a naïve developer would write it. Here is the code block:

Copyright 2024 Mahesh Chand and CSharp Inc.

```
// Create an Author object and add it to the file
 Author author = null;
 if (name.Length  > 0)
 {
     if (email.Length > 0)
     {
         if (country.Length > 0)
         {
             // Create an instance of the Author class
             author = new Author
             {
                 Name = name,
                 Email = email,
                 Country = country
             };
         }
     }
 }
 if (author != null)
 {
     // Call the AddAuthorToFile method
     AddAuthorToFile(filePath, author);

 }
```

I select a code block that I want to refactor and ask Copilot to refactor it.

Copyright 2024 Mahesh Chand and CSharp Inc.



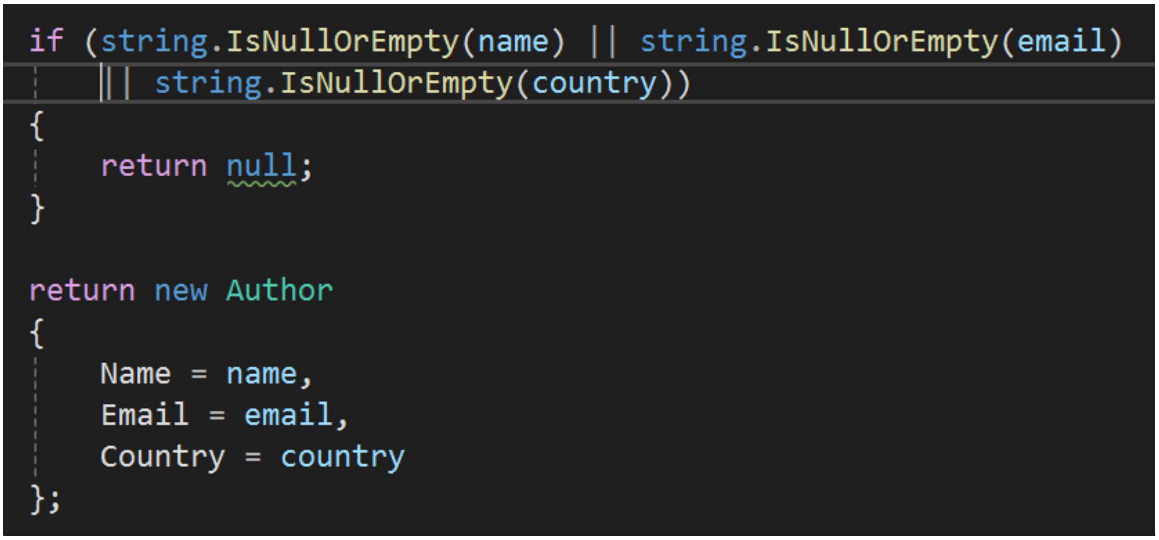
```
// Create an Author object and add it to the f
Author author = null;
if (name.Length  > 0)
{
    if (email.Length > 0)
    {
        if (country.Length > 0)
        {
            // Create an instance of the Author class
            author = new Author
            {
                Name = name,
                Email = email,
                Country = country
            };
        }
    }
}
if (author != null)
{
    // Call the AddAuthorToFile method
    AddAuthorToFile(filePath, author);
}
```

(local variable) Author author

'author' is not null here.

Can you refactor this code?

The new code generated by Copilot looks like the following:



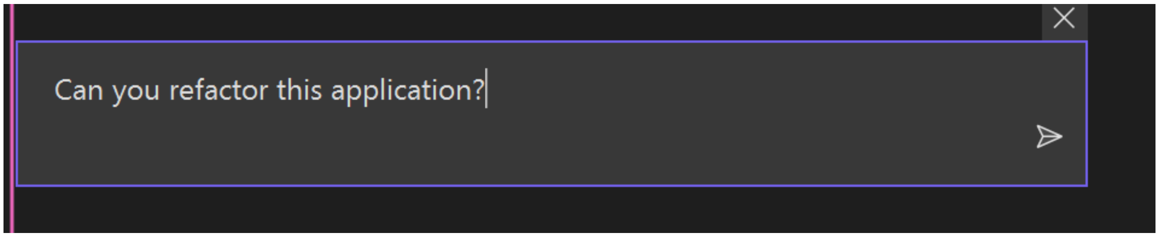
```
if (string.IsNullOrEmpty(name) || string.IsNullOrEmpty(email)
    || string.IsNullOrEmpty(country))
{
    return null;
}

return new Author
{
    Name = name,
    Email = email,
    Country = country
};
```

Not only this code is optimized but it is also readable and using updated syntaxes.

Copyright 2024 Mahesh Chand and CSharp Inc.

Refactor a program or application

Now, let’s try to refactor entire program.



Can you refactor this application?

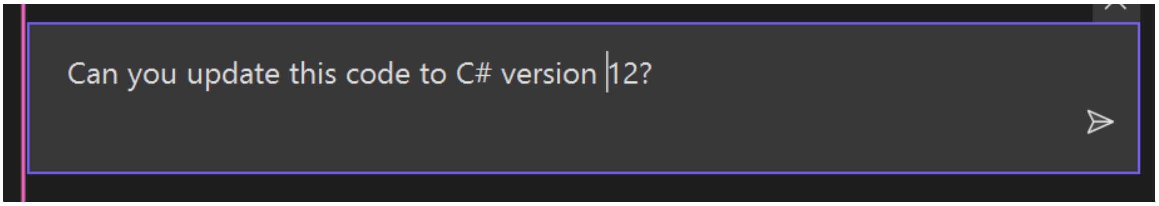
You will see the suggested changes.

Hit Accept button to accept the changes.

Update code to new version

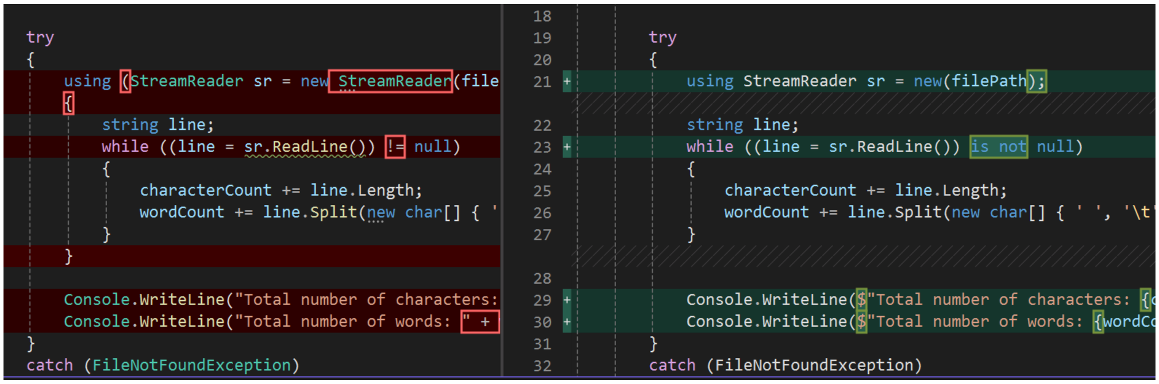
You can also use Copilot to update existing code from an older version to the new version.

The following prompts updates an existing program to C# version 12.0.



Can you update this code to C# version 12?

You will see C# version 12 updates are recommended by Copilot.



```
try
{
    using (StreamReader sr = new StreamReader(file
    {
        string line;
        while ((line = sr.ReadLine()) != null)
        {
            characterCount += line.Length;
            wordCount += line.Split(new char[] { '
        }
    }

    Console.WriteLine("Total number of characters:
    Console.WriteLine("Total number of words: " +
}
catch (FileNotFoundException)
```

```
  try
  {
+     using StreamReader sr = new(filePath);

      string line;
+     while ((line = sr.ReadLine()) is not null)
      {
          characterCount += line.Length;
          wordCount += line.Split(new char[] { ' ', '\t'
      }

+     Console.WriteLine($"Total number of characters: {c
+     Console.WriteLine($"Total number of words: {wordCo
  }
  catch (FileNotFoundException)
```

Hit Accept to accept the changes.

Now your application is written in C# 12.

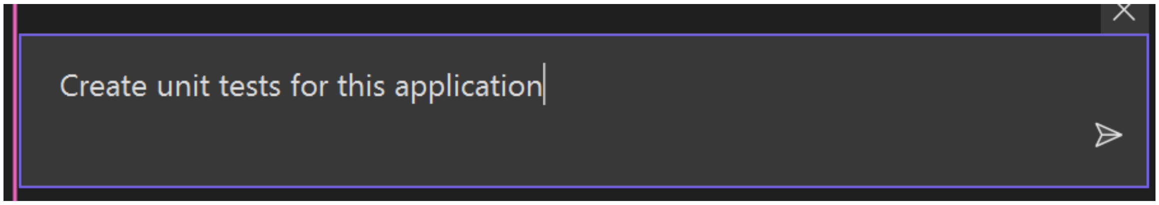
Copyright 2024 Mahesh Chand and CSharp Inc.

CHAPTER 6

WRITE TEST CASES USING COPILOT

Many programmers and developers avoid writing test cases. Copilot can be used and write unit tests in minutes.

Try the following prompt:



Create unit tests for this application

You will see a new class ProgramTests is added to the project. Accept it to add it to the program. The ProgramTests file is listed below:

```
using System;
using System.IO;
using Xunit;

public class ProgramTests
{
    [Fact]
    public void Test_GetAuthorDetailsFromConsole_ValidInput_ReturnsAuthor()
    {
        // Arrange
        var input = new StringReader("John Doe\njohndoe@example.com\nUSA\n");
        Console.SetIn(input);

        // Act
        var author = Program.GetAuthorDetailsFromConsole();

        // Assert
        Assert.NotNull(author);
        Assert.Equal("John Doe", author.Name);
        Assert.Equal("johndoe@example.com", author.Email);
        Assert.Equal("USA", author.Country);
    }

    [Fact]
    public void Test_GetAuthorDetailsFromConsole_InvalidInput_ReturnsNull()
    {
        // Arrange
        var input = new StringReader("\n\n\n");
```

Copyright 2024 Mahesh Chand and CSharp Inc.

```
        Console.SetIn(input);

        // Act
        var author = Program.GetAuthorDetailsFromConsole();

        // Assert
        Assert.Null(author);
    }

    [Fact]
    public void Test_AddAuthorToFile_ValidAuthor_WritesAuthorDetailsToFile()
    {
        // Arrange
        var filePath = "C:/temp/test.txt";
        var author = new Author
        {
            Name = "John Doe",
            Email = "johndoe@example.com",
            Country = "USA"
        };

        // Act
        Program.AddAuthorToFile(filePath, author);

        // Assert
        var lines = File.ReadAllLines(filePath);
        Assert.Equal(4, lines.Length);
        Assert.Equal("Name: John Doe", lines[0]);
        Assert.Equal("Email: johndoe@example.com", lines[1]);
        Assert.Equal("Country: USA", lines[2]);
        Assert.Equal("", lines[3]);
    }
}
```

Copilot is good in generating content, but you will still have to verify the unit test code and make sure it works as expected.

Copyright 2024 Mahesh Chand and CSharp Inc.

CHAPTER 7

FIX ERRORS USING COPILOT

Copilot can help us debug the code and provide help with the errors and exceptions in the code. Not only Copilot can explain the reason for the errors and exceptions, but it also suggests possible solutions.

If you are new to programming or copied some old code from somewhere, you can use Copilot to fix it and.

Let's see this in action.

Create a C# Console Application in Visual Studio and add the following code. Let's assume you copied this code from Google search:

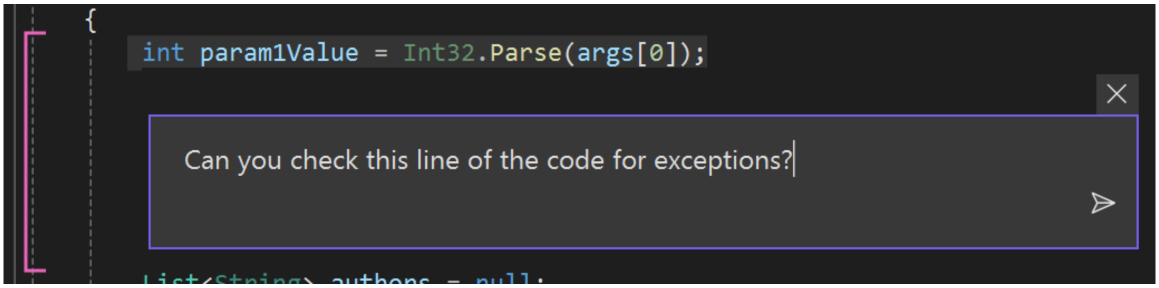
```
public class DebugWithCopilot
{
    public static void Main(string[] args)
    {
        int param1Value = Int32.Parse(args[0]);
        List<String> authors = null;
        if (param1Value > 0)
            authors = new List<String>();

        authors.Add("Mahesh Chand");

    }
}
```

Let's say, you notice something on the following line and you are not sure about this code:

`int param1Value = Int32.Parse(args[0]);`

You can ask Copilot to check it out for you.

