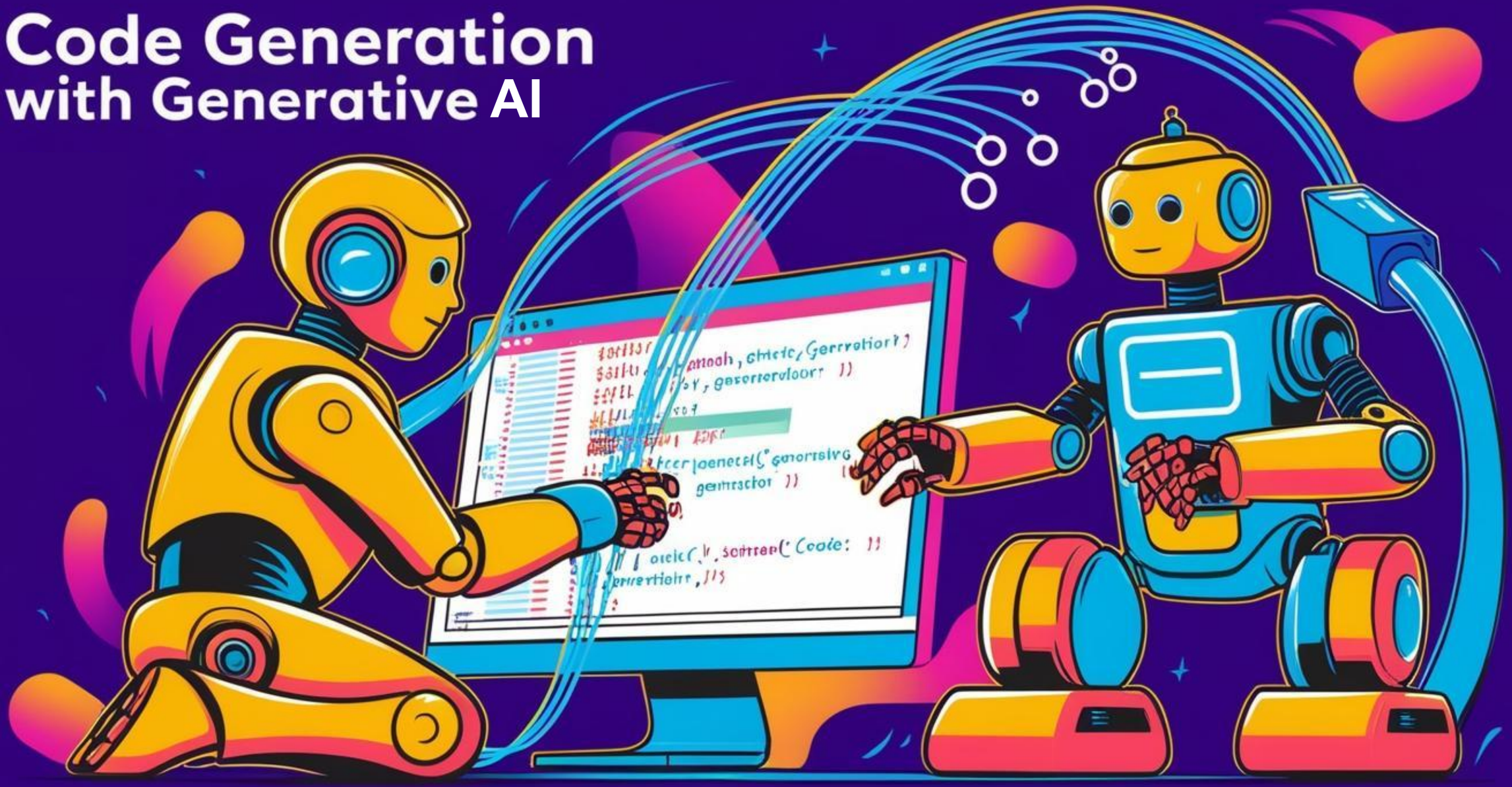
Code Generation with Generative AI



Presented By - Ms. Charu Rathour

Generated with canva

Outline

01. How Software Development Evolved

02. Github Survey and Experiment on Developers

03. Benefits of Code Generation with Gen AI

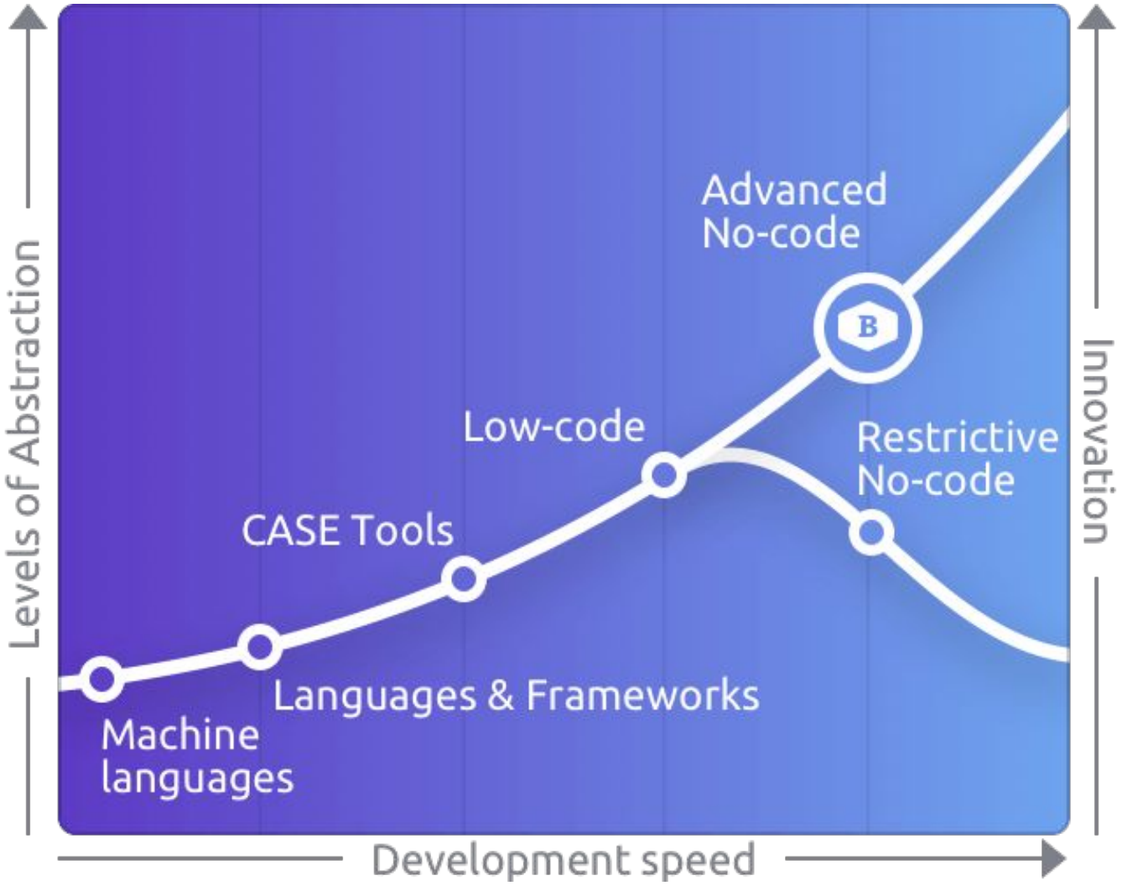
04. LLMs for Code Generation and Datasets used for their Training

05. AI tools for Code Generation

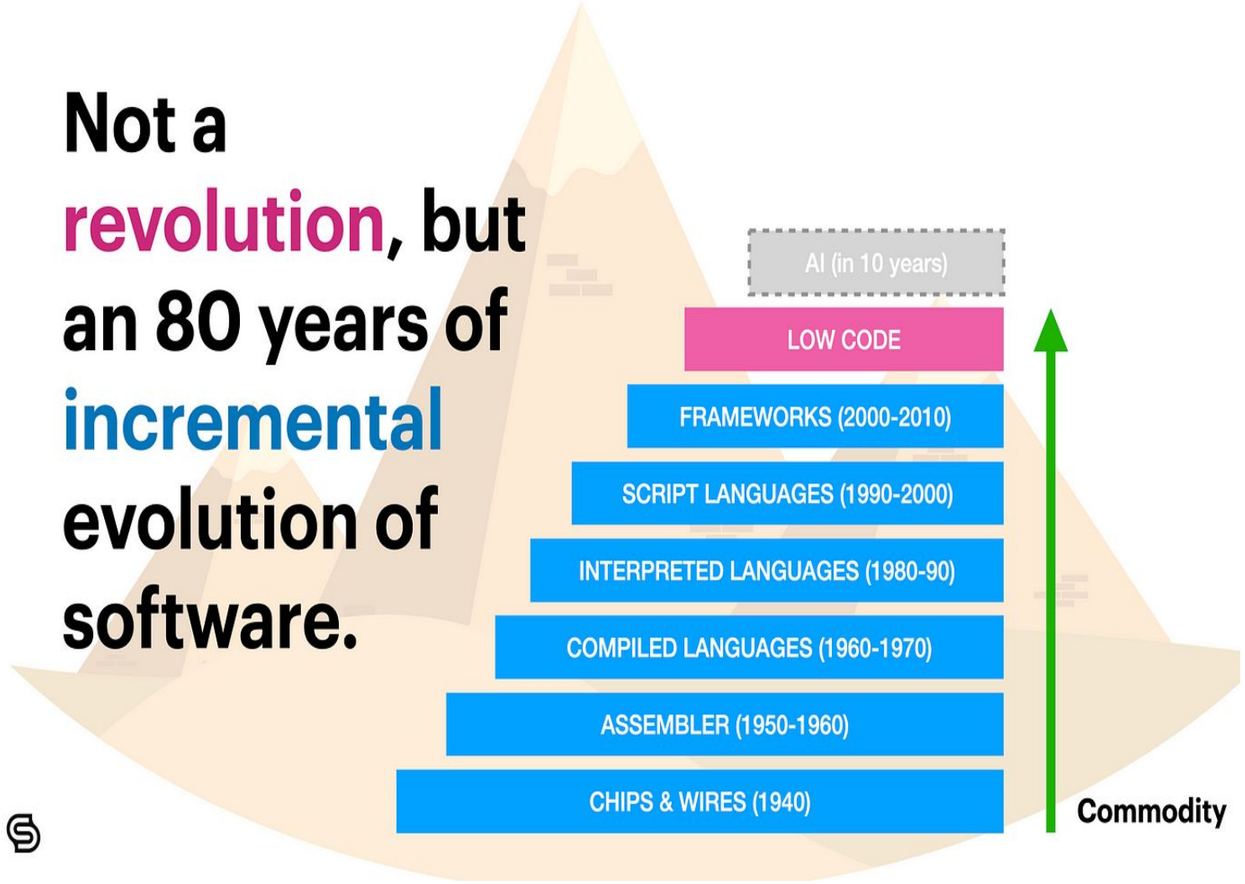
Outline

How Software Development Evolved

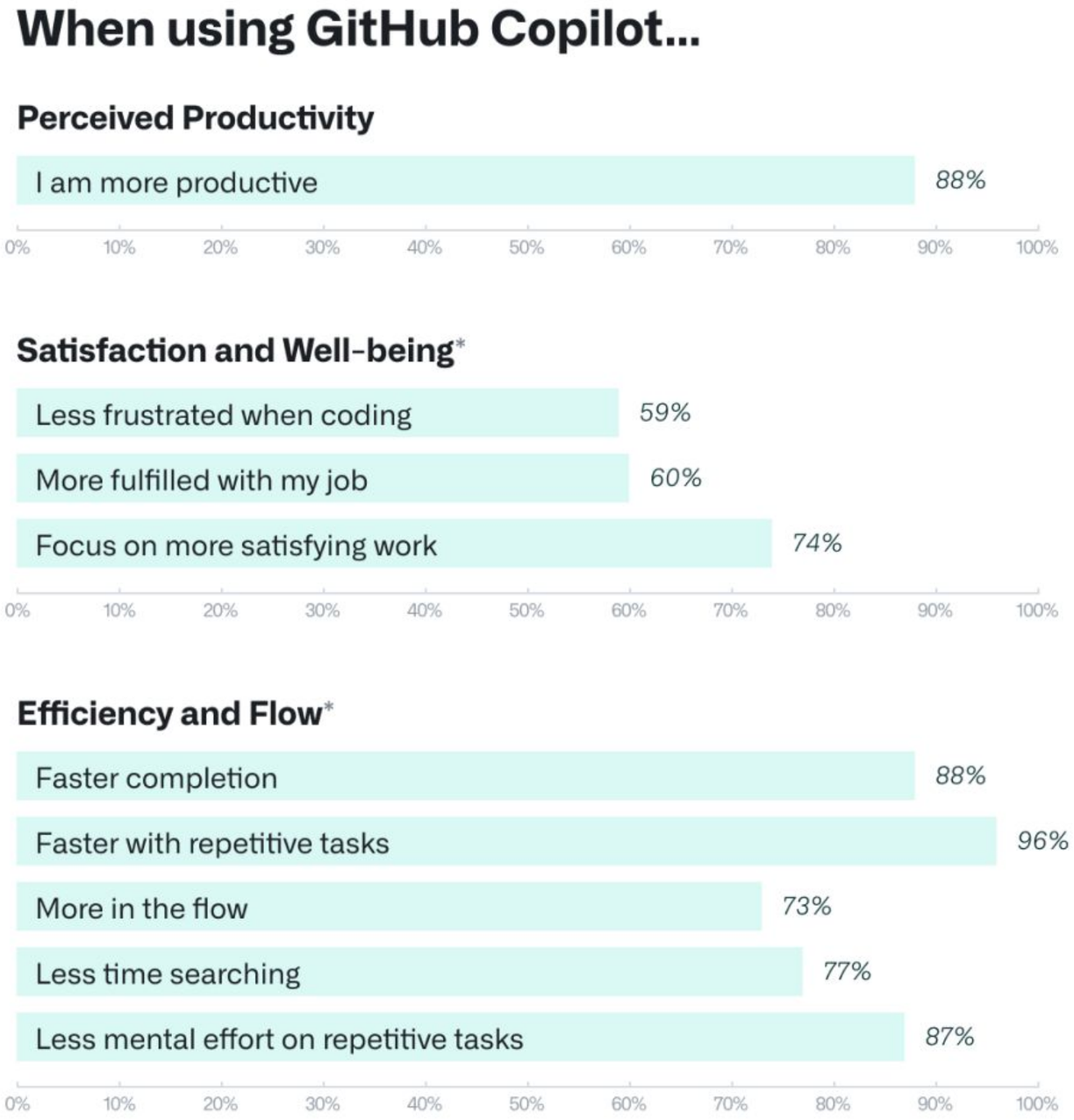
- Coding Evolution: From manual binary coding to high-level programming languages, complexity has reduced significantly over time.
- Efficiency Growth: Each phase introduced tools that made writing, debugging, and scaling code faster and easier.
- AI's Role: Today, AI tools simplify coding further, automating tasks and enhancing productivity.