Copyright 2024 Mahesh Chand and CSharp Inc.

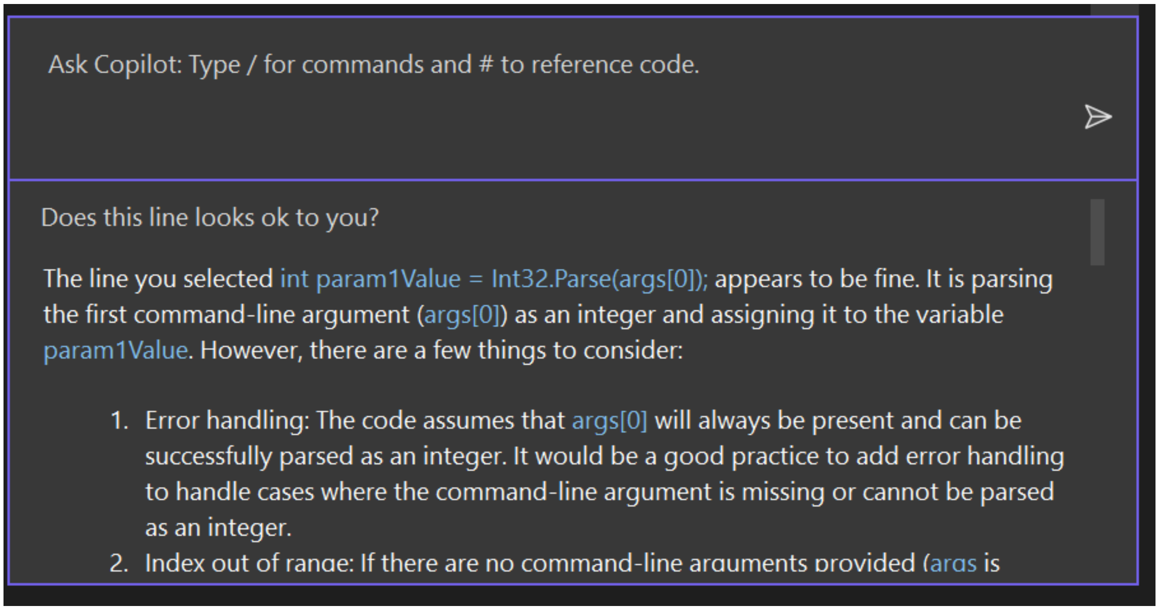


```
{
    int param1Value = Int32.Parse(args[0]);
```

Can you check this line of the code for exceptions?

```
    List<String> authors = null;
```

Copilot reviews the code and answers this:



Ask Copilot: Type / for commands and # to reference code.

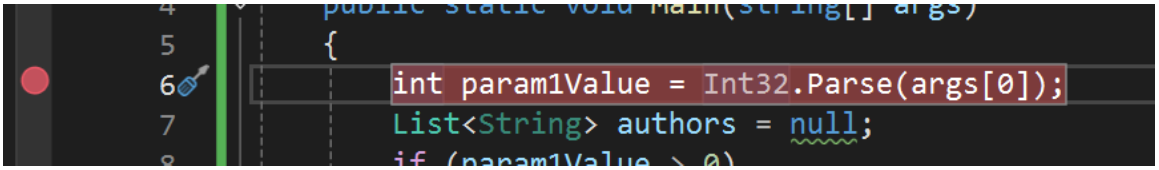
Does this line looks ok to you?

The line you selected `int param1Value = Int32.Parse(args[0]);` appears to be fine. It is parsing the first command-line argument (`args[0]`) as an integer and assigning it to the variable `param1Value`. However, there are a few things to consider:

1. Error handling: The code assumes that `args[0]` will always be present and can be successfully parsed as an integer. It would be a good practice to add error handling to handle cases where the command-line argument is missing or cannot be parsed as an integer.
2. Index out of range: If there are no command-line arguments provided (`args` is

It also suggests updated code that handles non integer values of args[0].

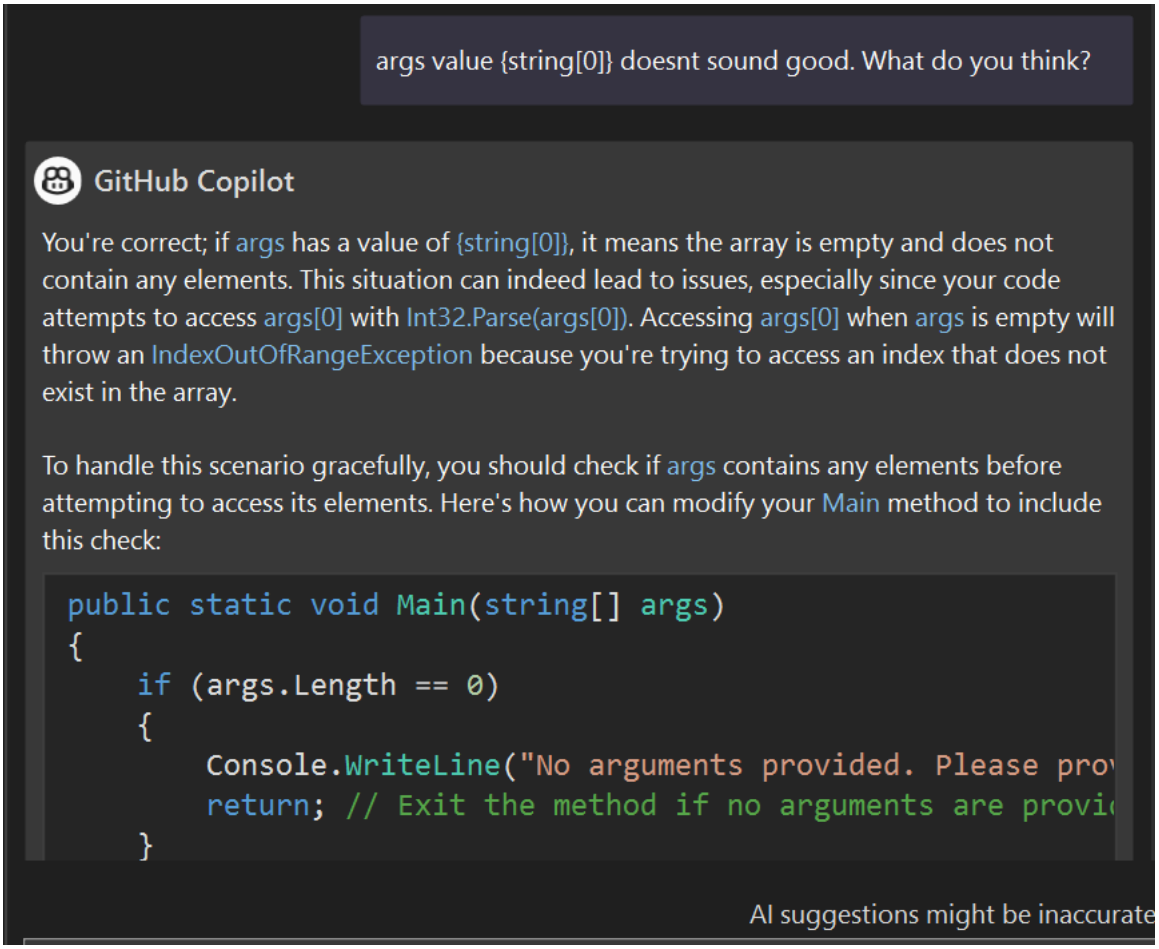
Let’s set a breakpoint at the same line:



```
    public static void Main(string[] args)
    {
        int param1Value = Int32.Parse(args[0]);
        List<String> authors = null;
        if (param1Value > 0)
```

During debugging, on the same line, I ask Copilot the following question and it answers that yes, there may be a problem with the code.

Copyright 2024 Mahesh Chand and CSharp Inc.



args value {string[0]} doesnt sound good. What do you think?

GitHub Copilot

You're correct; if args has a value of {string[0]}, it means the array is empty and does not contain any elements. This situation can indeed lead to issues, especially since your code attempts to access args[0] with Int32.Parse(args[0]). Accessing args[0] when args is empty will throw an IndexOutOfRangeException because you're trying to access an index that does not exist in the array.

To handle this scenario gracefully, you should check if args contains any elements before attempting to access its elements. Here's how you can modify your Main method to include this check:

```
public static void Main(string[] args)
{
    if (args.Length == 0)
    {
        Console.WriteLine("No arguments provided. Please prov
        return; // Exit the method if no arguments are provi
    }
```

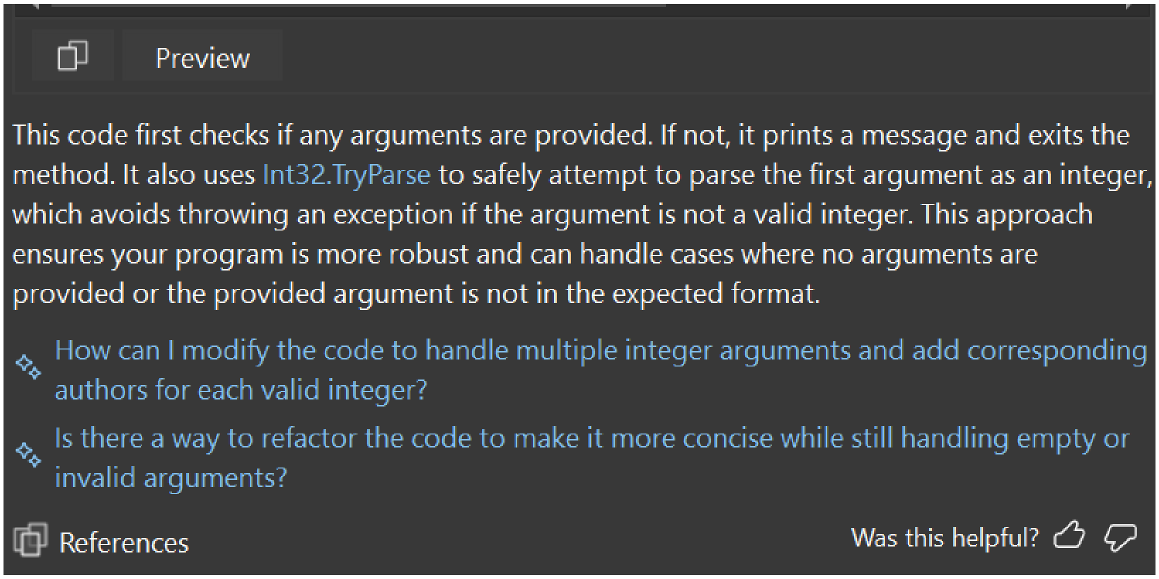
AI suggestions might be inaccurate

It also suggested the updated code.

Here is the updated code:

```
public static void Main(string[] args)
{
    if (args.Length == 0)
    {
        Console.WriteLine("No arguments provided. Please provide an
argument.");
        return; // Exit the method if no arguments are provided
    }

    int param1Value;
    // Attempt to parse the first argument as an integer
    if (!Int32.TryParse(args[0], out param1Value))
    {
        Console.WriteLine("The provided argument is not a valid integer.");
        return; // Exit the method if the argument is not a valid integer
    }

    List<String> authors = null;
    if (param1Value > 0)
    {
        authors = new List<String>();
        authors.Add("Mahesh Chand");
    }
    // Ensure authors is not null before attempting to use it
    if (authors != null)
    {
        // Safe to use authors here
    }
}
```

Copilot also explains the new code and also suggests more questions related to the code and suggests to modify further to handle other possible exceptions and also suggest to add more related functionality to the program.