Source - [turnk]



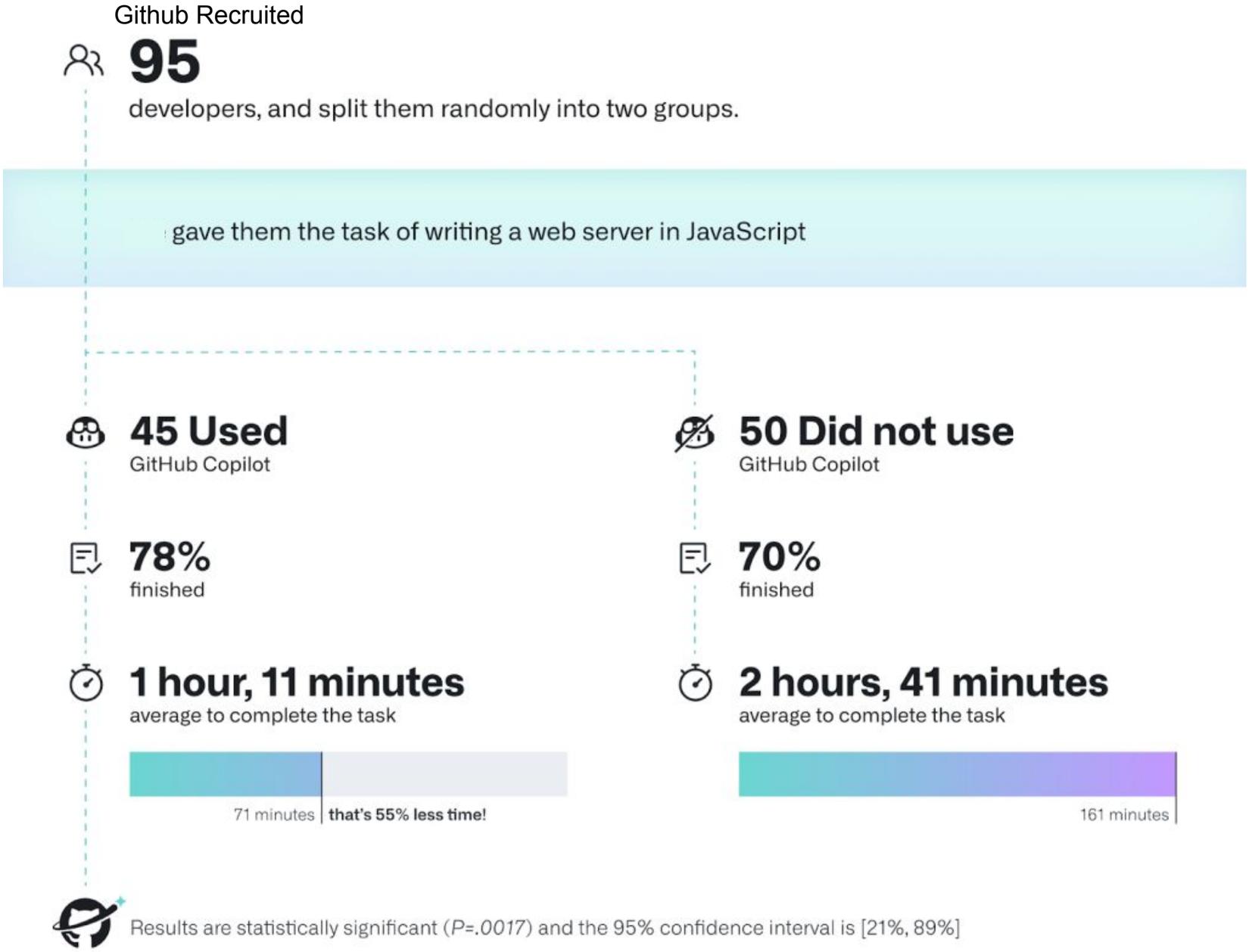
Source - [code.store]



Github surveyed more than 2,000 developers to learn at scale about their experience using GitHub Copilot and found out

Source- [Research: quantifying GitHub Copilot's impact on developer productivity and happiness]

Github Recruited

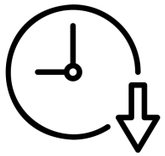


In Another Experiment Done by Github they observed

- Developers using Copilot completed tasks 55% faster and had higher completion rates than those without it.

Source - [Research: quantifying GitHub Copilot's impact on developer productivity and happiness]

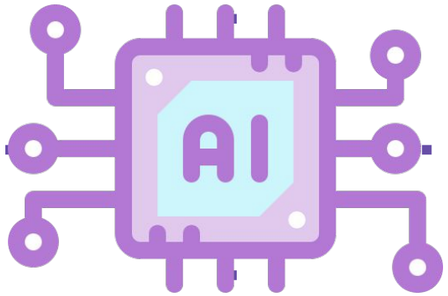
Benefits of AI Driven Code Generation



Reduced Development Time

Simplified Debugging

Improved Code Quality

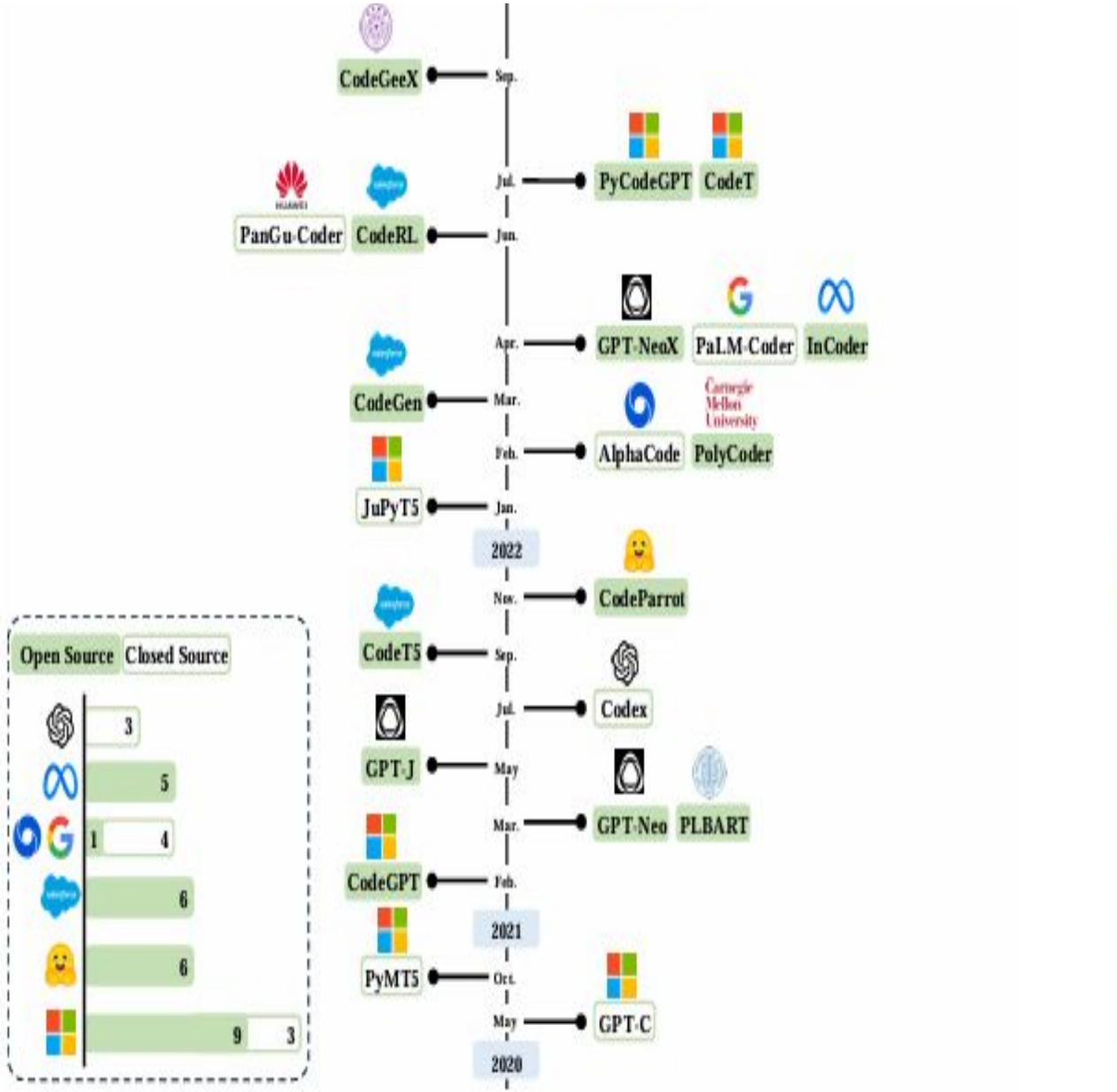


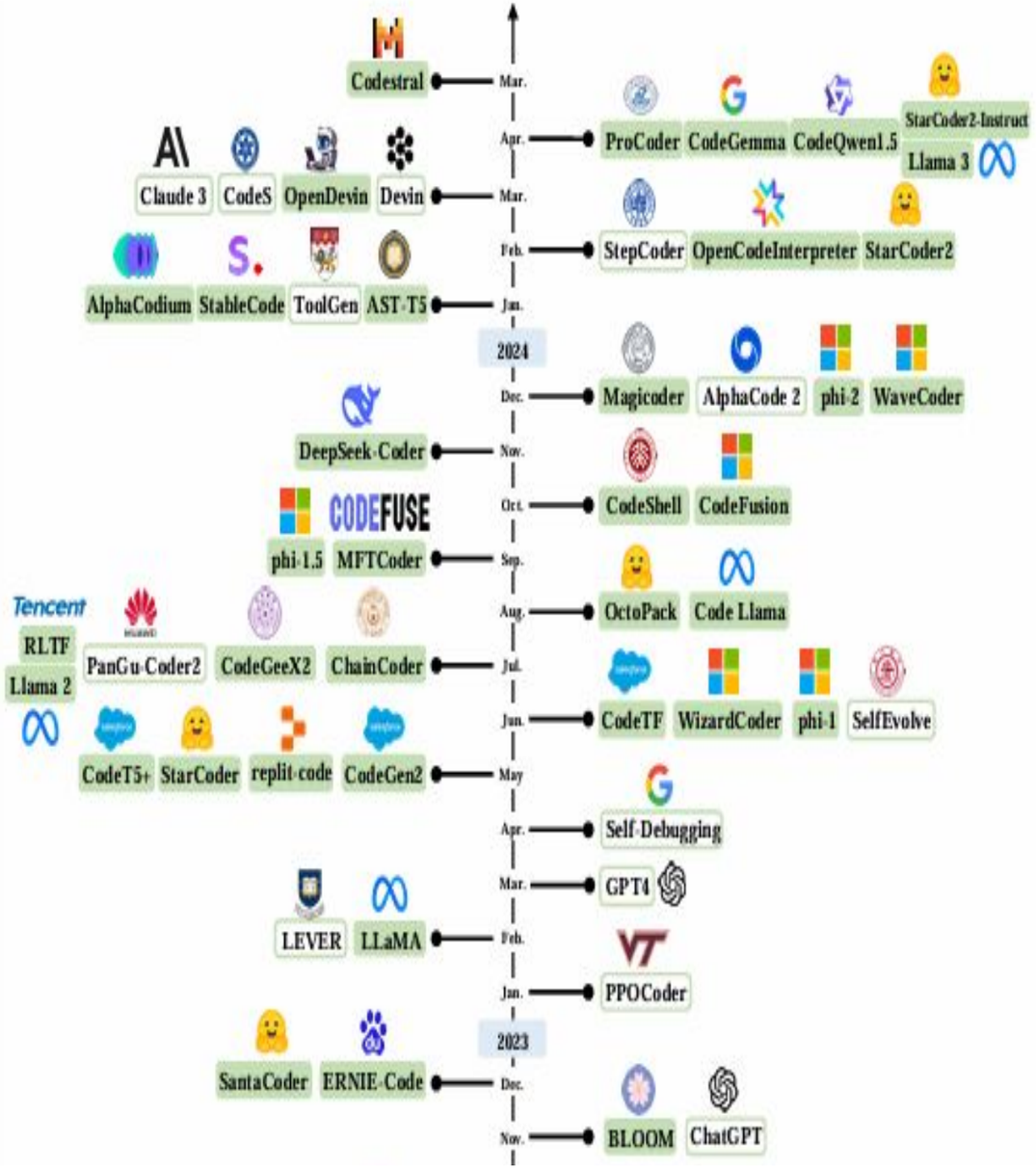
Learning Assistance

Cross Language Flexibility

Task Specific Outputs

Large Language Models Trained for Code Generation





Source- A Survey on Large Language Models for Code Generation

Datasets used for Training LLMs for Code Generation

Dataset	Size (GB)	Files (M)	#PL
CodeSearchNet [110]	20	6.5	6
Google BigQuery[96]	-	-	-
The Pile [78]	95	19	-
CodeParrot [254]	180	22	1
GitHub Code[254]	1,024	115	32
ROOTS [137]	163	15	13
The Stack [132]	3,136	317	30
The Stack v2 [170]	32K	3K	619

Datasets used for pre-training LLMS

Dataset	Size	#PL
CodeAlpaca-20K [43]	20k	-
CommitPackFT [187]	2GB	277
Evol-Instruct-Code-80k [225]	80k	-
evol-codealpaca-v1 [251]	110K	-
Magicoder-OSS-Instruct-75k [278]	75k	Python, Shell, TypeScript, C++, Rust, PHP, Java, Swift, C#
Self-OSS-Instruct-SC2-Exec-Filter-50k [304]	50k	Python