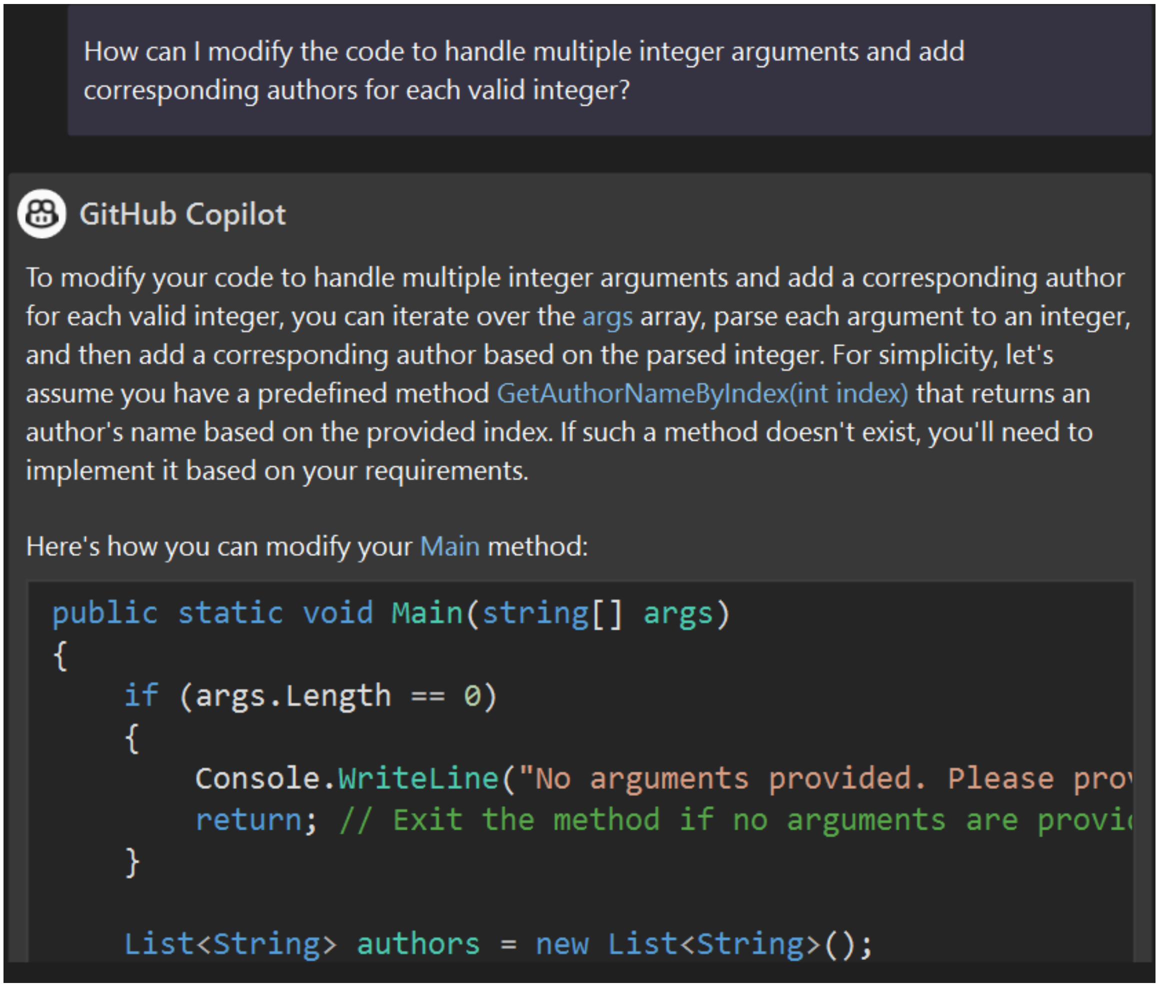
This code first checks if any arguments are provided. If not, it prints a message and exits the method. It also uses Int32.TryParse to safely attempt to parse the first argument as an integer, which avoids throwing an exception if the argument is not a valid integer. This approach ensures your program is more robust and can handle cases where no arguments are provided or the provided argument is not in the expected format.

How can I modify the code to handle multiple integer arguments and add corresponding authors for each valid integer?

Is there a way to refactor the code to make it more concise while still handling empty or invalid arguments?

I select the first suggestion.

Copyright 2024 Mahesh Chand and CSharp Inc.



How can I modify the code to handle multiple integer arguments and add corresponding authors for each valid integer?

To modify your code to handle multiple integer arguments and add a corresponding author for each valid integer, you can iterate over the `args` array, parse each argument to an integer, and then add a corresponding author based on the parsed integer. For simplicity, let's assume you have a predefined method `GetAuthorNameByIndex(int index)` that returns an author's name based on the provided index. If such a method doesn't exist, you'll need to implement it based on your requirements.

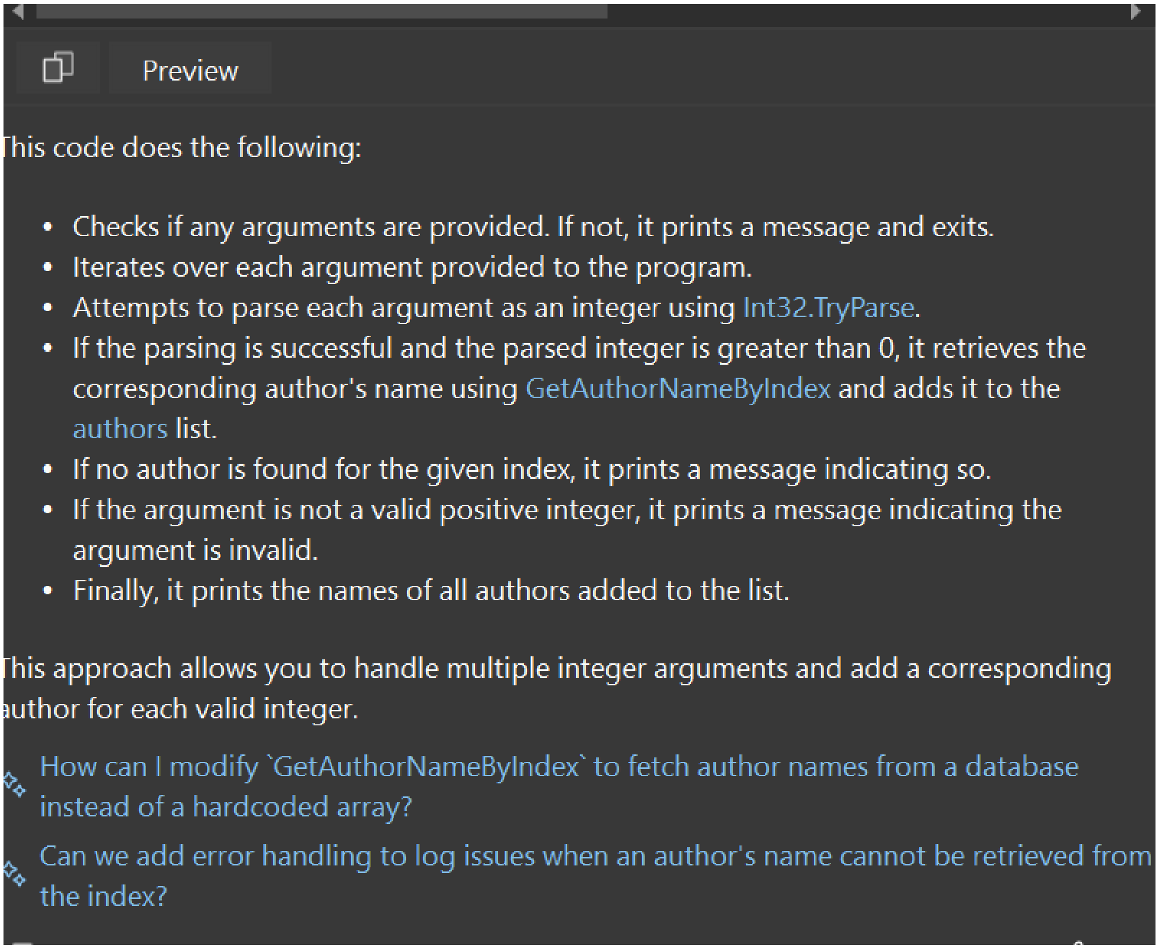
Here's how you can modify your `Main` method:

```
public static void Main(string[] args)
{
    if (args.Length == 0)
    {
        Console.WriteLine("No arguments provided. Please prov
        return; // Exit the method if no arguments are provi
    }

    List<String> authors = new List<String>();
```

It explains and writes the updated code.

And as obvious, explains the newly updated code in details.

Copyright 2024 Mahesh Chand and CSharp Inc.



This code does the following:

- Checks if any arguments are provided. If not, it prints a message and exits.
- Iterates over each argument provided to the program.
- Attempts to parse each argument as an integer using Int32.TryParse.
- If the parsing is successful and the parsed integer is greater than 0, it retrieves the corresponding author's name using GetAuthorNameByIndex and adds it to the authors list.
- If no author is found for the given index, it prints a message indicating so.
- If the argument is not a valid positive integer, it prints a message indicating the argument is invalid.
- Finally, it prints the names of all authors added to the list.

This approach allows you to handle multiple integer arguments and add a corresponding author for each valid integer.

How can I modify `GetAuthorNameByIndex` to fetch author names from a database instead of a hardcoded array?

Can we add error handling to log issues when an author's name cannot be retrieved from the index?

And of course, suggest more updates in the existing code including adding a log to capture error errors in the code.

If you click on the first link, How can I modify ‘GetAuthorNameByIndex’ to fetch author names from a database instead of a hardcoded array?, Copilot writes full ADO.NET code for you that just need the connection string updated to your database.

We will see how to build a complete data-driven application using Copilot in the following section of this book.

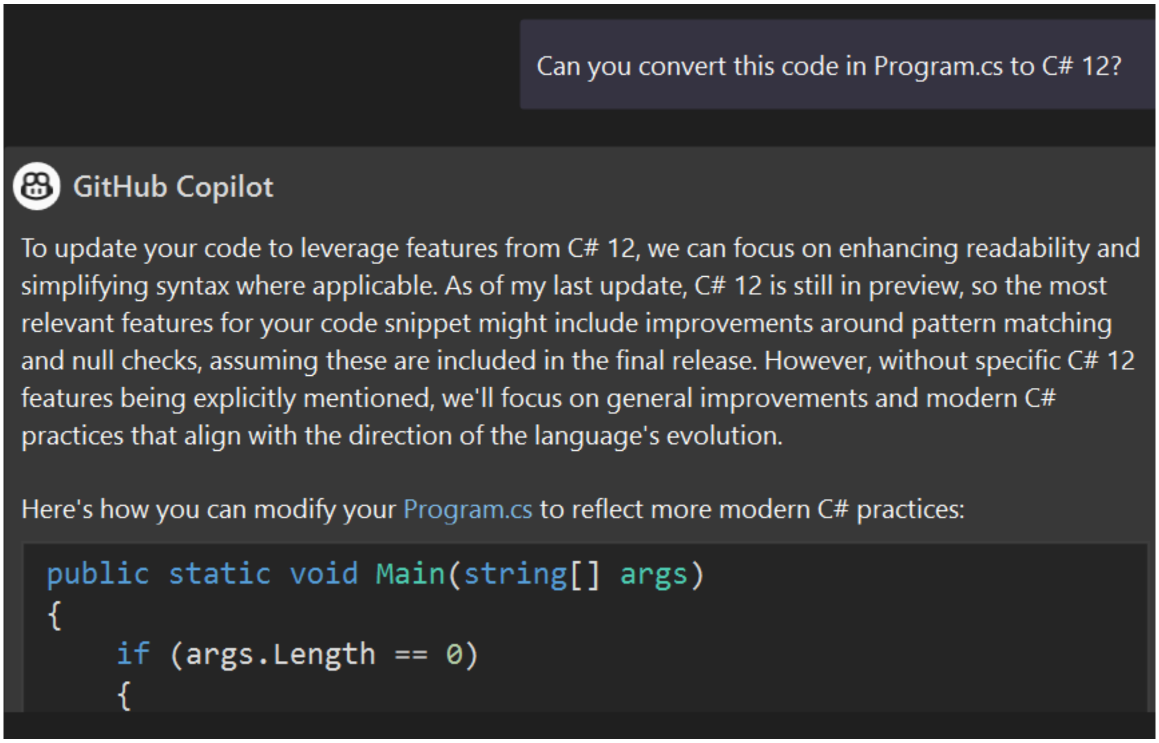
Code Review. Again!

The first rule of building better software is never trusting your code no matter if it’s generated by Copilot. Since Copilot uses publicly available historical open-source code samples, it’s possible that the code written by Copilot is not up to date.

So, I ask Copilot to update my code to C# 12, the latest version of the language.

While Copilot updates the code but still thinks that C# 12 is in Preview. However, in reality, C# 12 has already been released.

Copyright 2024 Mahesh Chand and CSharp Inc.



Can you convert this code in Program.cs to C# 12?