Datasets used for instruction tuning LLMS

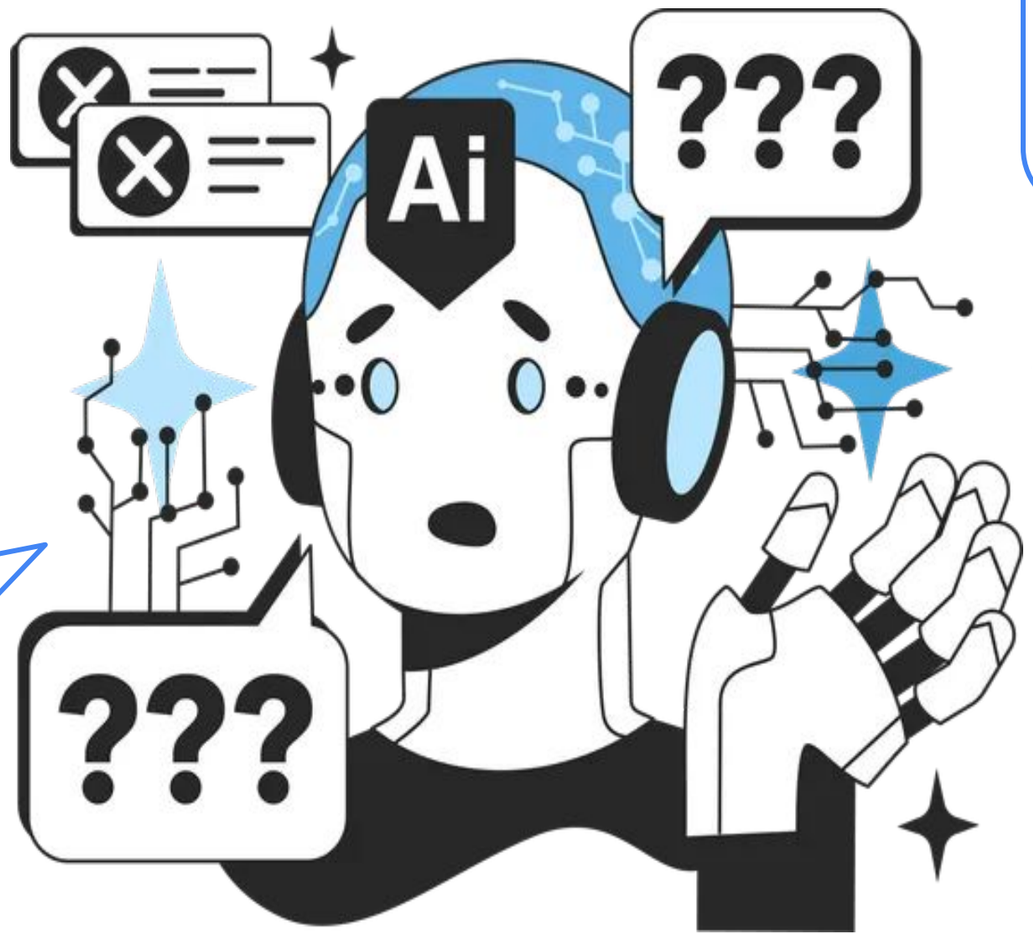
#PL = Number of programming languages included in the dataset

Source- A Survey on Large Language Models for Code Generation

Challenges of Code Generation with Generative AI

Understanding Natural and Programming Languages Both Which can be confusing

Handling Diverse Input/Output Requirements with varying Complexities



Balancing Readability, Efficiency, Correctness, and Completeness

Due to the inherent complexity of programming tasks LLMs can Hallucinate

Prompt Engineering to the Rescue

Be specific: Clearly define task requirements and expected functionality.

Specify language: Mention the programming language in the prompt explicitly.

Include examples: Provide input-output examples to clarify expected behavior.

State libraries: List required libraries or frameworks for the task.

Ask for comments: Request code with inline comments for better understanding.

Limit complexity: Break complex tasks into smaller, manageable subtasks for clarity.

Focus on constraints: Specify performance, scalability, or formatting requirements.

Iterate prompts: Refine and retry prompts to improve the generated code.

Use templates: Provide partial code or templates for the model to follow.

Request explanations: Ask for explanations alongside code to verify logic and intent

AI Tools For Code Generation

	GitHub Copilot	ChatGPT	tabnine	Amazon Q Developer	Devin
Supported Languages	All Langauges	All Popular Languages	20 Programming Languages	15 Programming Languages	All Popular Languages
Integration	Works with Visual Studio Code, JetBrains IDEs, and Neovim	Works through a chat interface	Works with popular IDEs like VS Code, JetBrains, Sublime Text	Integrated into IDEs such as VS Code, JetBrains, and AWS Cloud9	Works with IDEs like VS Code and JetBrains
Underlying Model	OpenAI Codex	GPT-4o, o1	Tabnine Model	Unknown	Diverse
Price	Starting at $10 Per Month	Pro Plan Starts $20/Month	Pro Plan Costs $12/Month	Pro Version Costs $19/Month	Basic Plan Starts at $25/Month
Code Completion	✓	✓	✓	✓	✓
Code Generation	✓	✓	✓	✓	✓
Debugging Assistance	✓	✓	✓	✓	✓
Documentation Generation	✓	✓	✓	✗	✗

Features	ChatGPT	Amazon CodeWhisperer	GitHub Copilot
IDE Support	No IDE Support	JetBrains, Visual Studio Code, AWS Cloud9, or the AWS Lambda console	IntelliJ IDEA, Android Studio, AppCode, CLion, Code With Me Guest, DataGrip, DataSpell, GoLand, JetBrains Client, MPS, PhpStorm, PyCharm, Rider, RubyMine, WebStorm
First Release Time	Nov-30-2022	June-23-2022	Oct-29-2021
Developer	OpenAI	AWS	OpenAI-Microsoft
Providing References to Suggestions	NO	YES	NO
Explanation of Suggestions	YES	NO	NO
Providing Multiple Suggestions	NO (Theoretically user can manually ask for another suggestion.)	YES (Up to 5)	YES (Up to 10)
Training Data Source	GitHub Repositories, OpenAI Codex Dataset, other code repositories such as GitLab, Bitbucket, and SourceForge	"Vast amounts of publicly available code"	"...trained on all languages that appear in public repositories" (Fine-tuned)
Programming Languages work best with (according to the vendor)	N/A	C#, Java, JavaScript, Python, and TypeScript	C, C++, C#, Go, Java, JavaScript, PHP, Python, Ruby, Scala, and TypeScript
Multipurpose (other than programming)	YES	NO	NO
Subscription	ChatGPT Free ChatGPT Plus ($20 per month)	Free Preview	Copilot for Students (Free) Copilot for Individuals ($10 per month) Copilot for Business ($19 per user, per month)
Can be Used Offline?	NO	NO	NO
Can it Access Local Files?	NO	YES	YES

Comparing Relevant Code Generation Tools

Source- https://arxiv.org/pdf/2304.10778

Getting started with

GitHub Copilot

Free Version

What features are included in Copilot Free?

Code completion	in different IDEs Visual Studio Code, Visual Studio, JetBrains IDES, Vim/Neovim, Xcode, and Azure Data Studio
Copilot Edits	to make changes across multiple files (only in Visual Studio Code and Visual Studio)
Copilot Chat	Chat window in Visual Studio Code, Visual Studio, JetBrains IDES, and GitHub.com
Models available	Claude 3.5 Sonnet and GPT 4o

What are the limitations of Copilot Free?

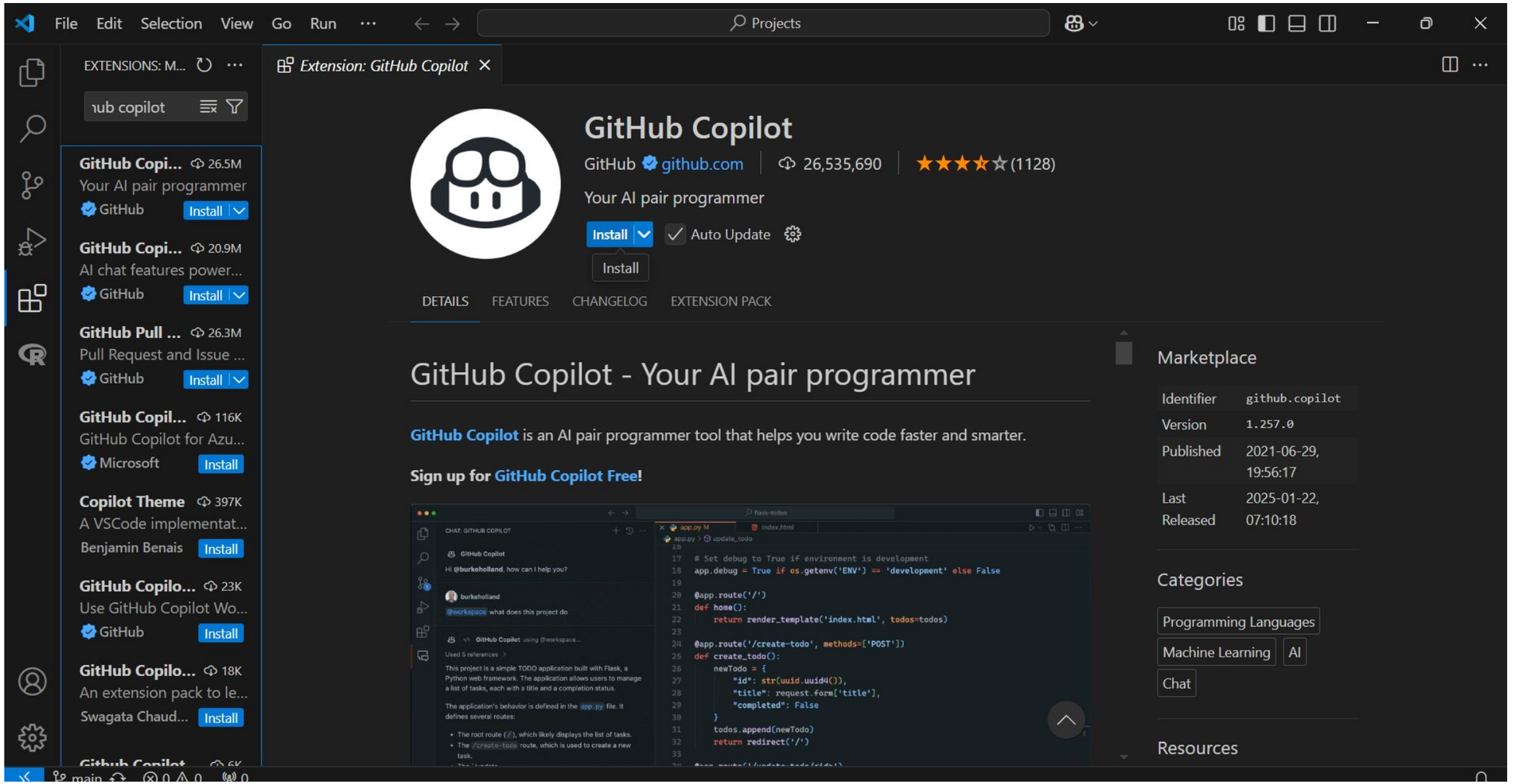
Code completions	2000 completions per month.
Copilot Chat	50 chat messages per month. This limit includes both standard chats and multi-file editing chats in VS Code and Visual Studio

When you reach these limits, you can upgrade to Copilot Pro to continue using Copilot.

Set up Visual Studio Code with Copilot

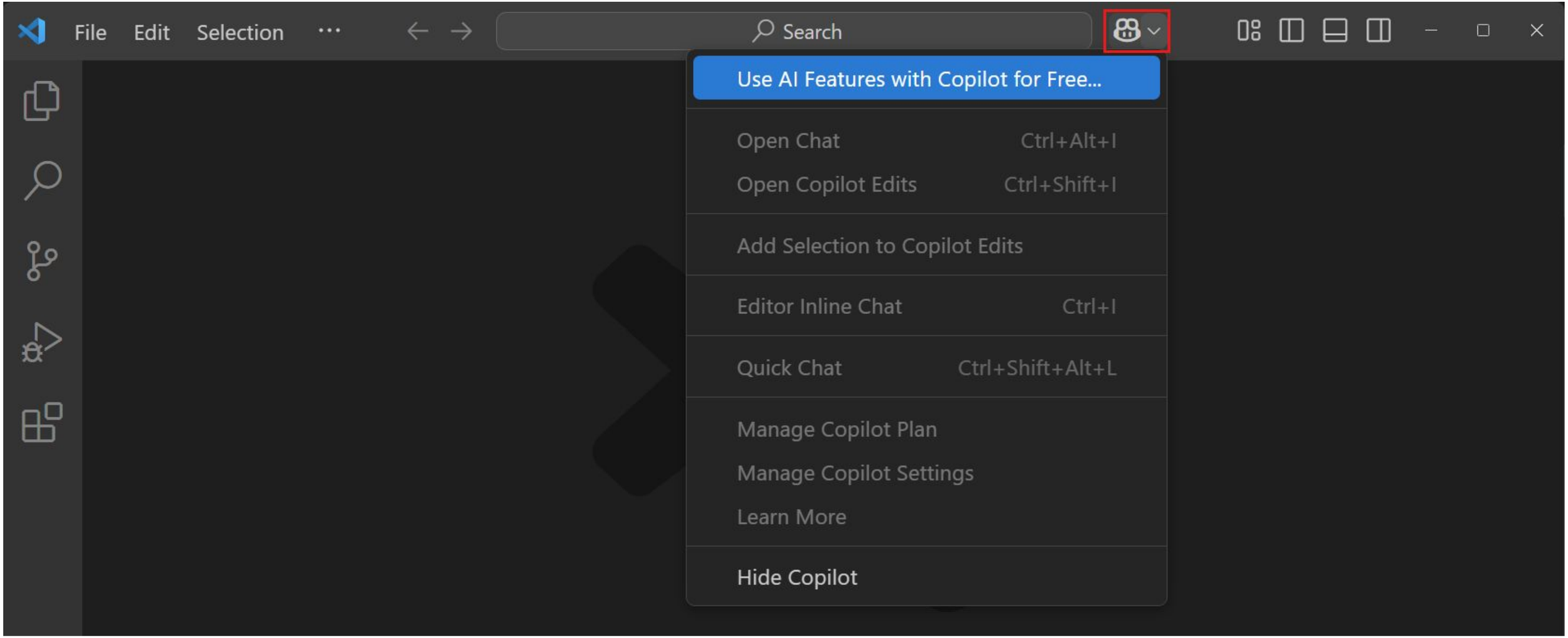
To get started with Copilot Free in Visual Studio, you need:

1. Ensure you have a recent version of VS Code
2. GitHub Copilot Extension in VS Code



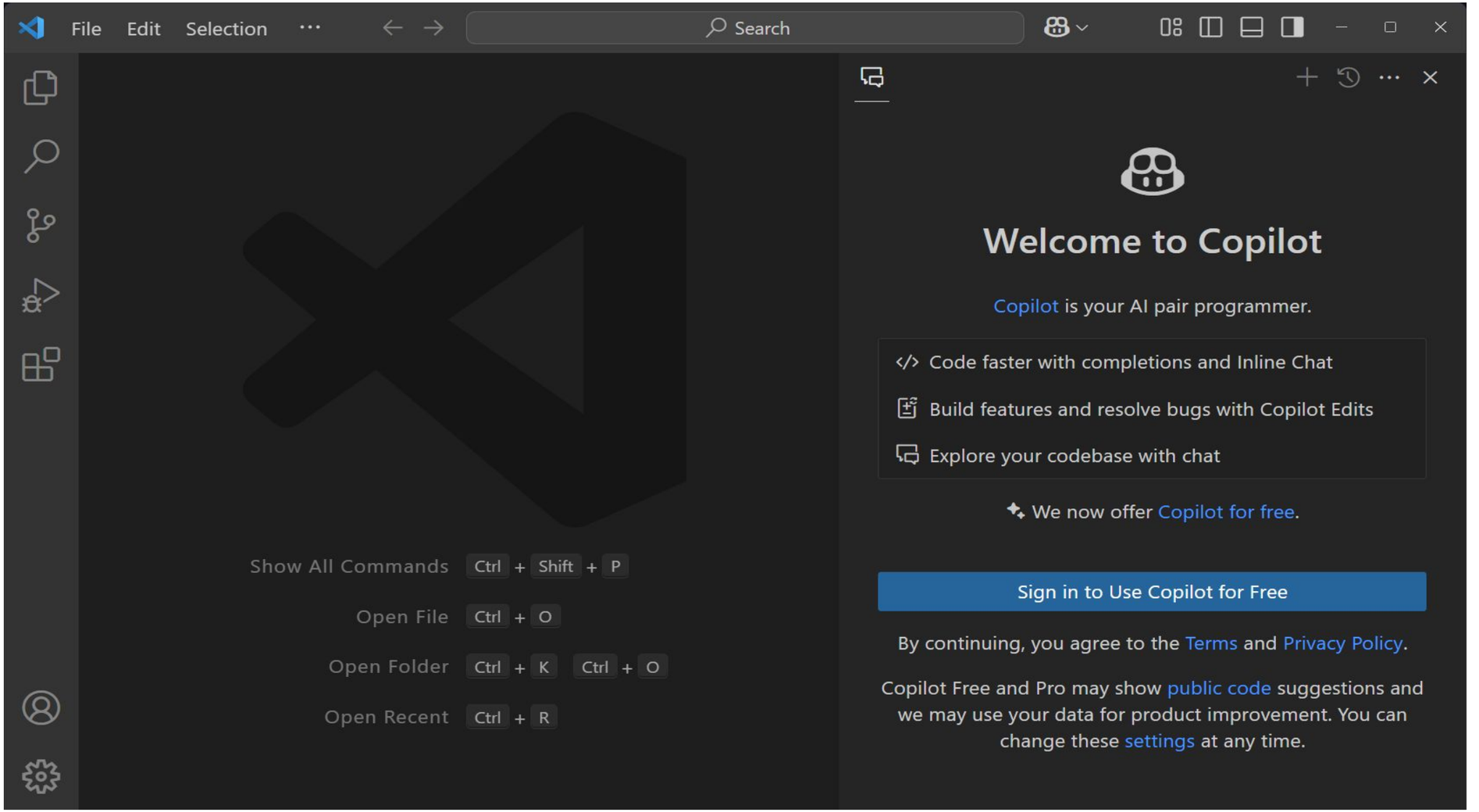
Set up Visual Studio Code with Copilot

3. Select Use AI Features with Copilot for Free... from the Copilot menu in the title bar or from the Command Palette (Ctrl+Shift+P)



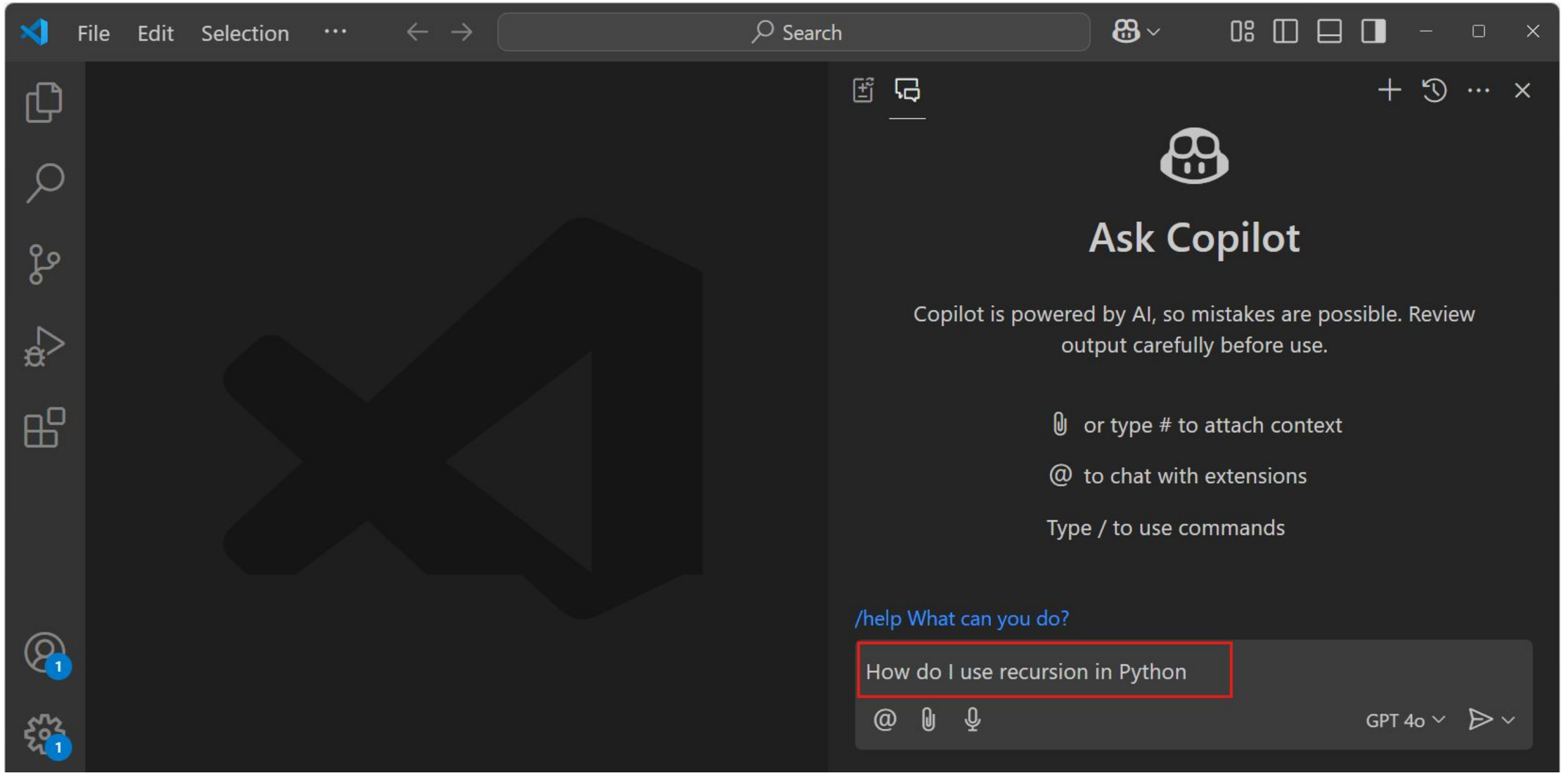
Set up Visual Studio Code with Copilot

4. Select Sign in to Use Copilot for Free to sign in to your GitHub account and sign up for Copilot Free
If you already have a Copilot subscription associated with your GitHub account, VS Code uses that one after you sign in.



Set up Visual Studio Code with Copilot

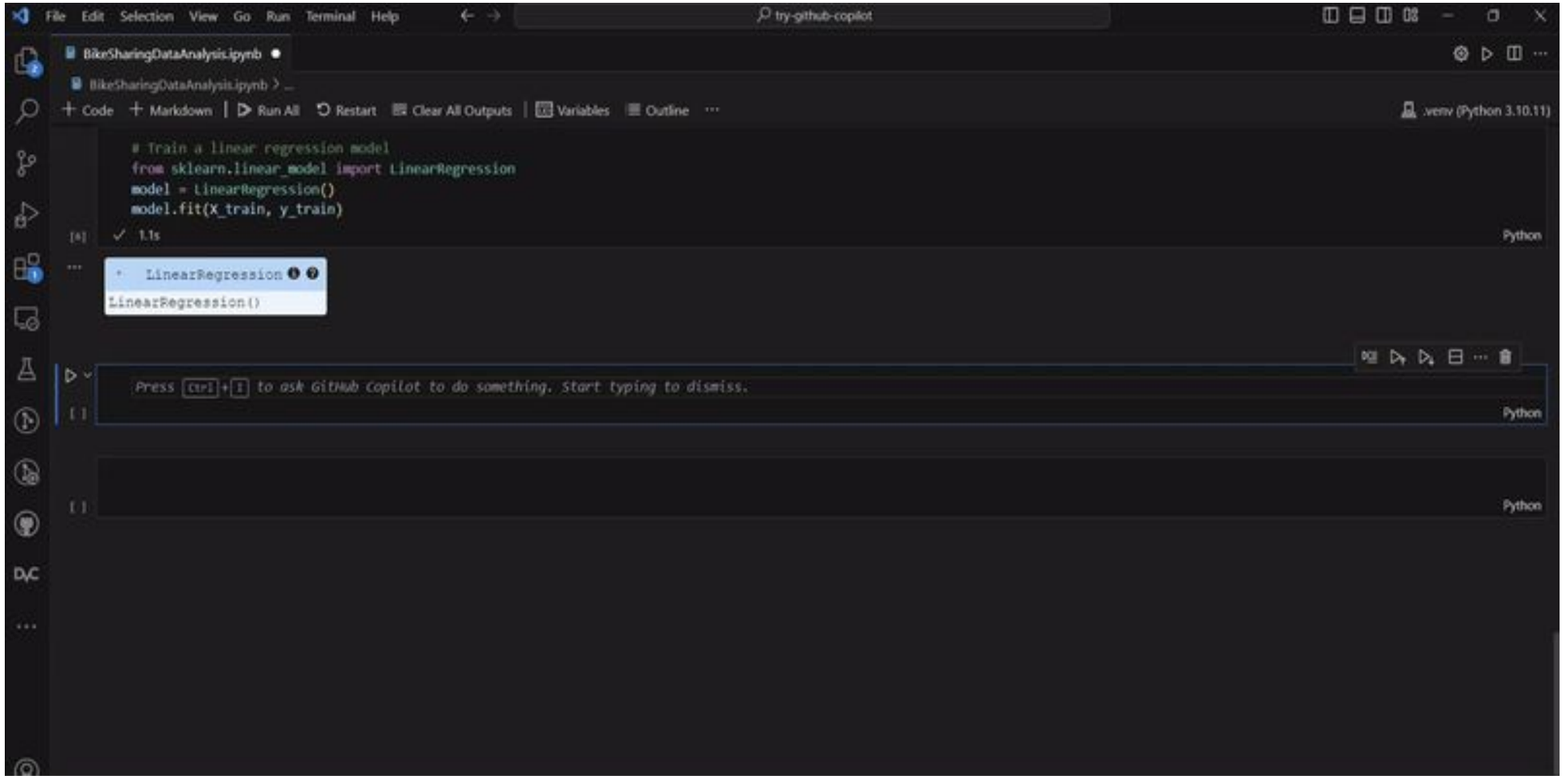
5. Get started by entering a prompt in the chat input field



Using GitHub Copilot method 1

Start **typing code** and accept suggestions with the Tab key

Writing code for evaluating a model with help of copilot



Using GitHub Copilot method 2 -

Use // and write your requirements Accept suggestions with the Tab key

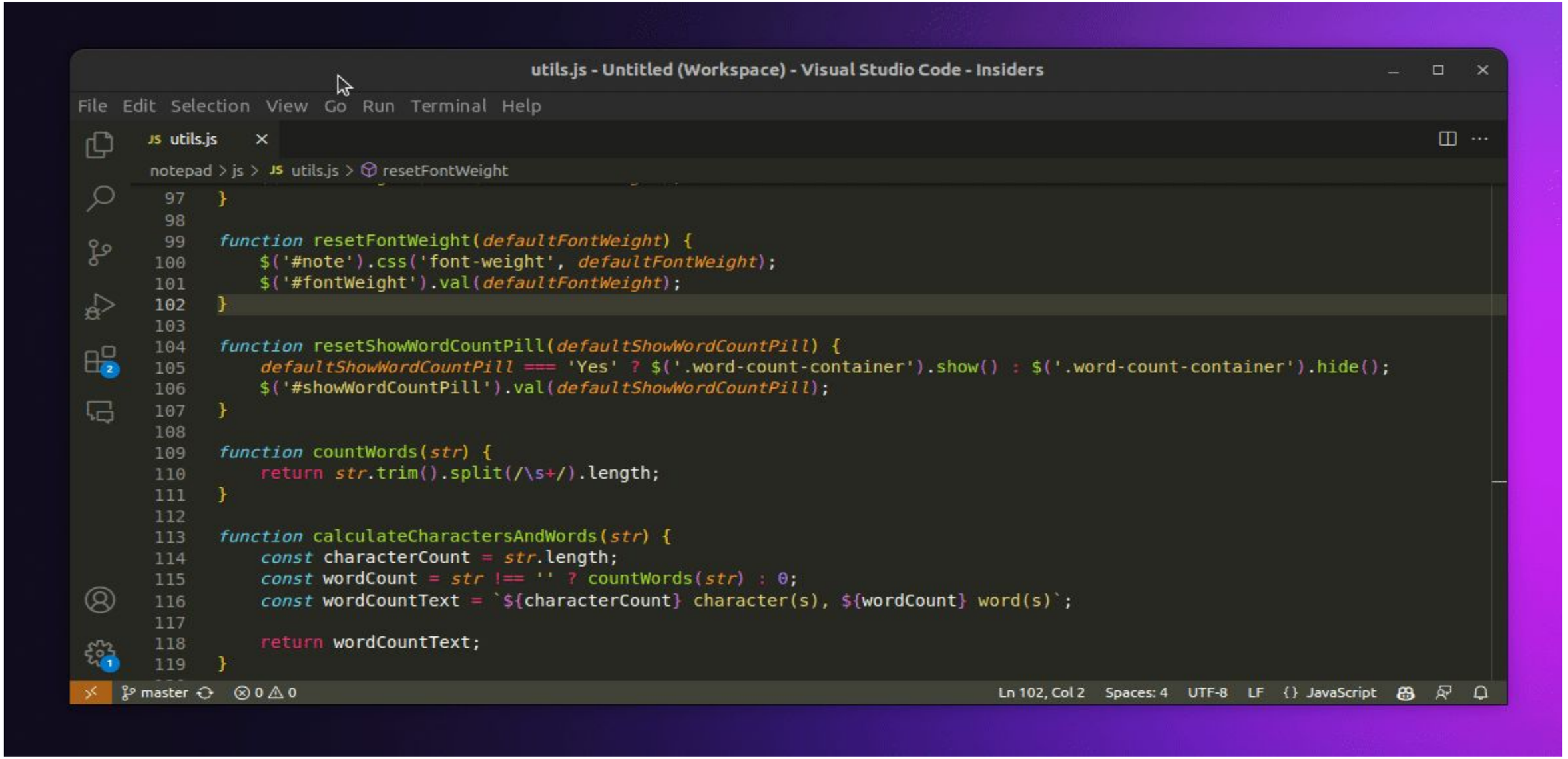
Generating a function that retrieves DB items by priority

```
0 references | 0 changes | 0 authors, 0 changes
public static void CreateTables()
{
    using (var context = new TaskContext())
    {
        context.Database.ExecuteSqlRaw("CREATE TABLE tasks (id INT PRIMARY KEY, title VARCHAR(50), priority INT)");
    }
}
```