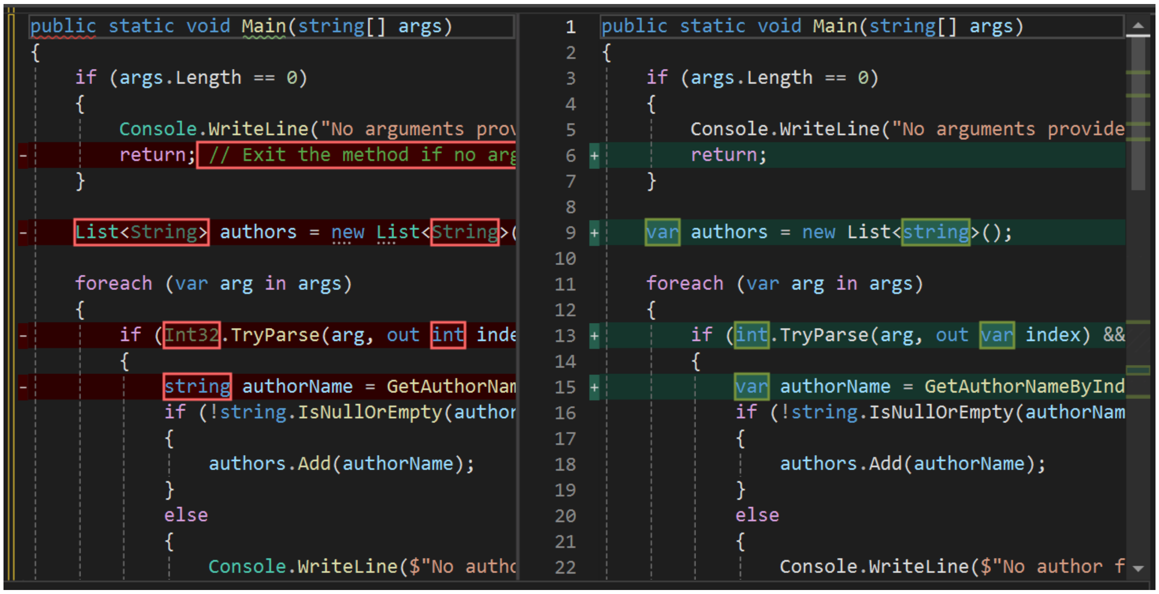
GitHub Copilot

To update your code to leverage features from C# 12, we can focus on enhancing readability and simplifying syntax where applicable. As of my last update, C# 12 is still in preview, so the most relevant features for your code snippet might include improvements around pattern matching and null checks, assuming these are included in the final release. However, without specific C# 12 features being explicitly mentioned, we'll focus on general improvements and modern C# practices that align with the direction of the language's evolution.

Here's how you can modify your Program.cs to reflect more modern C# practices:

```
public static void Main(string[] args)
{
    if (args.Length == 0)
    {
```

Copilot suggests several changes in the code.



```
  public static void Main(string[] args)
  {
      if (args.Length == 0)
      {
          Console.WriteLine("No arguments prov
-         return; // Exit the method if no arg
      }

-     List<String> authors = new List<String>(

      foreach (var arg in args)
      {
-         if (Int32.TryParse(arg, out int inde
          {
-             string authorName = GetAuthorNam
              if (!string.IsNullOrEmpty(author
              {
                  authors.Add(authorName);
              }
              else
              {
                  Console.WriteLine($"No autho
```

```
  public static void Main(string[] args)
  {
      if (args.Length == 0)
      {
          Console.WriteLine("No arguments provide
+         return;
      }

+     var authors = new List<string>();

      foreach (var arg in args)
      {
+         if (int.TryParse(arg, out var index) &&
          {
+             var authorName = GetAuthorNameByInd
              if (!string.IsNullOrEmpty(authorNam
              {
                  authors.Add(authorName);
              }
              else
              {
                  Console.WriteLine($"No author f
```

I accept the changes.

Here is the explanation by Copilot about the updated code:

Key changes and considerations:

1. Use of var: The use of var for local variable declaration is encouraged for readability, especially when the right-hand side of the declaration makes the type obvious.

Copyright 2024 Mahesh Chand and CSharp Inc.

2. Nullable Reference Types: Marking the return type of GetAuthorNameByIndex as string? explicitly indicates that the method can return null. This is part of C#'s nullable reference types feature, which helps developers write more null-safe code.
3. Simplified Array Initialization: The array of authorNames is initialized using the new[] syntax, which is concise and clear.
4. Pattern Matching: While the provided code snippet doesn't directly benefit from C# 9's pattern matching enhancements, keeping an eye on pattern matching improvements in C# 12 and beyond can help simplify certain conditional checks.
5. Expression-bodied Members: For methods like GetAuthorNameByIndex that consist of a single expression, consider using expression-bodied members for conciseness. This has been available since C# 7 for methods.
6. Top-level Statements: If your application's entry point (Main method) is straightforward, you could consider using top-level statements to reduce boilerplate code. However, since your Main method contains significant logic, it's appropriate to keep it as is.

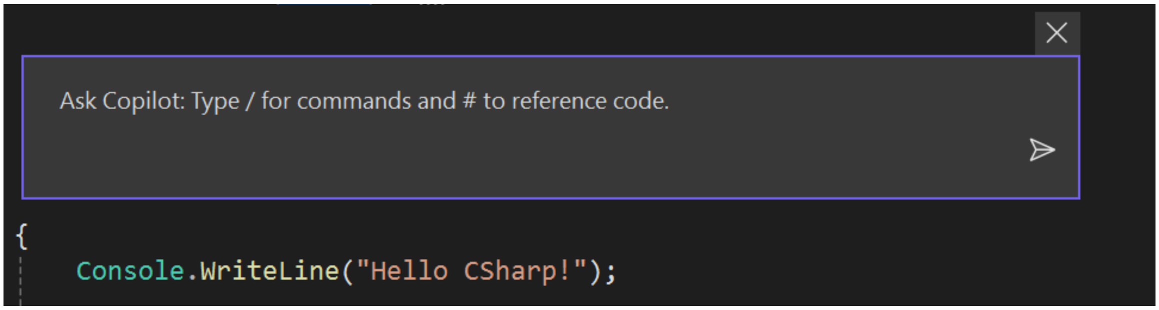
Remember, the adoption of new language features should be balanced with the readability and maintainability of your code, especially in team environments where not all members may be familiar with the latest features.

Copyright 2024 Mahesh Chand and CSharp Inc.

CHAPER 8

COPILOT SLASH COMMANDS

In Visual Studio, ALT + / launches the CoPilot.



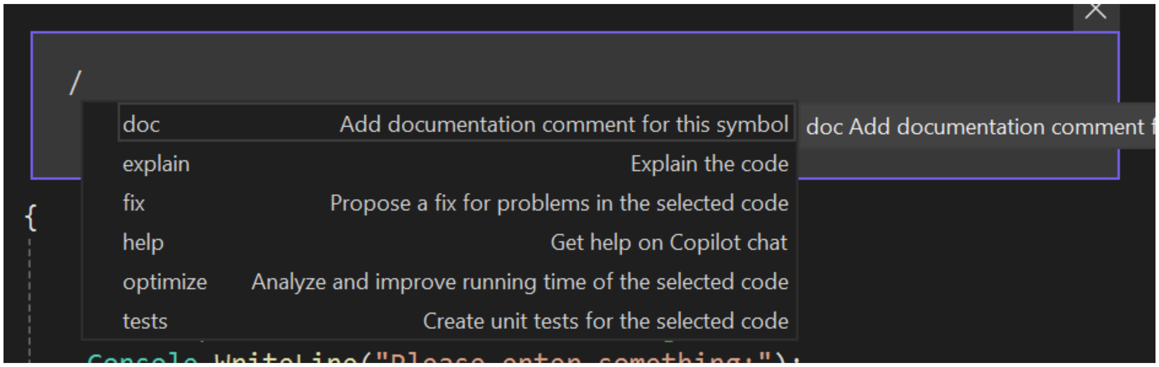
```
{
    Console.WriteLine("Hello CSharp!");
```

In this Window, you can type the Copilot command in plain English, or you can use slash commands. Slash commands are a shorter version to ask Copilot to generate content for you.

What are Copilot slash commands?

Slash commands are short cuts to give instructions to Copilot to create and review content.

You can see all slash commands by typing slash (/) in the Copilot window.


/
doc Add documentation comment for this symbol
explain Explain the code
fix Propose a fix for problems in the selected code
help Get help on Copilot chat
optimize Analyze and improve running time of the selected code
tests Create unit tests for the selected code
doc Add documentation comment

```
{
```

/doc: Add a documentation comment.

/explain: Explain the code.

/fix: Propose a fix for problems in the selected code.

Copyright 2024 Mahesh Chand and CSharp Inc.

/help: Get help on Copilot chat.

/generate: Generate code to answer your question.

/tests: Create unit tests for the selected code.

Slash commands explained

Let’s look at the slash command in more detail and how to use them.

/generate: Generate code to answer your question.

Suppose you’re working on a Python project and need to read data from a CSV file. You can use the /generate command to get relevant code snippets. Here’s how:

Type a question or a description of what you want to achieve. For instance:

How do I read a CSV file in Python?

Add /generate followed by relevant keywords:

/generate read CSV file Python

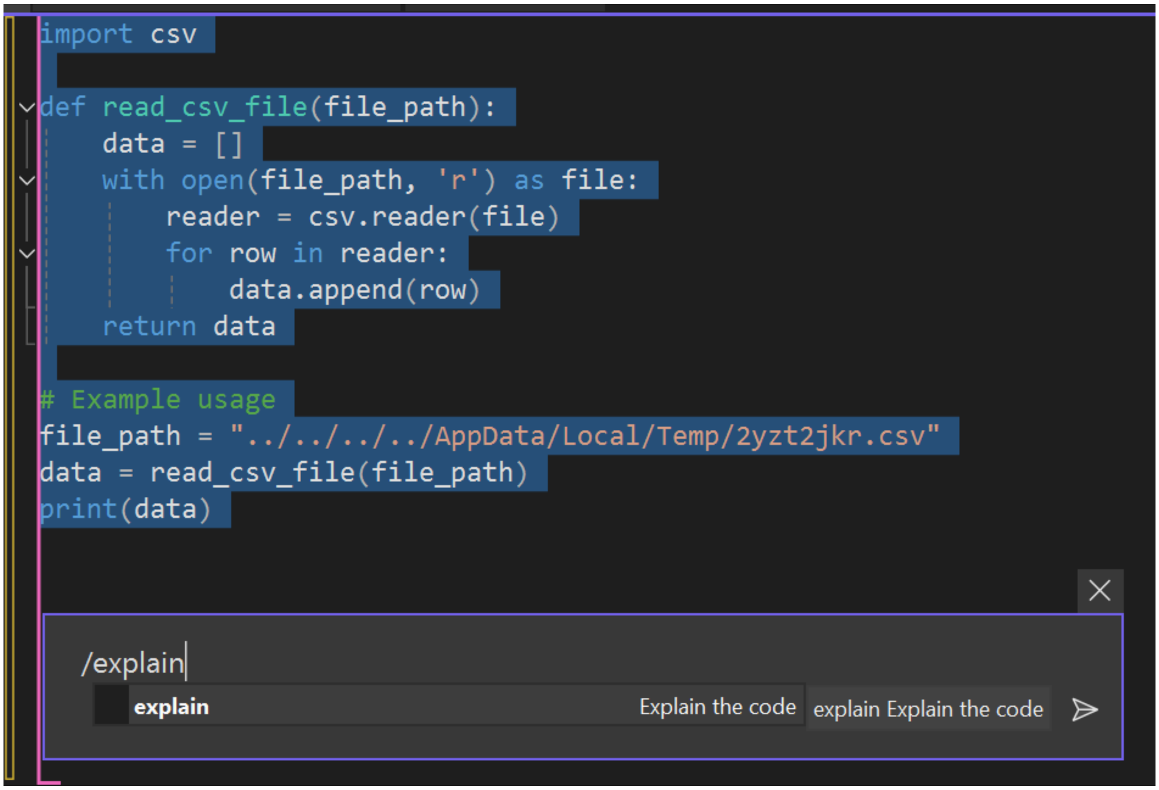
Copilot will provide code snippets related to reading CSV files in Python. It also creates a new file with .py extension. The code generated by Copilot looks like this:

```python
import csv

def read_csv_file(file_path):
    data = []
    with open(file_path, 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            data.append(row)
    return data

# Example usage
file_path = "../../../../AppData/Local/Temp/2yzt2jkr.csv"
data = read_csv_file(file_path)

print(data)
```

/explain: Explain the selected code.

You can use the /explain code which is like writing a documentation. You may wan to use the same command to write a detailed blog about your code.

Copyright 2024 Mahesh Chand and CSharp Inc.