Using GitHub Copilot method 3 -

Use **ctrl + I** and write your requirements and click on accept to accept changes

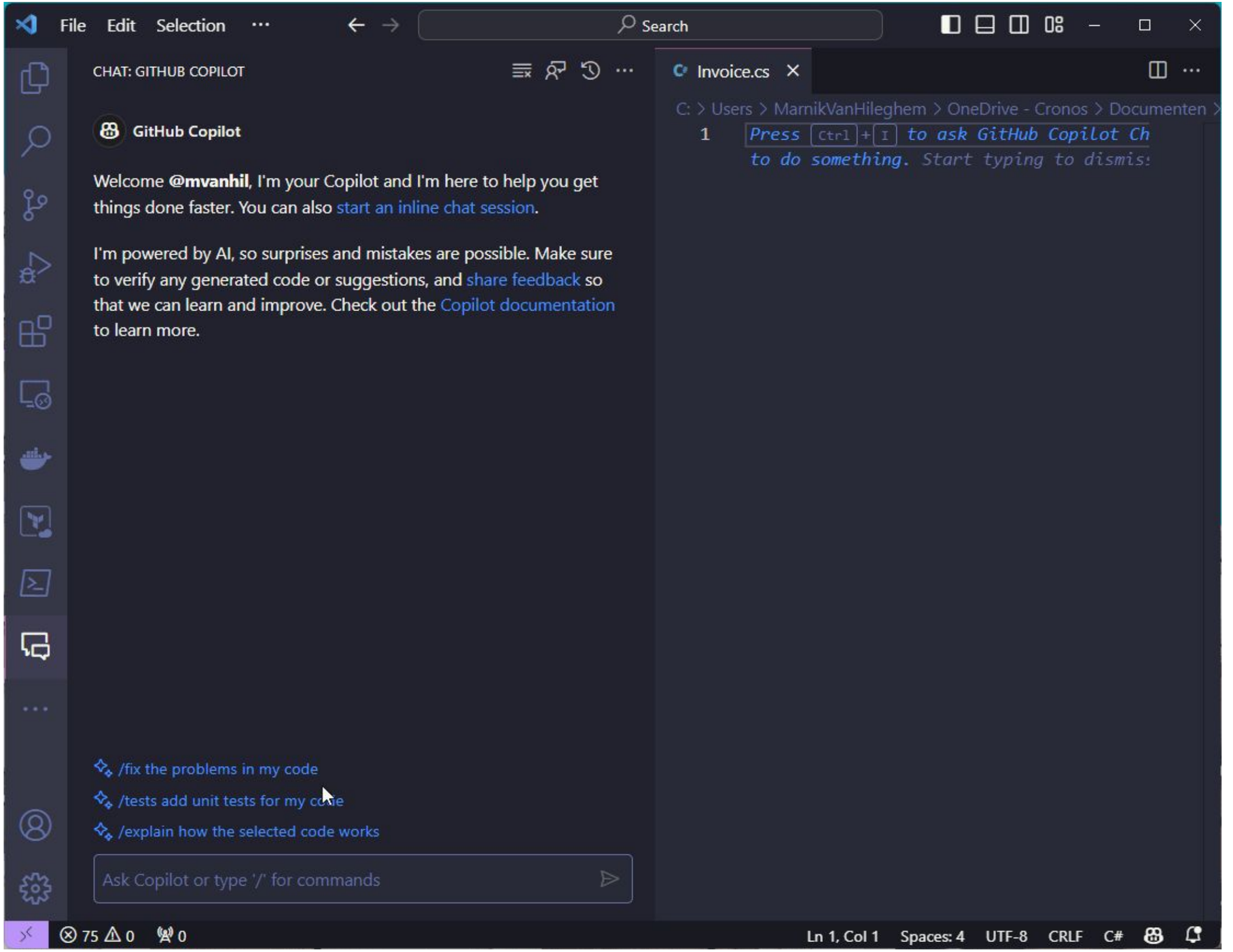
Asking copilot to make code readable in javascript



Using GitHub Copilot method 4 -

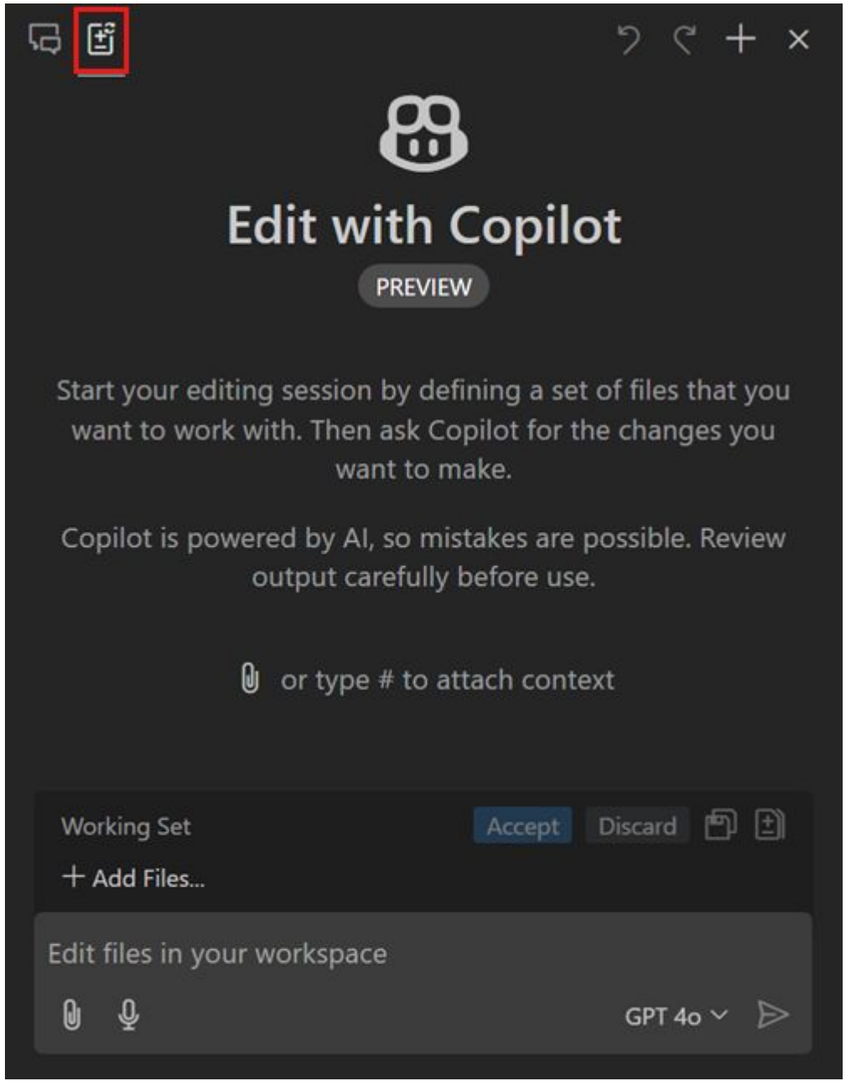
Use **Github copilot chat** to generate code and apply changes using apply to editor button

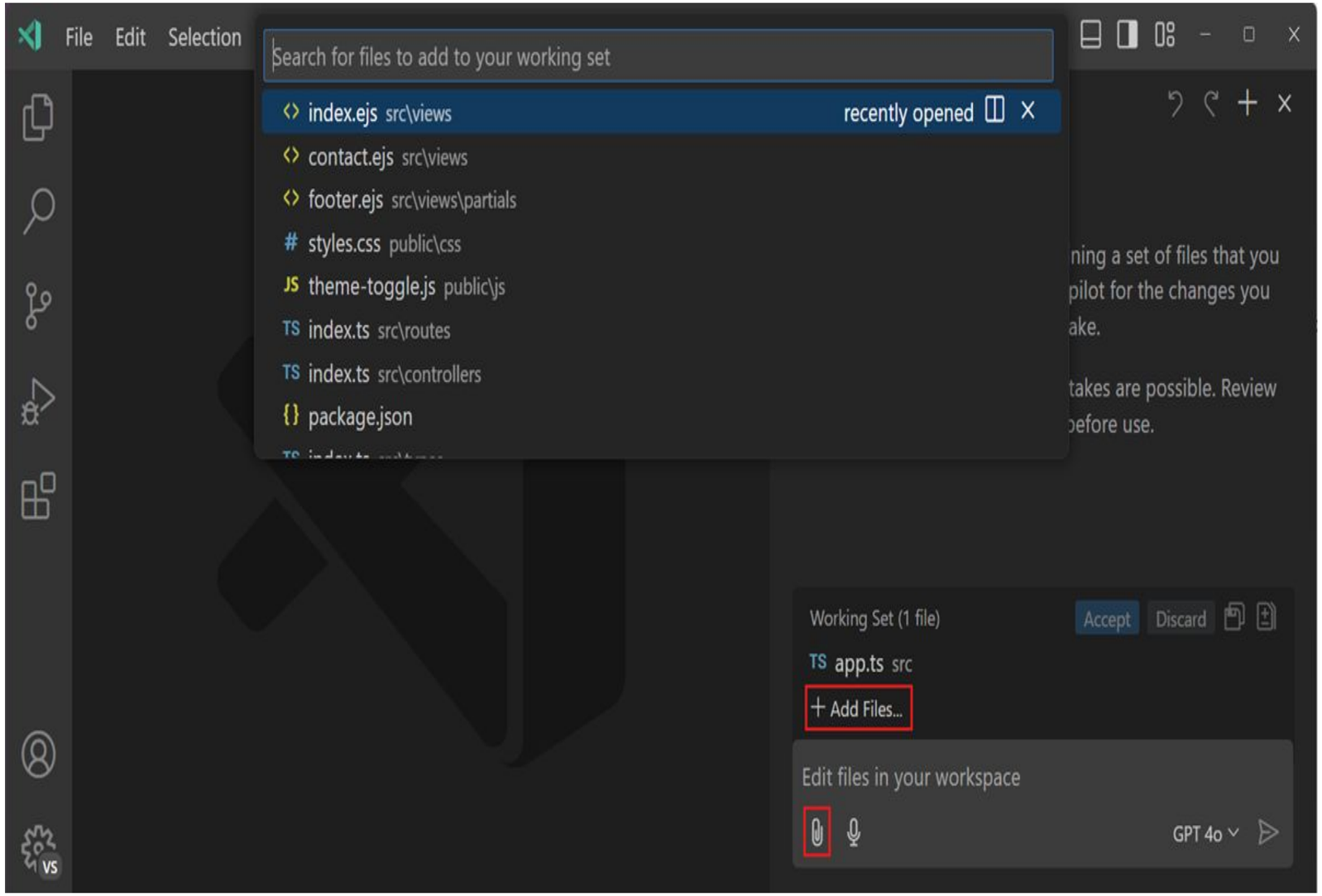
Asking copilot to create a realistic invoice model with line items in C-sharp



Using GitHub Copilot method 5 -

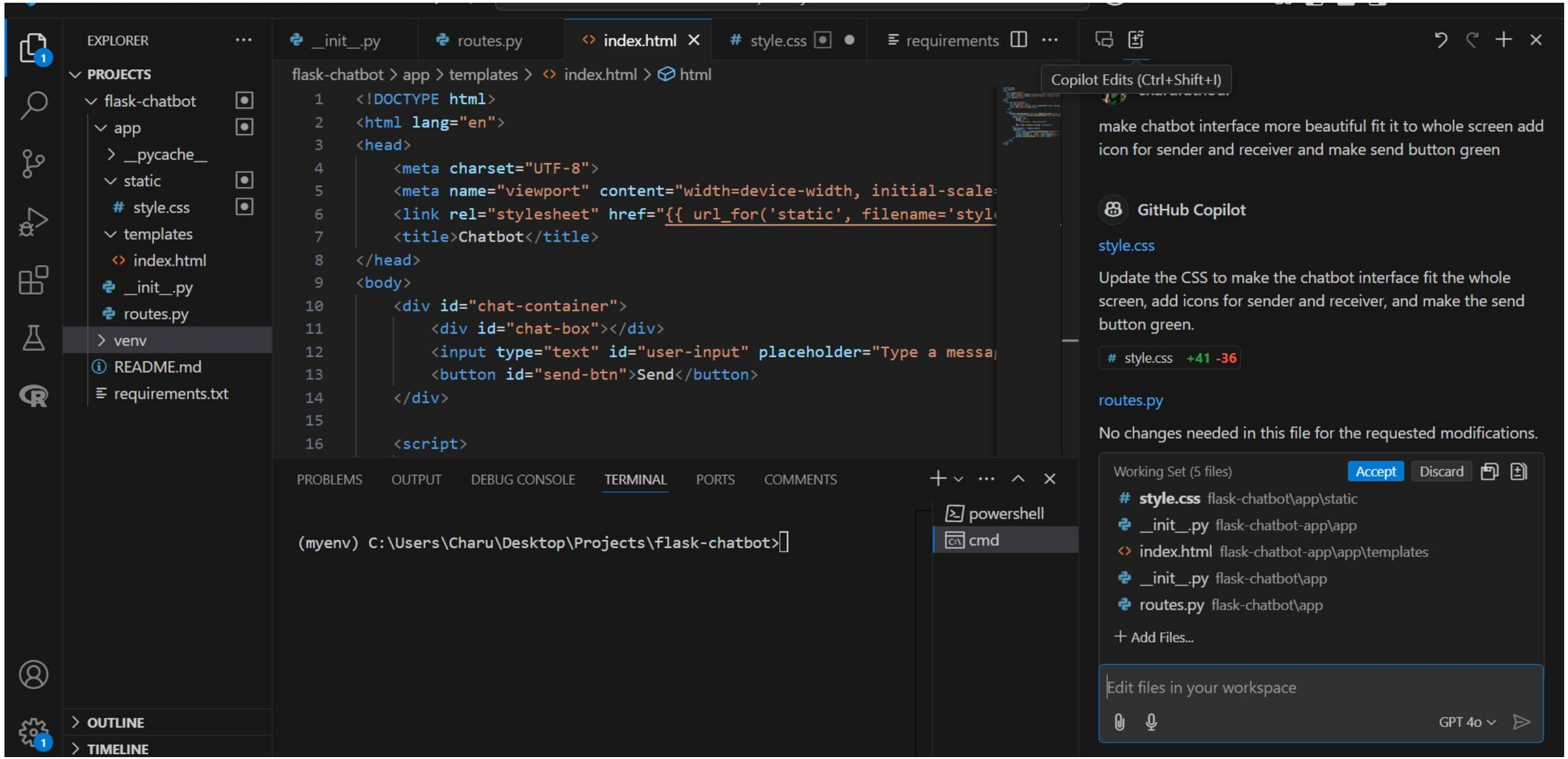
Using Copilot edits - Copilot Edits proposes code changes across multiple files in your workspace. These edits are applied directly in the editor, so you can quickly review them in-place, with the full context of the surrounding code.





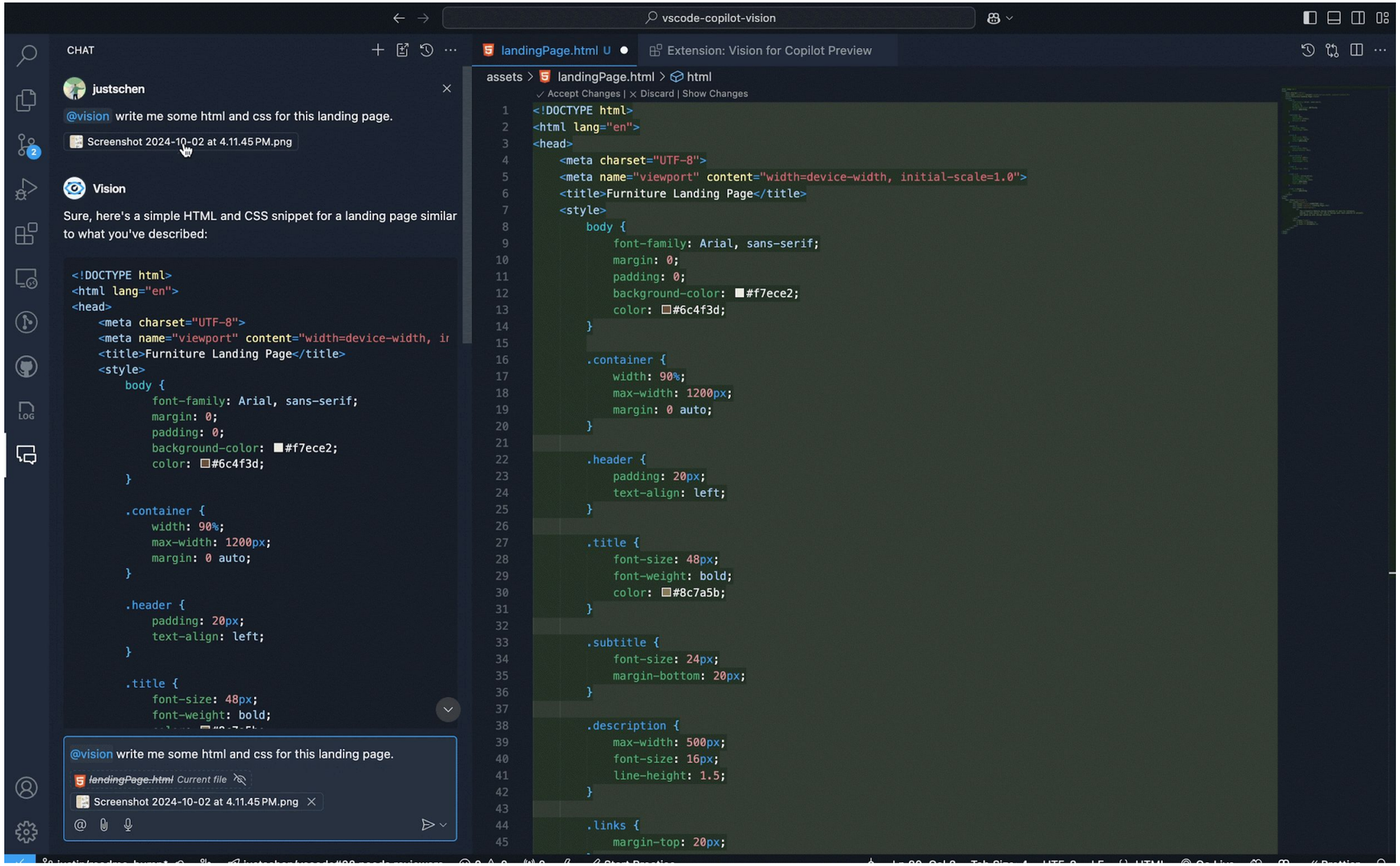
Using GitHub Copilot method 5 -

Using Copilot edits - Copilot Edits proposes code changes across multiple files in your workspace. These edits are applied directly in the editor, so you can quickly review them in-place, with the full context of the surrounding code.

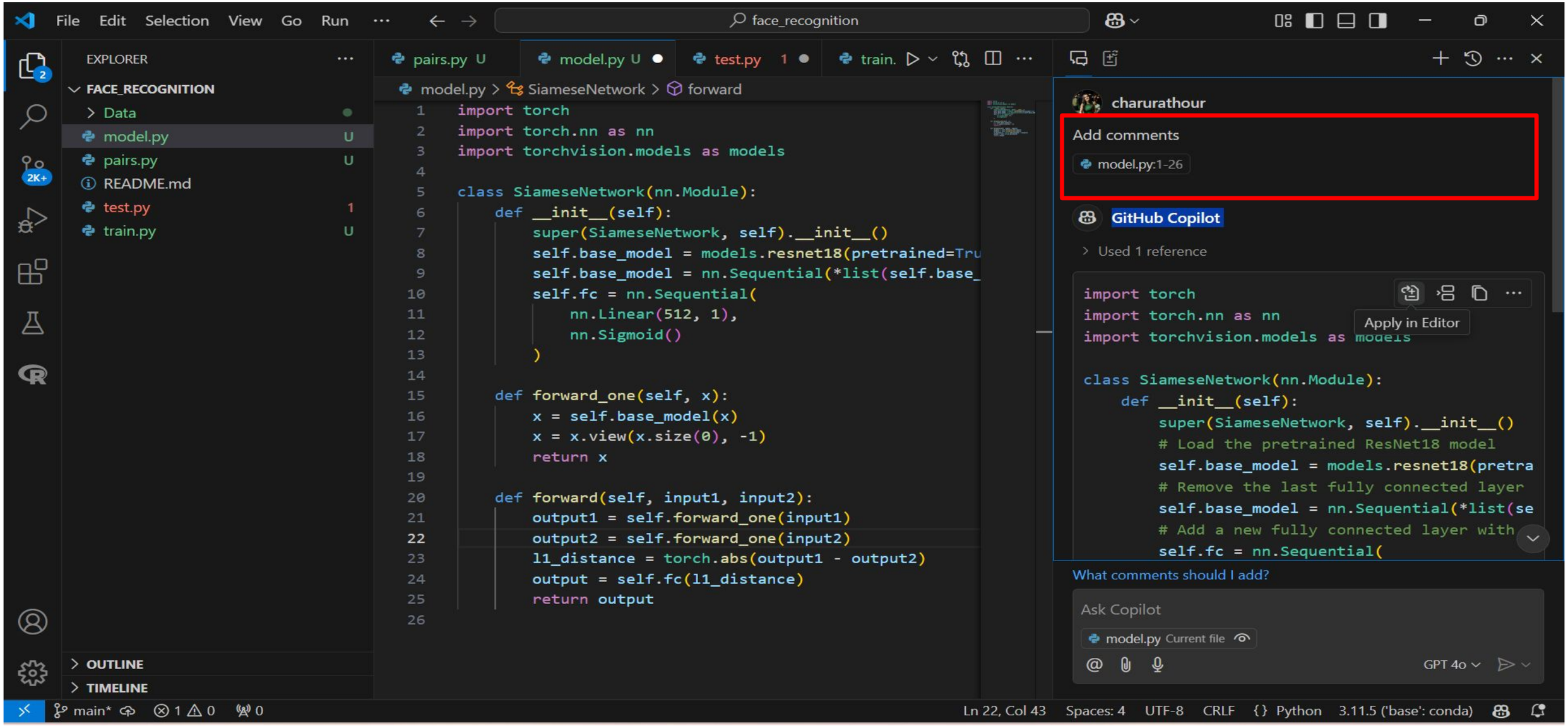


Added Extensions - Vision for Copilot

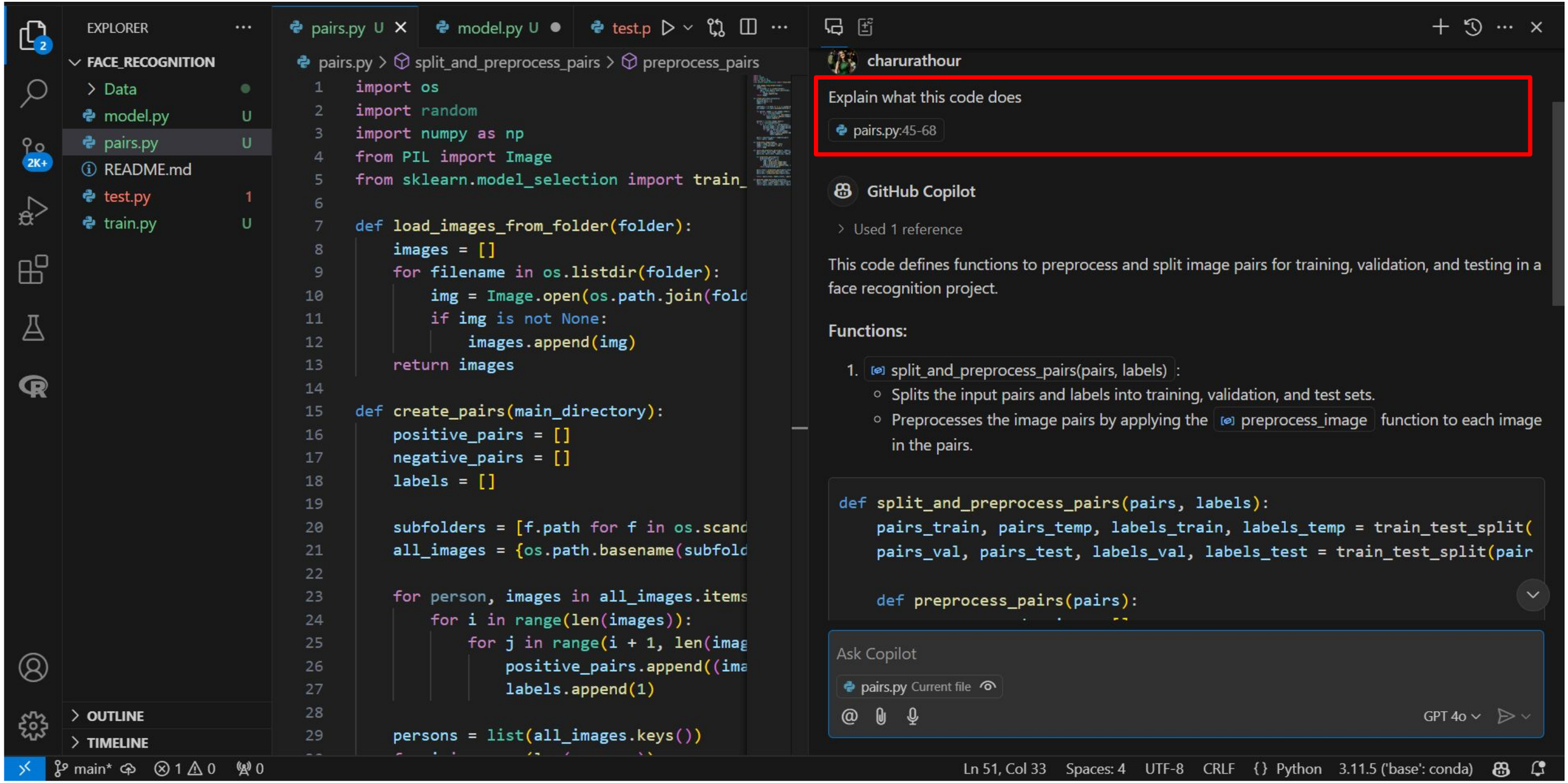
Vision for Copilot is an extension that **enhances chat by allowing users to attach images as contextual input,** enabling dynamic, visually-supported responses for richer interactions.