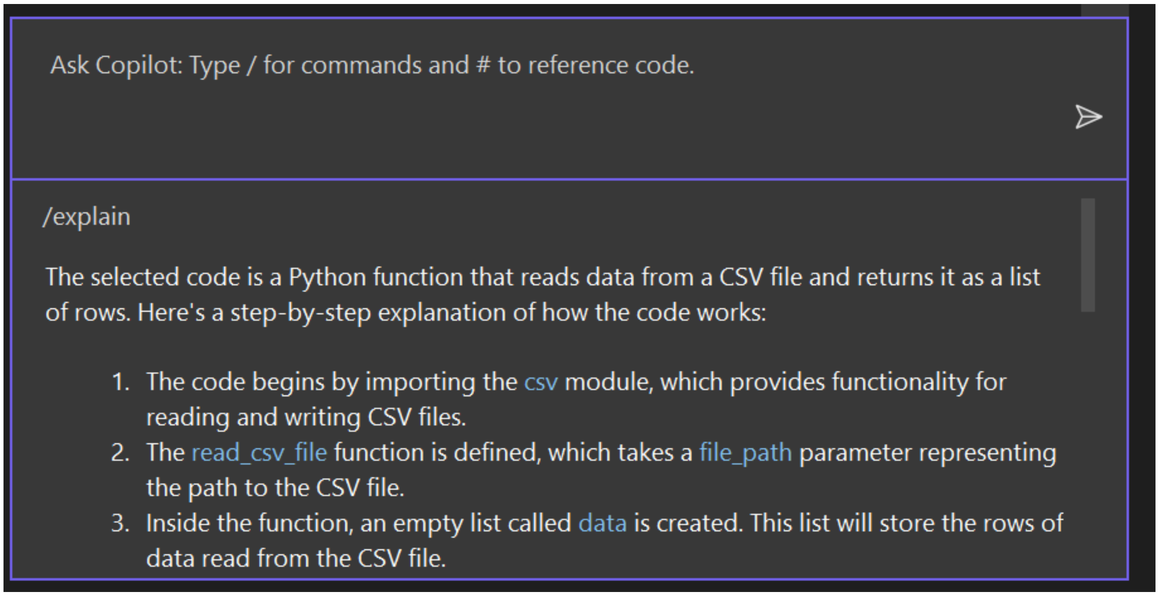


```
import csv

def read_csv_file(file_path):
    data = []
    with open(file_path, 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            data.append(row)
    return data

# Example usage
file_path = "../../../../AppData/Local/Temp/2yzt2jkr.csv"
data = read_csv_file(file_path)
print(data)
```

/explain

Hit ENTER.

Copilot writes detailed explanation of the selected code.



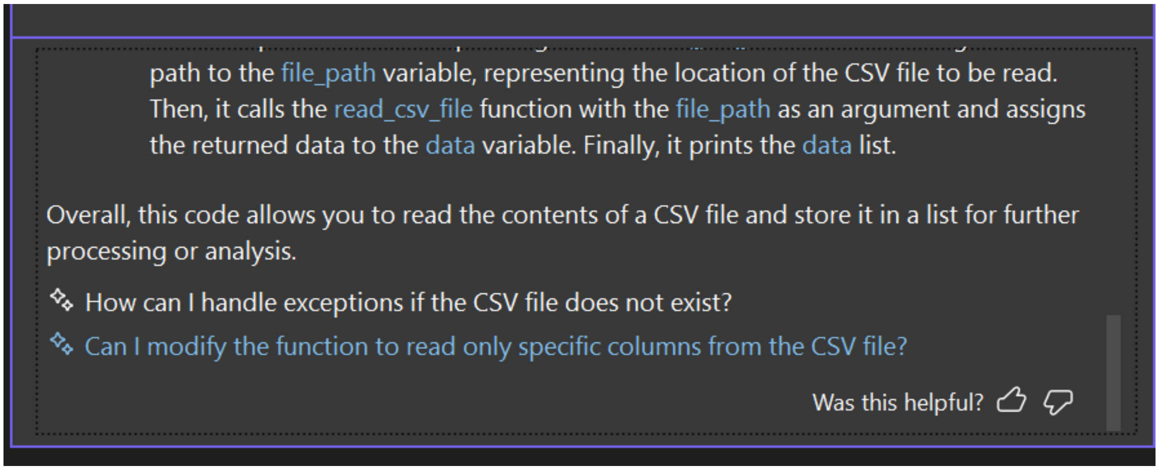
/explain

The selected code is a Python function that reads data from a CSV file and returns it as a list of rows. Here's a step-by-step explanation of how the code works:

1. The code begins by importing the csv module, which provides functionality for reading and writing CSV files.
2. The read_csv_file function is defined, which takes a file_path parameter representing the path to the CSV file.
3. Inside the function, an empty list called data is created. This list will store the rows of data read from the CSV file.

Copilot also suggest the next steps you want to add in this code:

Copyright 2024 Mahesh Chand and CSharp Inc.



path to the file_path variable, representing the location of the CSV file to be read. Then, it calls the read_csv_file function with the file_path as an argument and assigns the returned data to the data variable. Finally, it prints the data list.

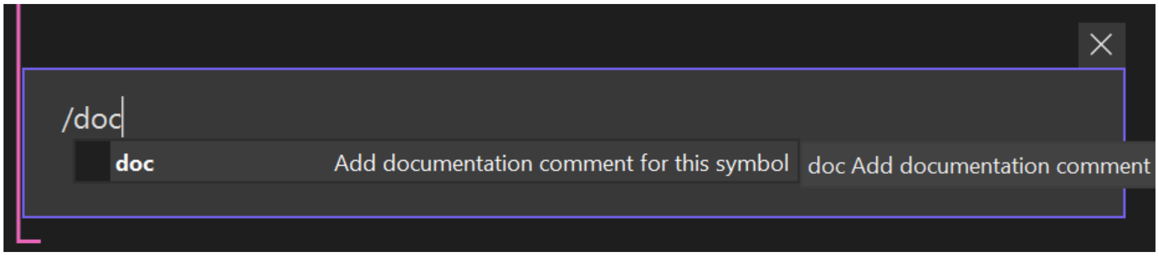
Overall, this code allows you to read the contents of a CSV file and store it in a list for further processing or analysis.

How can I handle exceptions if the CSV file does not exist?
Can I modify the function to read only specific columns from the CSV file?
Was this helpful?

/doc: Add a documentation comment.

This command add comments the code.

Select the code above and type /doc command in the Copilot command window and Hit ENTER.



/doc
doc
Add documentation comment for this symbol
doc Add documentation comment

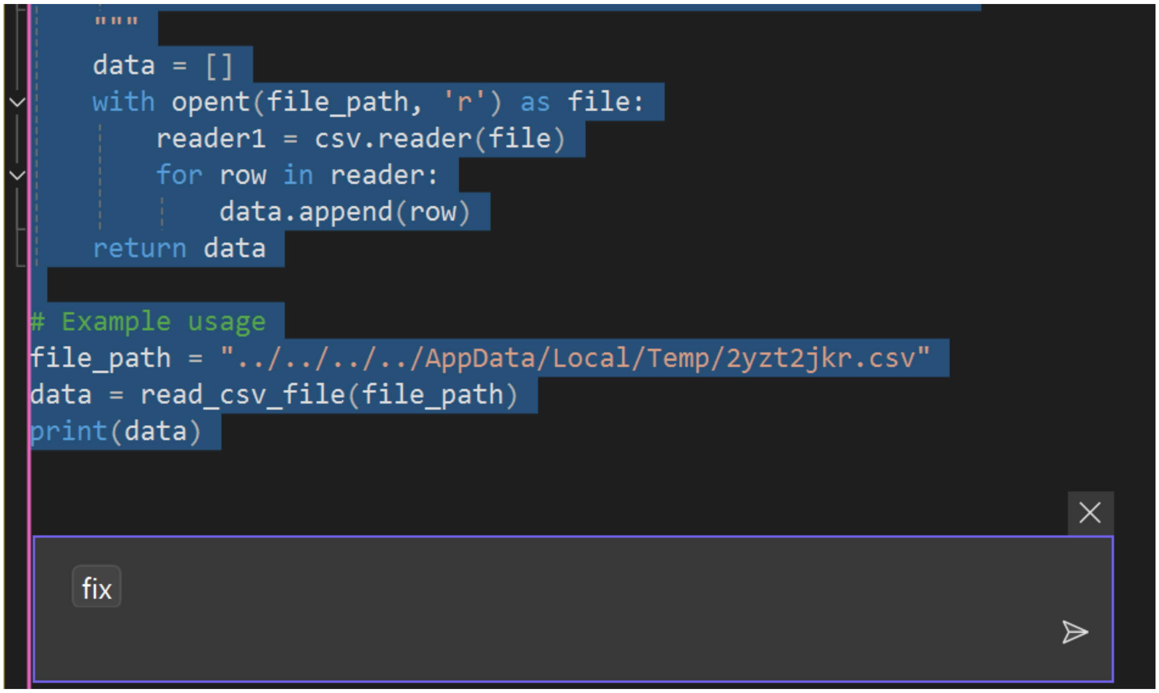
The Copilot adds the following comments to the selected code.

```python
import csv

def read_csv_file(file_path):
    """
    Read a CSV file and return the data as a list of rows.

    Args:
        file_path (str): The path to the CSV file.

    Returns:
        list: The data from the CSV file as a list of rows.
    """
    data = []
    with open(file_path, 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            data.append(row)
    return data

# Example usage
file_path = "../../../../AppData/Local/Temp/2yzt2jkr.csv"
data = read_csv_file(file_path)
print(data)
```

Copyright 2024 Mahesh Chand and CSharp Inc.

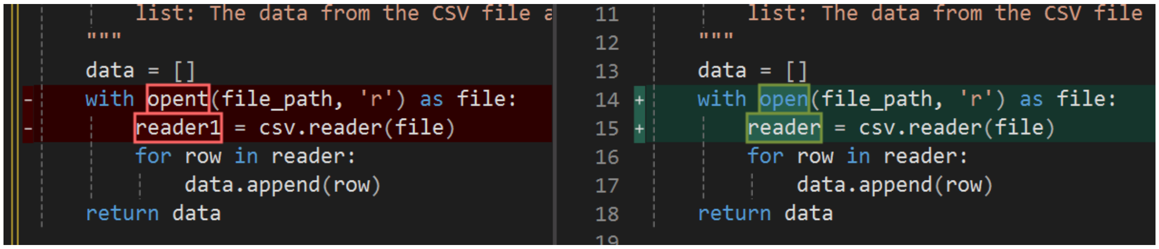
/fix: Propose a fix for problems in the selected code.

Let’s say there is an error in the code like spelling mistake or wrong variables, you can use /fix slash command for that.



```
    """
    data = []
    with opent(file_path, 'r') as file:
        reader1 = csv.reader(file)
        for row in reader:
            data.append(row)
    return data

# Example usage
file_path = "../../../../AppData/Local/Temp/2yzt2jkr.csv"
data = read_csv_file(file_path)
print(data)
```

The Copilot will show the fixed code that you may accept to fix the file.

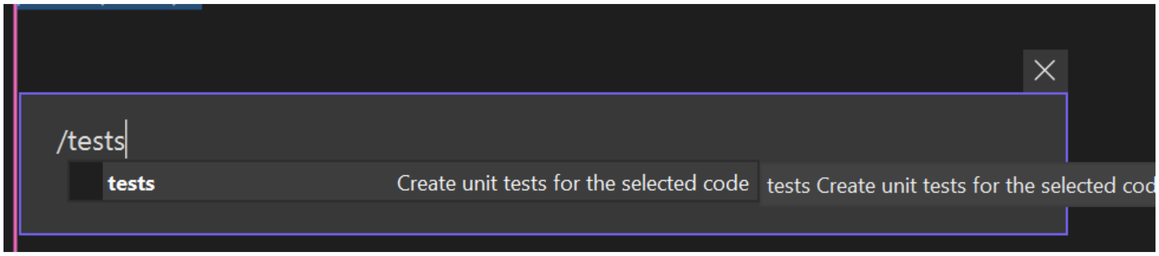


```
        list: The data from the CSV file a
    """
    data = []
-   with opent(file_path, 'r') as file:
-       reader1 = csv.reader(file)
        for row in reader:
            data.append(row)
    return data
```

```
        list: The data from the CSV file
    """
    data = []
+   with open(file_path, 'r') as file:
+       reader = csv.reader(file)
        for row in reader:
            data.append(row)
    return data
```

/tests: Create unit tests for the selected code.

Copilot generates test cases based on your selected code. Let’s select out previous code and ask it to generate test cases.