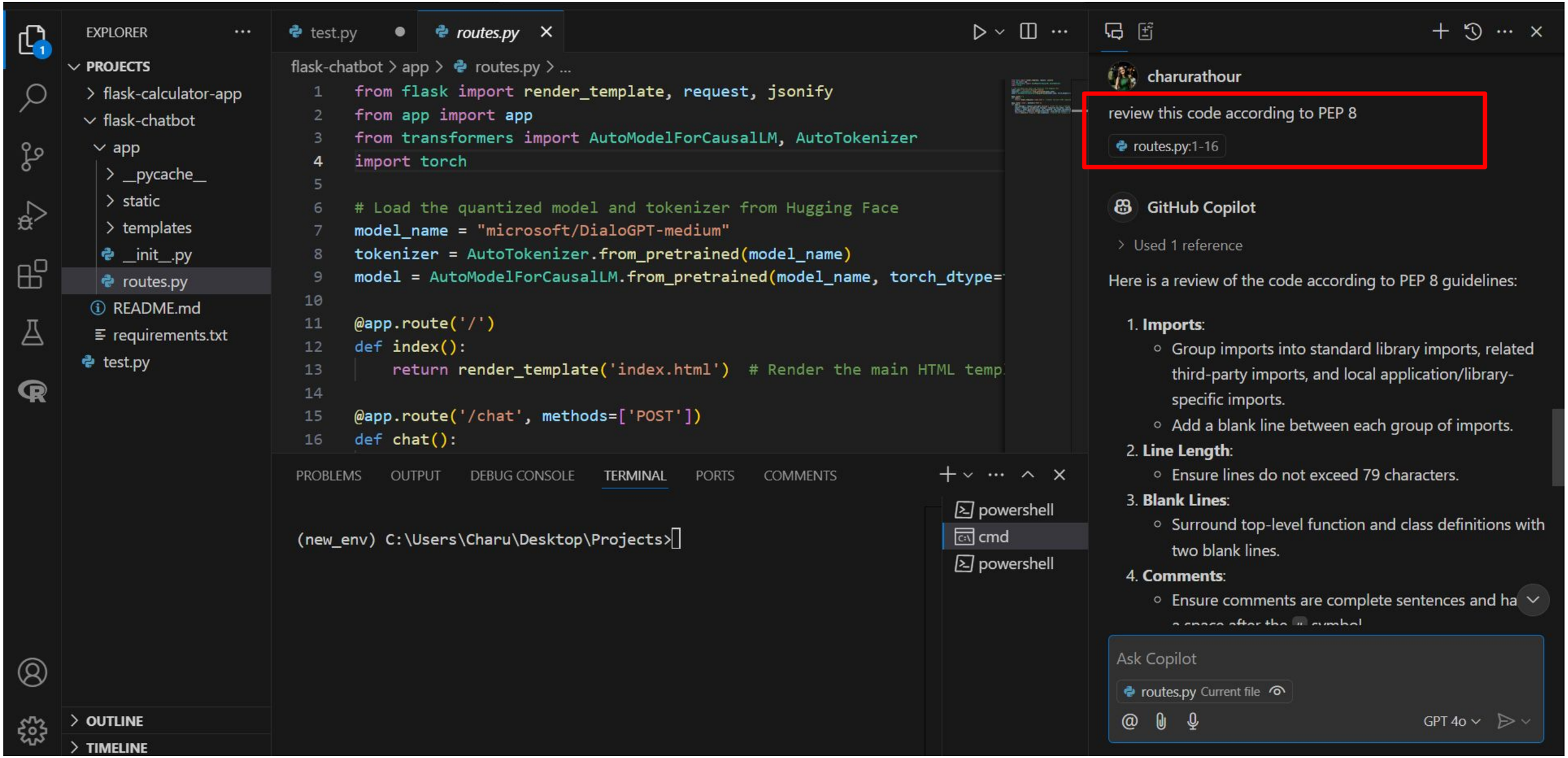
Application 1 - Adding Comments



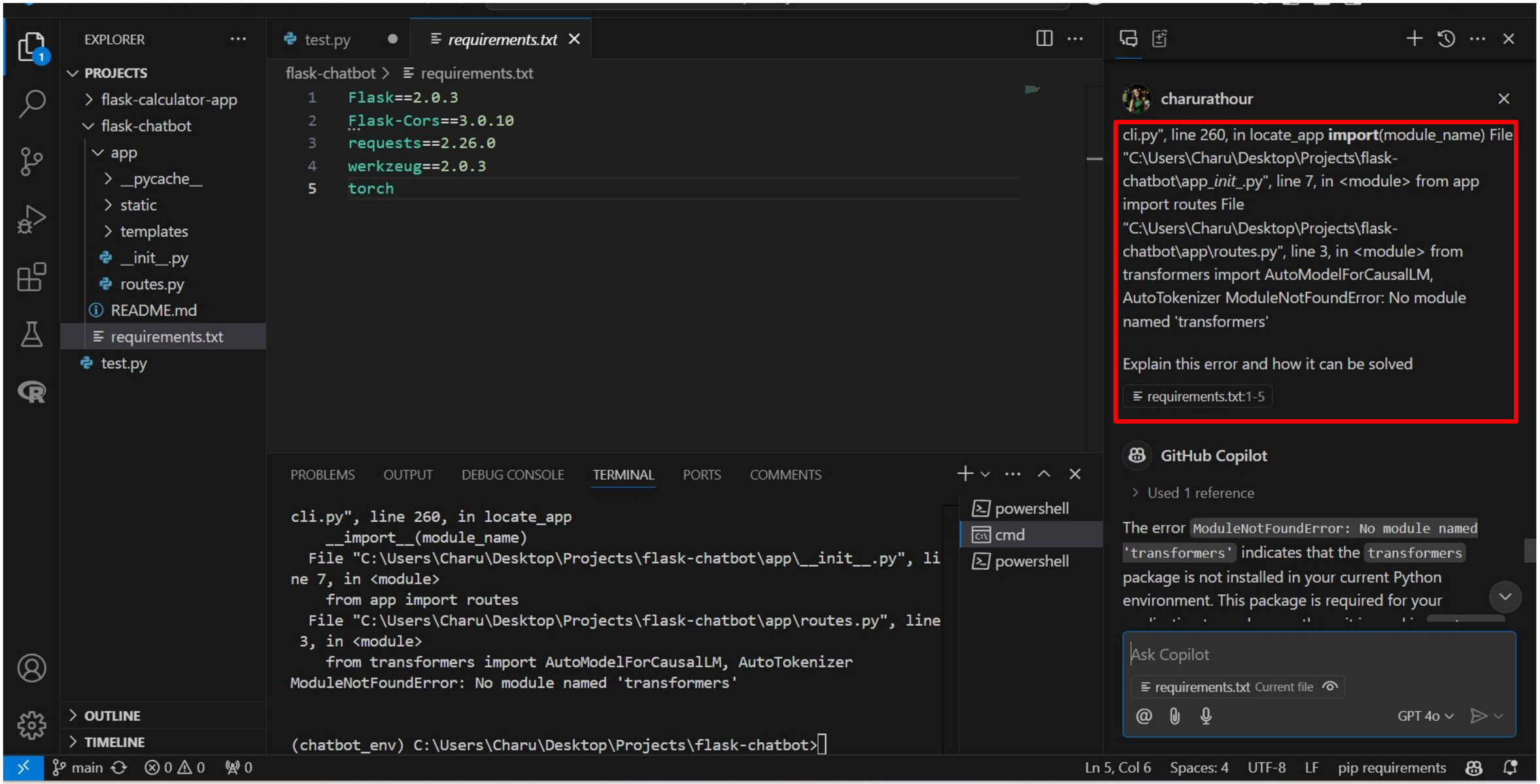
Application 2 – Generating Explanation of Existing Code



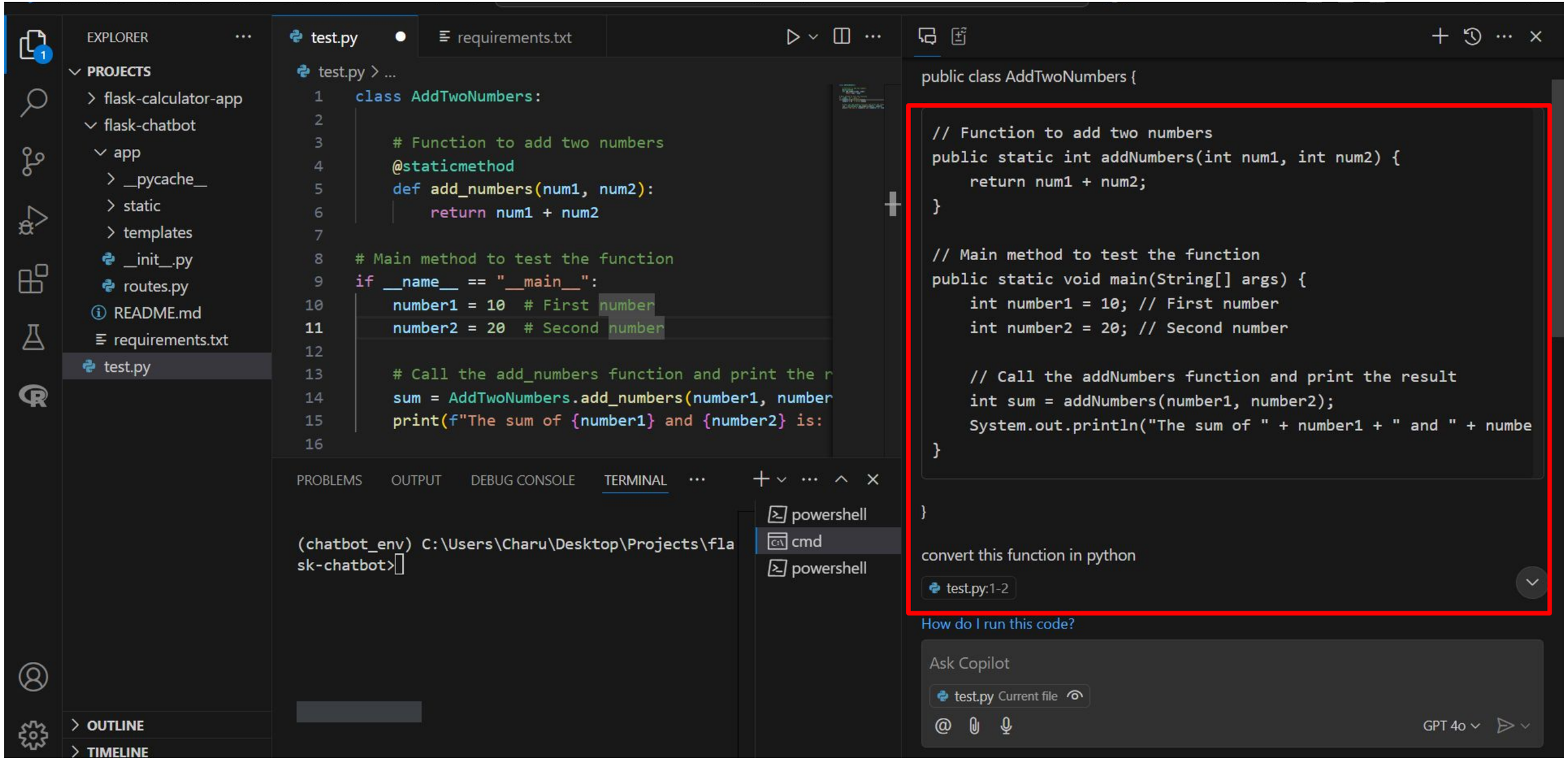
Application 3 - Code Review



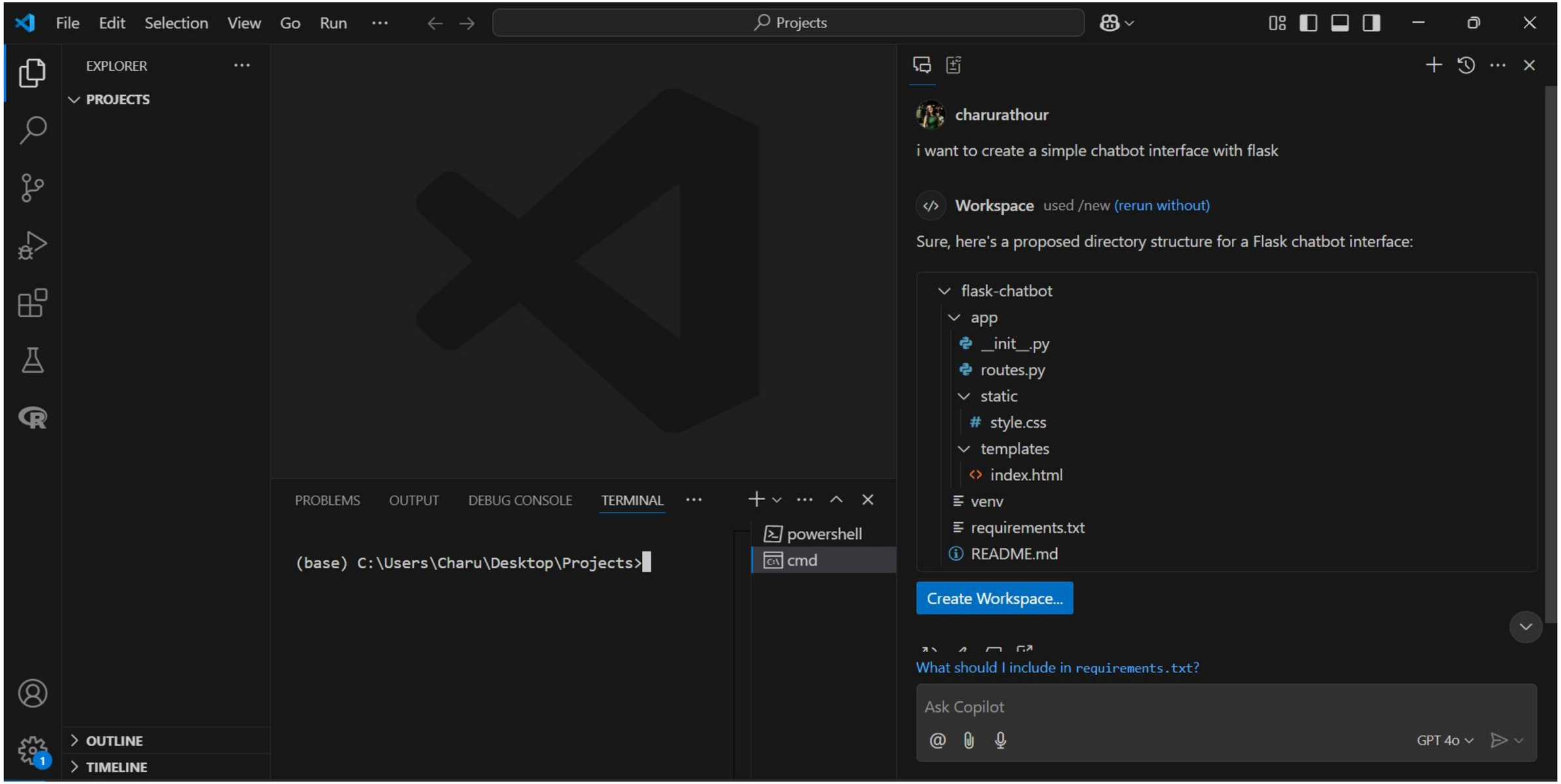
Application 4 – Debugging



Application 5 – Language Conversion



Application 6 – Creation of Workspace with Copilot



Use Case – Building a Chatbot Application with GitHub Copilot

Step 1 : Setting Goals and Structuring Your Workflow

Set Your Goal

- Define what you want to build.
- Example: "I want to create a chatbot for answering basic user queries."
- Be specific about the problem your application will solve.