/tests
tests
Create unit tests for the selected code
tests Create unit tests for the selected cod

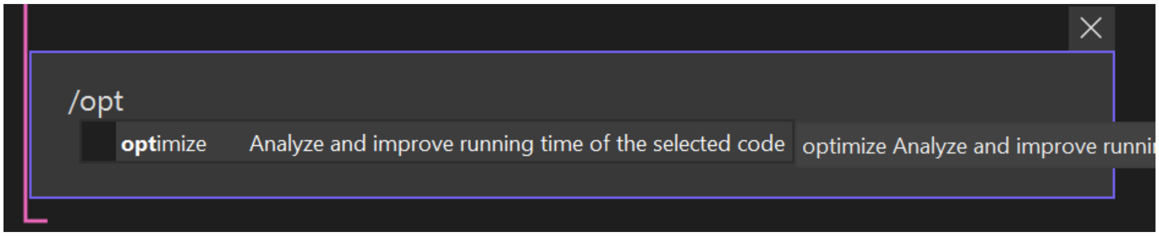
Copilot generates a tests file that looks like this:

Copyright 2024 Mahesh Chand and CSharp Inc.

```
import unittest
from unittest.mock import patch
from io import StringIO
from read_csv_file import read_csv_file

class ReadCSVFileTestCase(unittest.TestCase):
    def test_read_csv_file(self):
        file_path = "../../../../AppData/Local/Temp/test.csv"
        expected_data = [['1', 'John', 'Doe'], ['2', 'Jane', 'Smith'], ['3',
'Bob', 'Johnson']]

        with patch('builtins.open',
return_value=StringIO('1,John,Doe\n2,Jane,Smith\n3,Bob,Johnson\n')) as
mock_file:
            data = read_csv_file(file_path)
            self.assertEqual(data, expected_data)
            mock_file.assert_called_once_with(file_path, 'r')

    def test_read_csv_file_empty_file(self):
        file_path = "../../../../AppData/Local/Temp/empty.csv"
        expected_data = []

        with patch('builtins.open', return_value=StringIO('')) as mock_file:
            data = read_csv_file(file_path)
            self.assertEqual(data, expected_data)
            mock_file.assert_called_once_with(file_path, 'r')

    def test_read_csv_file_invalid_file(self):
        file_path = "../../../../AppData/Local/Temp/invalid.csv"

        with patch('builtins.open', side_effect=FileNotFoundError) as
mock_file:
            with self.assertRaises(FileNotFoundError):
                read_csv_file(file_path)
            mock_file.assert_called_once_with(file_path, 'r')

if __name__ == '__main__':
    unittest.main()
```

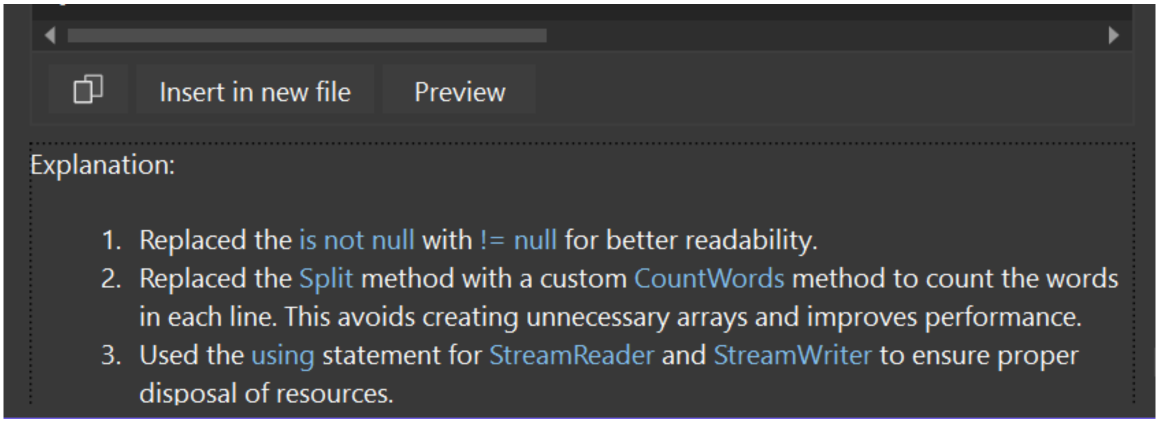
/optimize: Improve code performance

The /optimize slash command analyzes the existing code—whether it's the entire file or just a selected portion—and suggests optimizations. Use the following syntax to execute the /optimize slash command.


/opt
optimize
Analyze and improve running time of the selected code
optimize Analyze and improve runni

After completion, Copilot also explains what optimization has done in the selected code. It also gives you options to create a new file or preview and replace the selected code.

Copyright 2024 Mahesh Chand and CSharp Inc.



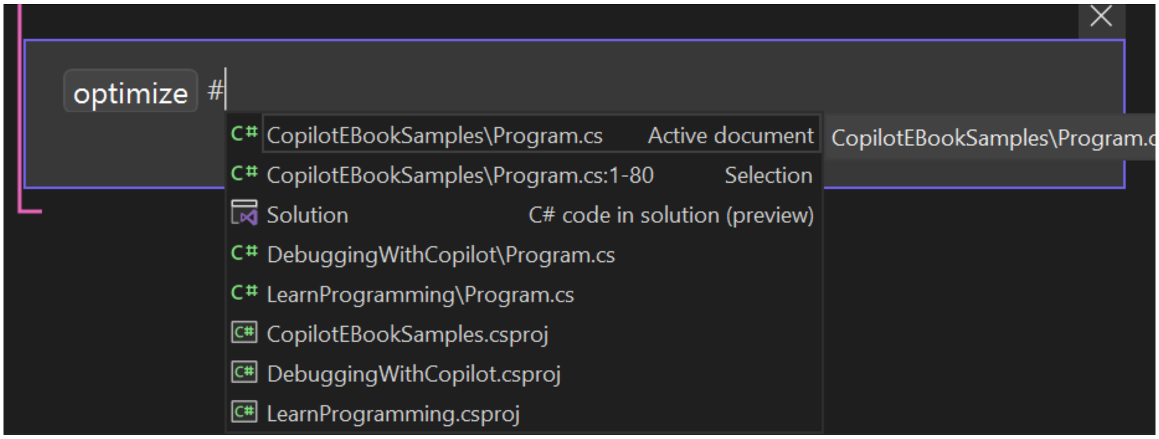
Explanation:

1. Replaced the is not null with != null for better readability.
2. Replaced the Split method with a custom CountWords method to count the words in each line. This avoids creating unnecessary arrays and improves performance.
3. Used the using statement for StreamReader and StreamWriter to ensure proper disposal of resources.

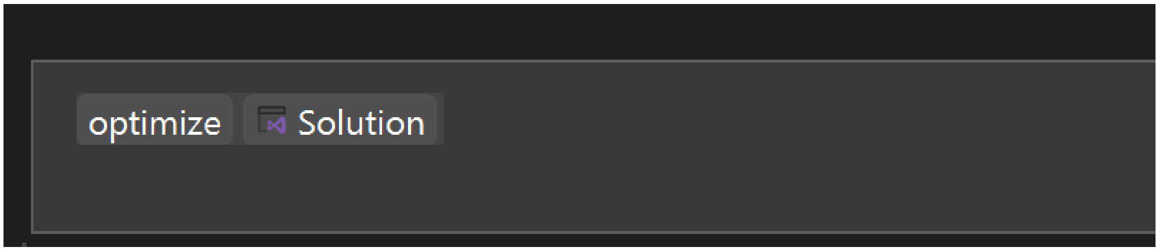
Reference: scope Copilot results to a particular file or entire solution

All the slash commands can be applied to selected code, a particular file, or even entire solution.

To select a particular file or entire solution, you use # symbol that load all available files and 'Solution" option.

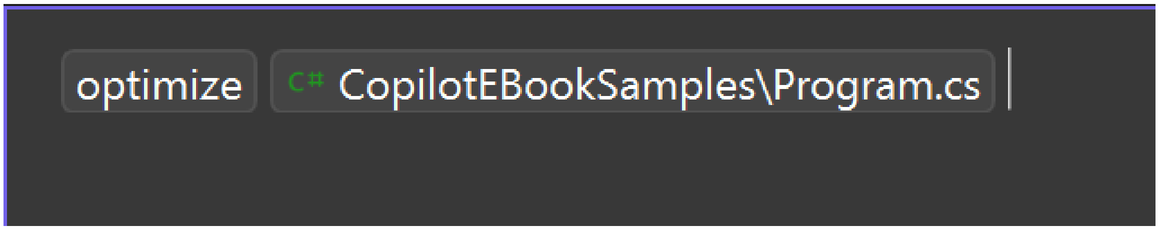

optimize #
CopilotEBookSamples\Program.cs Active document
CopilotEBookSamples\Program.cs:1-80 Selection
Solution C# code in solution (preview)
DebuggingWithCopilot\Program.cs
LearnProgramming\Program.cs
CopilotEBookSamples.csproj
DebuggingWithCopilot.csproj
LearnProgramming.csproj
CopilotEBookSamples\Program.c

If you select or type #Solution, the command works for the entire solution. For example, /optimize #Solution below will optimize the entire solution.



The following command will optimize only Program.cs in CopilotEBookSamples folder.

Copyright 2024 Mahesh Chand and CSharp Inc.



Copyright 2024 Mahesh Chand and CSharp Inc.

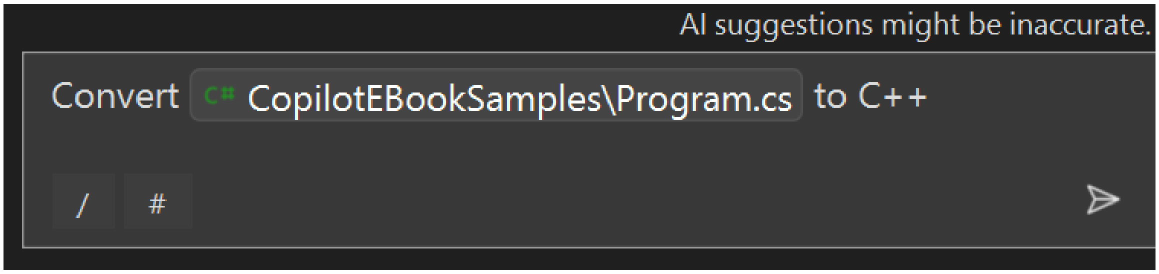
CHAPTER 9

CODE CONVERSION

Imagine you found some useful code, but it’s written in a different language than you need for your project. You can use Copilot to convert your code from one language to another. Let’s see how it is done.

I’ve a simple console app with Program.cs as the main code. I want to convert this code into C++.

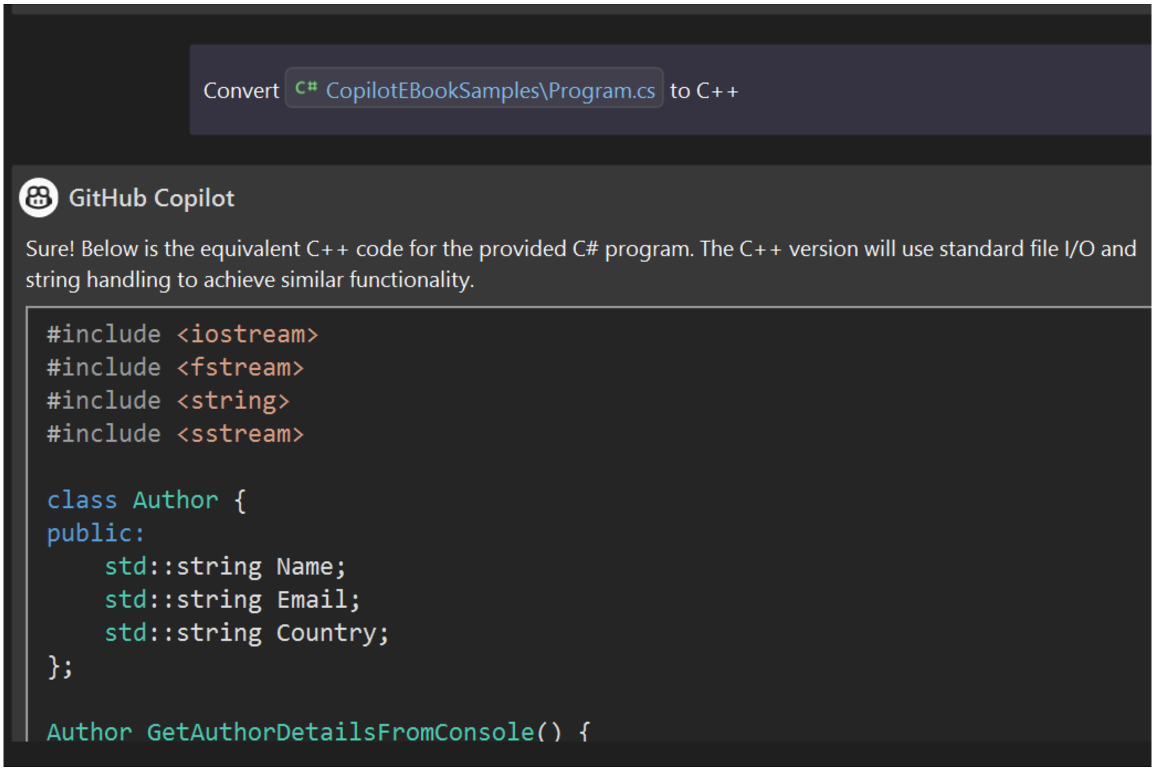
I simply write the following command:



AI suggestions might be inaccurate.

Convert C# CopilotEBookSamples\Program.cs to C++

Copilot generates C# code to C++ that looks like the following:



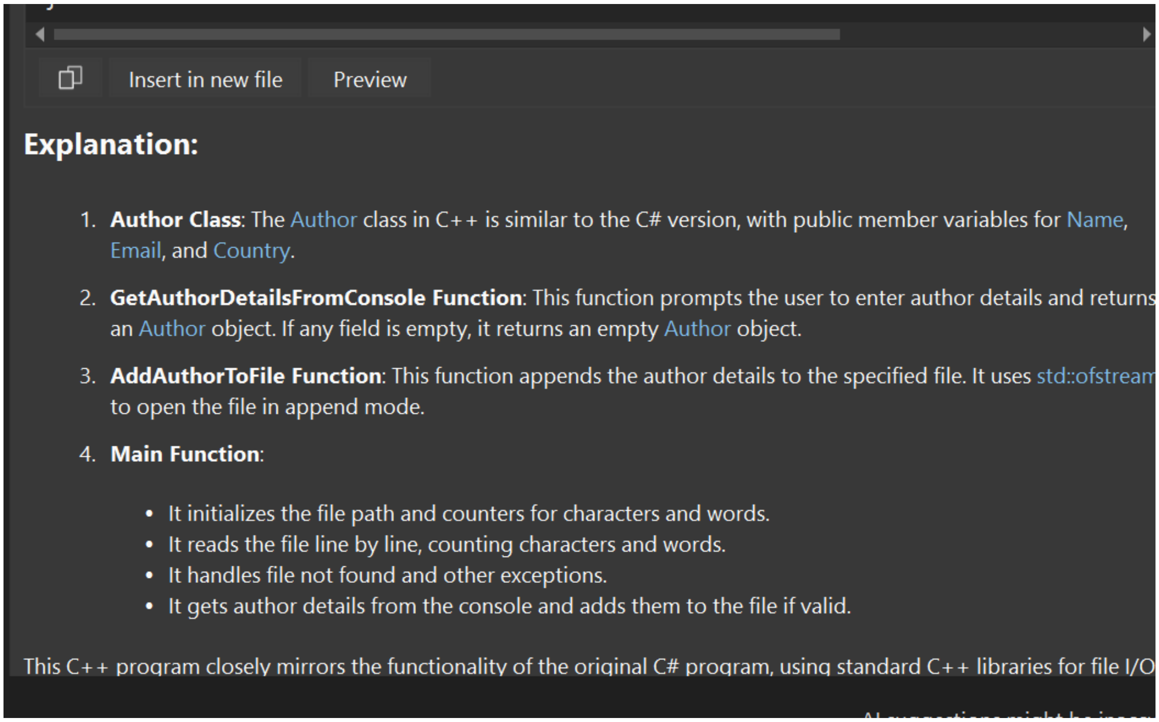
Convert C# CopilotEBookSamples\Program.cs to C++

Sure! Below is the equivalent C++ code for the provided C# program. The C++ version will use standard file I/O and string handling to achieve similar functionality.

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

class Author {
public:
    std::string Name;
    std::string Email;
    std::string Country;
};

Author GetAuthorDetailsFromConsole() {
```

Copyright 2024 Mahesh Chand and CSharp Inc.

And also writes an explanation of the C++ code in details.



Explanation:

1. **Author Class**: The Author class in C++ is similar to the C# version, with public member variables for Name, Email, and Country.
2. **GetAuthorDetailsFromConsole Function**: This function prompts the user to enter author details and returns an Author object. If any field is empty, it returns an empty Author object.
3. **AddAuthorToFile Function**: This function appends the author details to the specified file. It uses std::ofstream to open the file in append mode.
4. **Main Function**:
    - It initializes the file path and counters for characters and words.
    - It reads the file line by line, counting characters and words.
    - It handles file not found and other exceptions.
    - It gets author details from the console and adds them to the file if valid.

This C++ program closely mirrors the functionality of the original C# program, using standard C++ libraries for file I/O

On this screen, you can insert this code in a new file or preview it.

Using the same method, you can easily convert code from one language to another.

As a reminder, you want to check the newly converted code is written the way you like and doesn't have any bugs or issues. While Copilot is good at creating new code, it learns from existing old code. That means, the newly generated code may work just fine but there is no guarantee that the code is optimized and uses the latest version of C++.

Copyright 2024 Mahesh Chand and CSharp Inc.

SUMMARY

Copilot is your AI buddy that is available to you all the time. It never takes vacation. If used properly, it can boost your productivity and help you code faster.

In this book, we discussed the basics of Copilot, how it works, and how to use Copilot in Visual Studio. Copilot can be used in other popular IDEs as well.

You can use Copilot to learn programming in the following ways:

- Learn a new programming language from scratch as we have already seen in the beginning of this book.
- Update your existing code to a newer version and ask Copilot to explain it to you.
- Convert your code from one programming language to another. For example, you can ask your Java code to convert to C# or Python.
- Fix your existing code and check for performance.
- Document and add comments to your existing code or project.
- And more

Copyright 2024 Mahesh Chand and CSharp Inc.

ABOUT C# CORNER

C# Corner is a prominent online community for software developers established in 1999. Initially, it began as a small group of Microsoft technology focused programmers but has since grown exponentially to include over 3 million registered IT professionals globally, with nearly 30 million unique visitors annually and now covering various cutting-edge technologies including Microsoft tech stack, cloud, mobile, blockchain, and AI.

C# Corner community focuses on its members growth by providing learning resources, guidance, mentorship and helps members earn reputation for their contributions.

C# Corner is a member contributions based community where members contribute in form of sharing their code, publishing articles, blogs, tutorials, submit training, and speak at CSharp,TV and conferences.

C# Corner soon will be accessible using CSharp.com URL.

History and Growth

C# Corner was founded by Mahesh Chand in 1999. The platform started as a niche community focused on C# and .NET programming. Over the years, it expanded its scope to cover a wide range of topics relevant to software development, including various programming languages, frameworks, and technologies.

Copyright 2024 Mahesh Chand and CSharp Inc.

The community's growth has been organic, driven by the active participation and contributions of its members. C# Corner has become a hub for developers to share knowledge, learn new skills, collaborate on projects, and grow in their professional lives. The platform provides a variety of resources, including articles, tutorials, forums, events, growth features, and conferences, which have contributed to its popularity and widespread use.

Features and Offerings

Educational Resources:

C# Corner offers a vast array of educational materials, including articles, tutorials, and videos. These resources cover a wide range of topics from beginner to advanced levels, making it a valuable resource for developers at any stage of their career.

Community Engagement:

The platform hosts forums and discussion boards where members can ask questions, share insights, and collaborate on projects. This interactive aspect fosters a sense of community and continuous learning among its members.

Copyright 2024 Mahesh Chand and CSharp Inc.

Events and Conferences:

C# Corner organizes various events, including webinars, workshops, and conferences. These events provide opportunities for members to learn from industry experts, network with peers, and stay updated on the latest trends and technologies.

Local user group chapters are organized in more than 50 cities around the world. C# Corner also hosts an annual C# Corner conference that focuses on emerging technologies. Check out the conference website here: CSharpCorner Annual Conference.

Copyright 2024 Mahesh Chand and CSharp Inc.

Download eBooks

C# Corner has a wide range of free eBooks starting from topics of C#, ASP.NET and Python programming to AI and Blockchain. More new eBooks are being added each month. Check out free eBooks library here: C# Corner eBooks Library.

Python Libraries for Machine Learning

Free

Machine Learning for Future Engineers

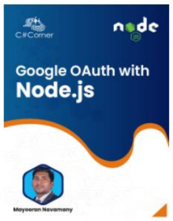
Free

Pattern Matching in C#

~~$19.00~~ Free

Python Overview

~~$9.99~~ Free



Google OAuth with node js

Free

Yatharth Machine Learning

~~$9.99~~ Free

Mastering SOLID Principles in CSharp

Free

Mastering Docker A Comprehensive Guide

Free

Basics of SQL Server

~~$29.00~~ Free

MongoDB Interview Q & A

Free

Copyright 2024 Mahesh Chand and CSharp Inc.

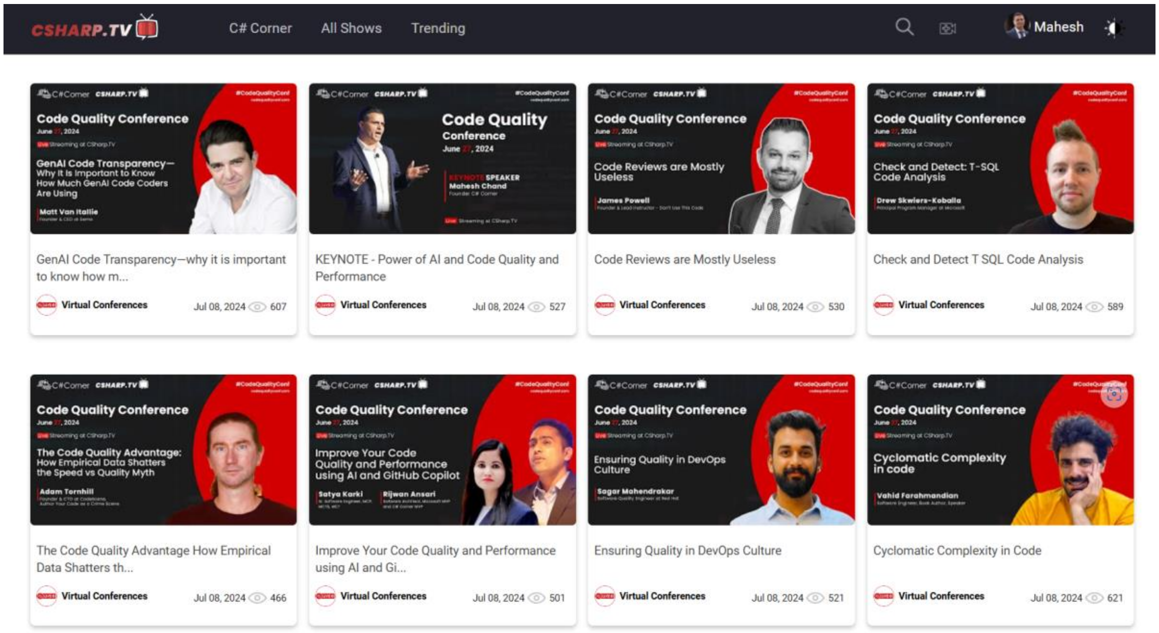
Recognition Programs:

To encourage active participation, C# Corner has recognition programs such as the Most Valuable Professional (MVP) awards. These awards honor members who make significant contributions to the community, thereby motivating others to engage more actively.

CSharp.TV

CSharp.TV is a streaming platform dedicated to software developers. CSharp.TV hosts various weekly live shows, virtual conferences, and local user group events. It also has thought leaders panel discussions on emerging technologies. Check out CSharp.TV here.

Copyright 2024 Mahesh Chand and CSharp Inc.



GenAI Code Transparency—why it is important to know how m...
Virtual Conferences Jul 08, 2024 607

KEYNOTE - Power of AI and Code Quality and Performance
Virtual Conferences Jul 08, 2024 527

Code Reviews are Mostly Useless
Virtual Conferences Jul 08, 2024 530

Check and Detect T SQL Code Analysis
Virtual Conferences Jul 08, 2024 589

The Code Quality Advantage How Empirical Data Shatters th...
Virtual Conferences Jul 08, 2024 466

Improve Your Code Quality and Performance using AI and Gi...
Virtual Conferences Jul 08, 2024 501

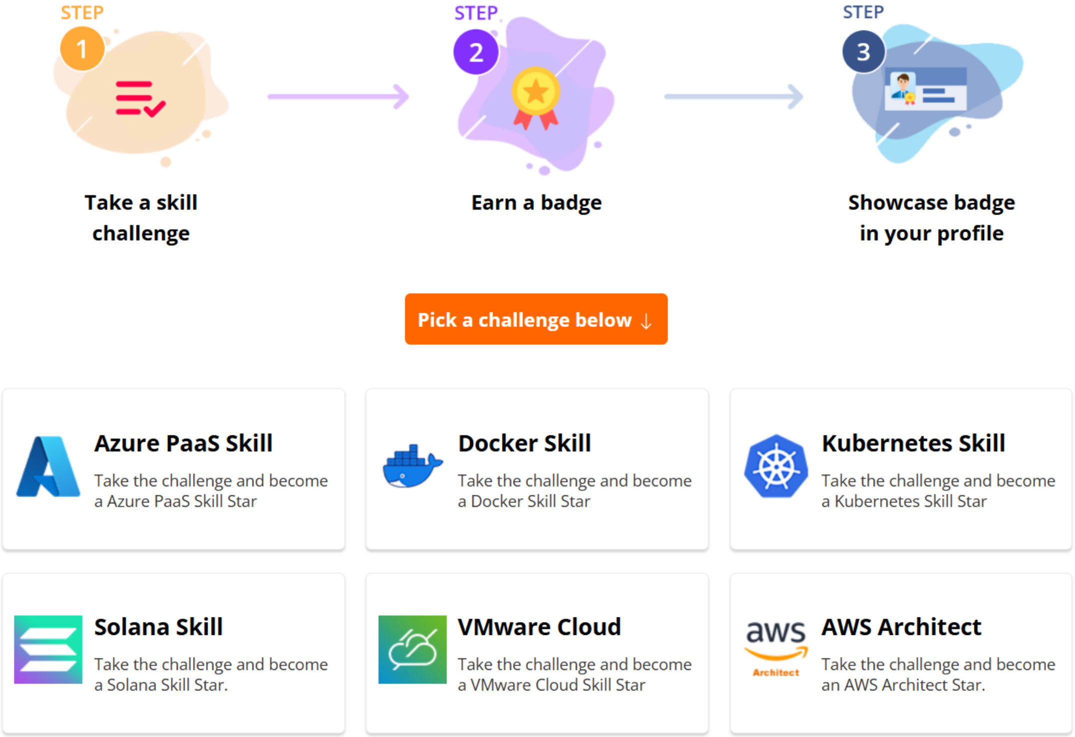
Ensuring Quality in DevOps Culture
Virtual Conferences Jul 08, 2024 521

Cyclomatic Complexity in Code
Virtual Conferences Jul 08, 2024 621

Challenges

C# Corner offers a variety of skill challenges designed to test and improve your knowledge in different areas of technology. These challenges cover a wide range of topics including C#, Azure, Docker, Kubernetes, and more. Each challenge allows you to earn badges and showcase them on your profile, providing a fun and competitive way to learn and demonstrate your skills. Check out Challenges here: [C# Corner - Skills Challenges.]()

Copyright 2024 Mahesh Chand and CSharp Inc.


STEP 1
Take a skill challenge

STEP 2
Earn a badge

STEP 3
Showcase badge in your profile

Pick a challenge below ↓

Azure PaaS Skill
Take the challenge and become a Azure PaaS Skill Star

Docker Skill
Take the challenge and become a Docker Skill Star

Kubernetes Skill
Take the challenge and become a Kubernetes Skill Star

Solana Skill
Take the challenge and become a Solana Skill Star.

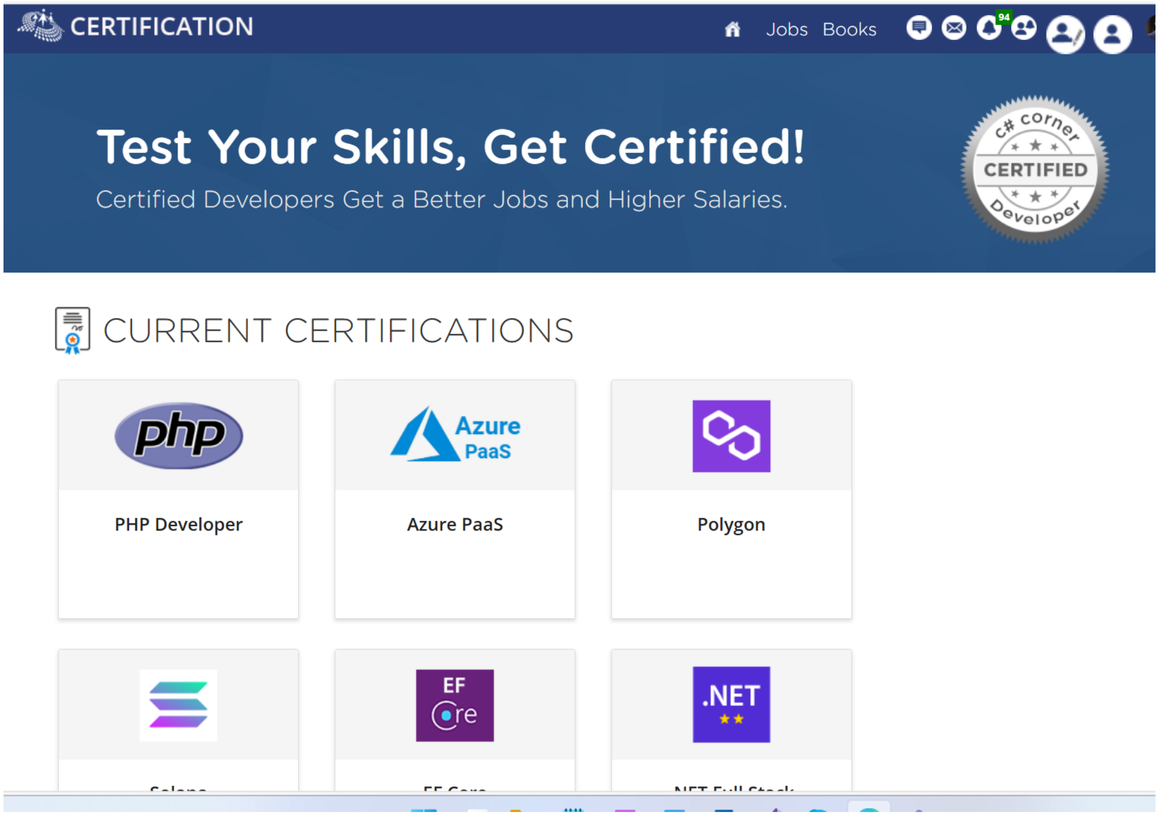
VMware Cloud
Take the challenge and become a VMware Cloud Skill Star

AWS Architect
Take the challenge and become an AWS Architect Star.

Certifications

C# Corner offers various certifications and skill challenges aimed at developers looking to enhance their expertise in different areas of technology. These certifications can be earned through completing specific challenges and courses available on the C# Corner platform. Check out Certifications here: [Get Certified! Certified Developers!.]()

Copyright 2024 Mahesh Chand and CSharp Inc.



Test Your Skills, Get Certified!

Certified Developers Get a Better Jobs and Higher Salaries.

CURRENT CERTIFICATIONS

PHP Developer

Azure PaaS

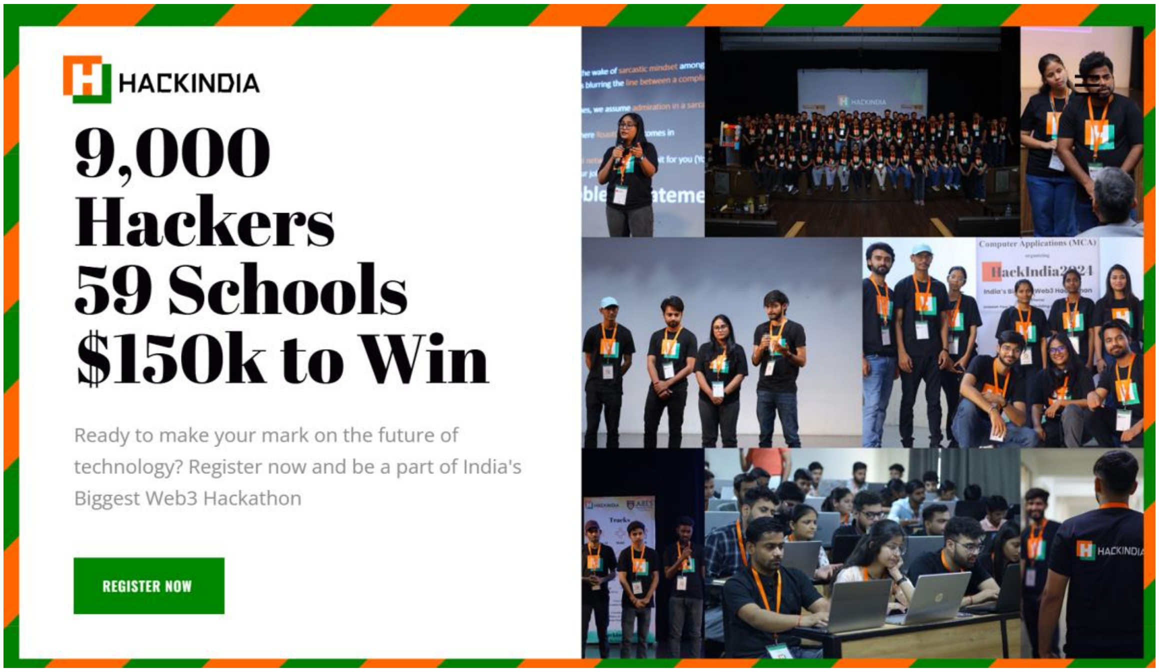
Polygon

HackIndia

HackIndia 2024 is a large-scale hackathon organized by CSharpCorner, a global community of software developers. It is focused on Web3 and AI technologies, running from May to October 2024. The event brings together over 9,000 students from various universities across India, competing for $150,000 in prizes. Participants gain insights from 36 speakers, guidance from 72 judges and mentors, and opportunities to showcase their skills and innovation in Web3 and AI. The final event will be held in Delhi in October 2024.

HackIndia Website: [HackIndia 2024 - India's Biggest Web3 Hackathon]

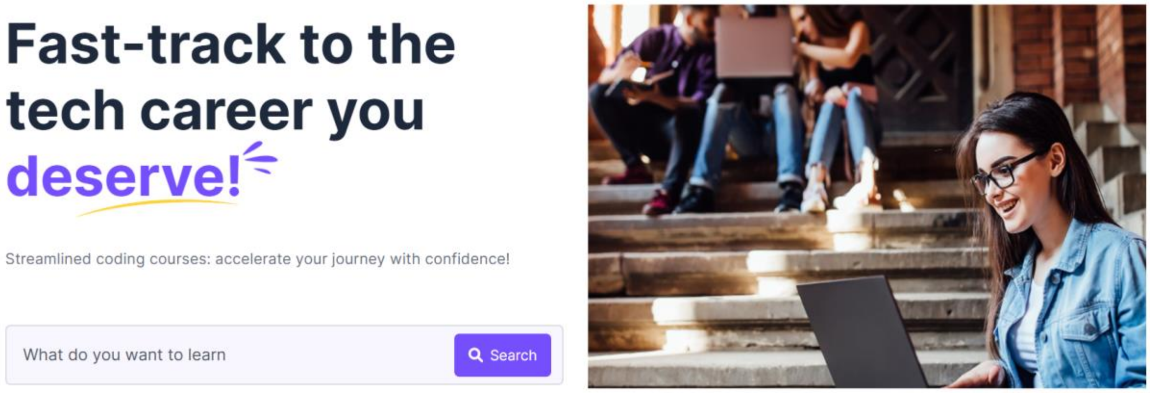
Copyright 2024 Mahesh Chand and CSharp Inc.



Ready to make your mark on the future of technology? Register now and be a part of India's Biggest Web3 Hackathon

Students Portal

Student Portal is designed for college students to get help with learning new technologies, signup for new learning courses, participate in skills challenges, find, and apply for internships and participate in hackathons. Check out CSharp.com Student Portal.


Fast-track to the tech career you deserve!
What do you want to learn
Search

Streamlined coding courses: accelerate your journey with confidence!

Join Today!

We’re calling for professionals to become active community members. C# Corner’s mission is to provide software professionals with a single platform to learn, share, collaborate, and growth.

Become a C# Corner Member Today!

Copyright 2024 Mahesh Chand and CSharp Inc.