Identify Requirements

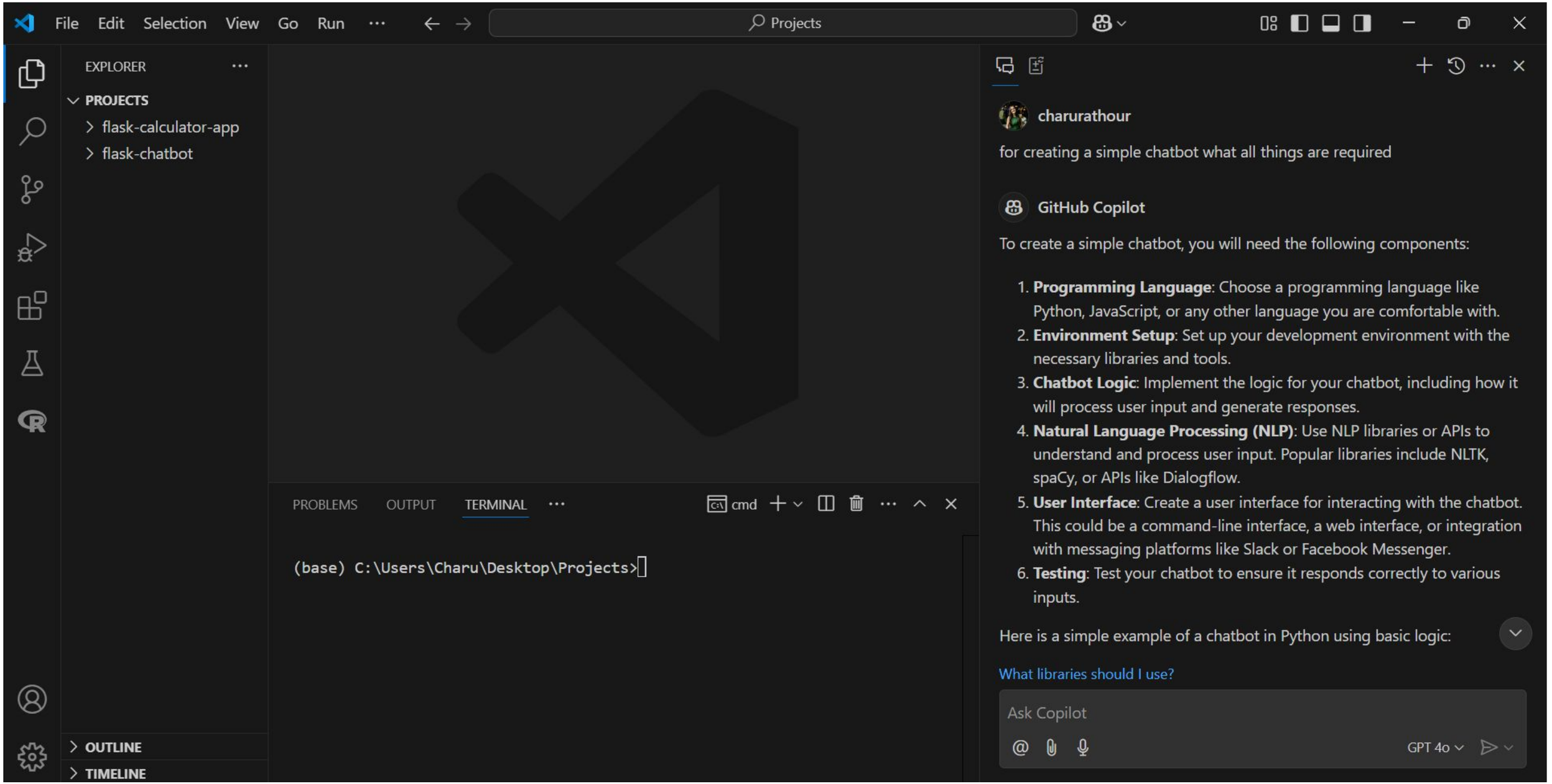
- What does your application need?
- Input: What will the user provide?
- Example: Text queries in a chat window.
- Output: What will the app deliver?
- Example: AI-generated text responses.

Decide on the Tech Stack

- **Language**: Choose the programming language.
  Example: Python for backend and AI models.
- **Libraries and Frameworks**: List what you'll need.
  Example: Flask for backend, Transformers from Hugging Face for chatbot logic, HTML/JavaScript for frontend.
- **Backend**: Will you use Flask/Django/Node.js, or something else?
- **Frontend**: Web-based interface (HTML/JS) or other frameworks like React.

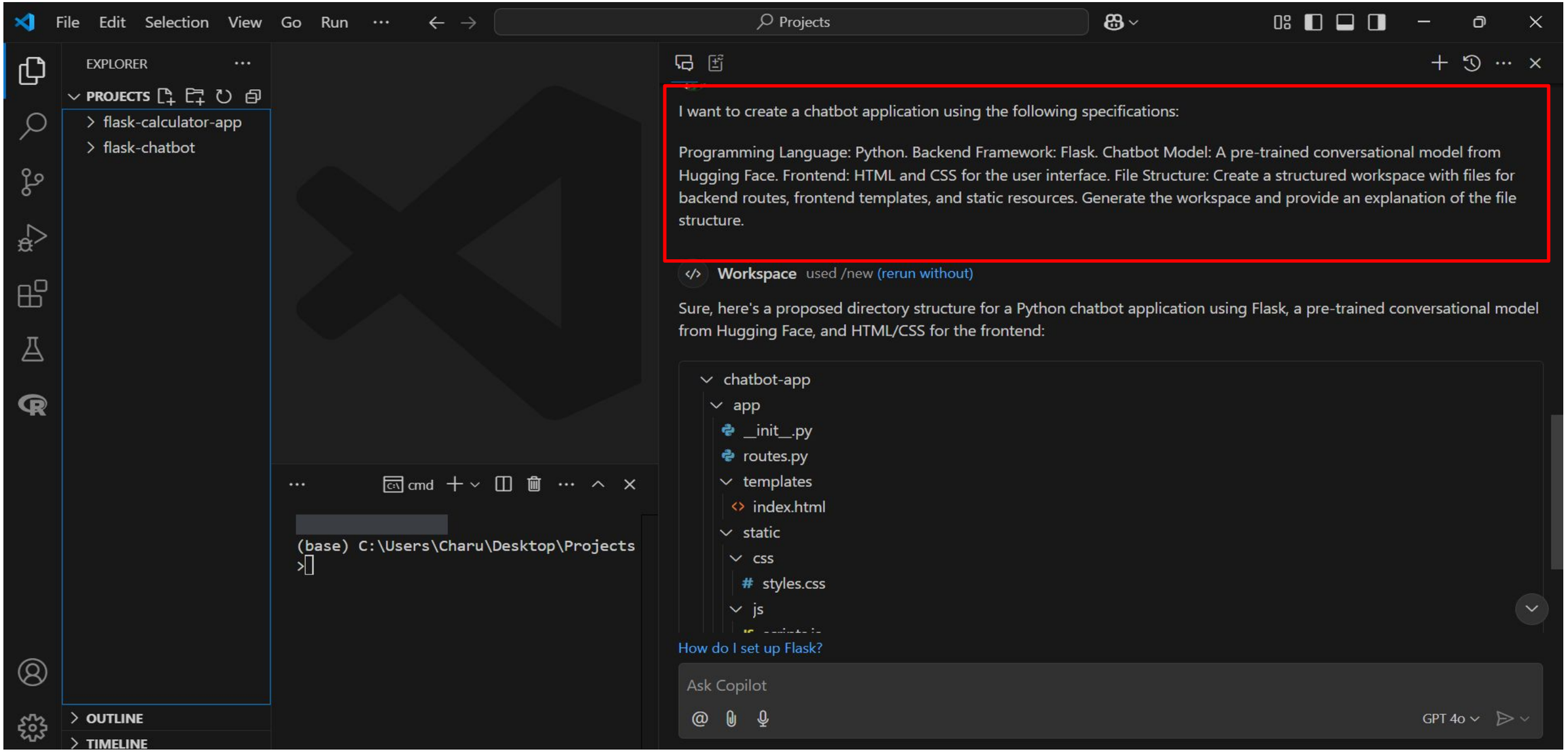
You can use Copilot chat to research and ideate for setting up goal

Step 1 : Setting Goals and Structuring Your Workflow



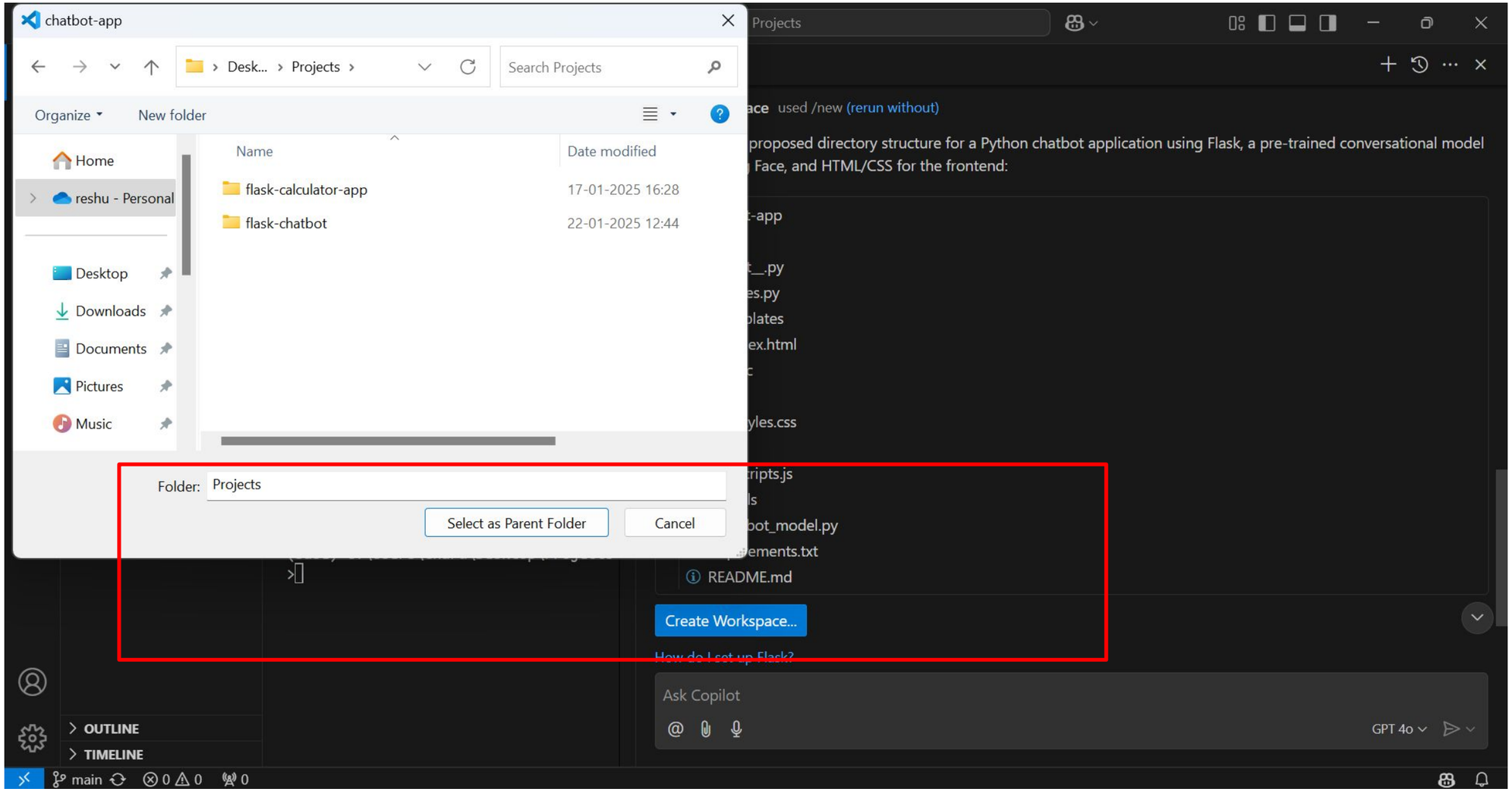
Use Case – Building a Chatbot Application with GitHub Copilot

Step 2 : Use Copilot to create workspace by defining your goal and file structure in copilot chat area



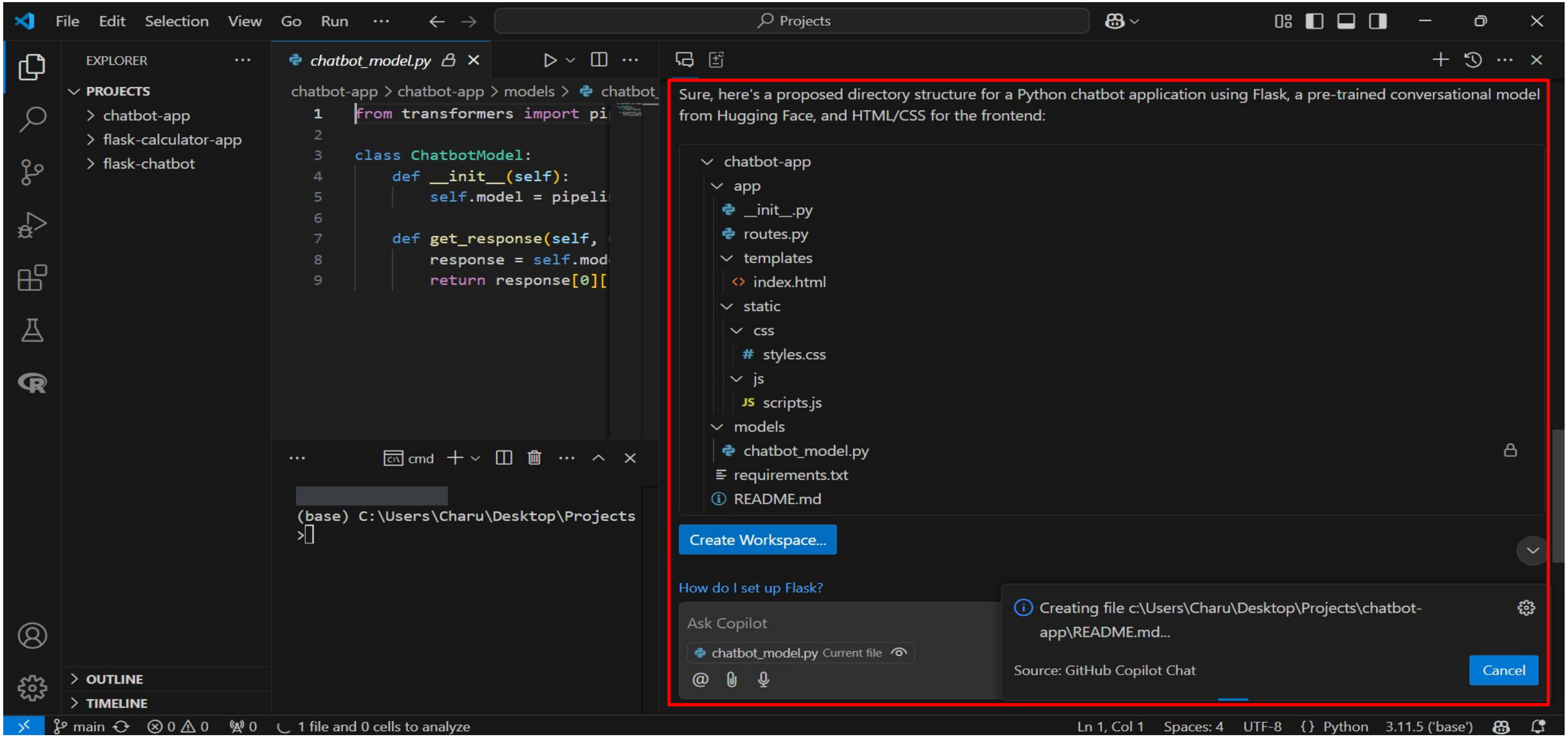
Use Case – Building a Chatbot Application with GitHub Copilot

Step 2 : Use Copilot to create workspace by defining your goal and file structure in copilot chat area



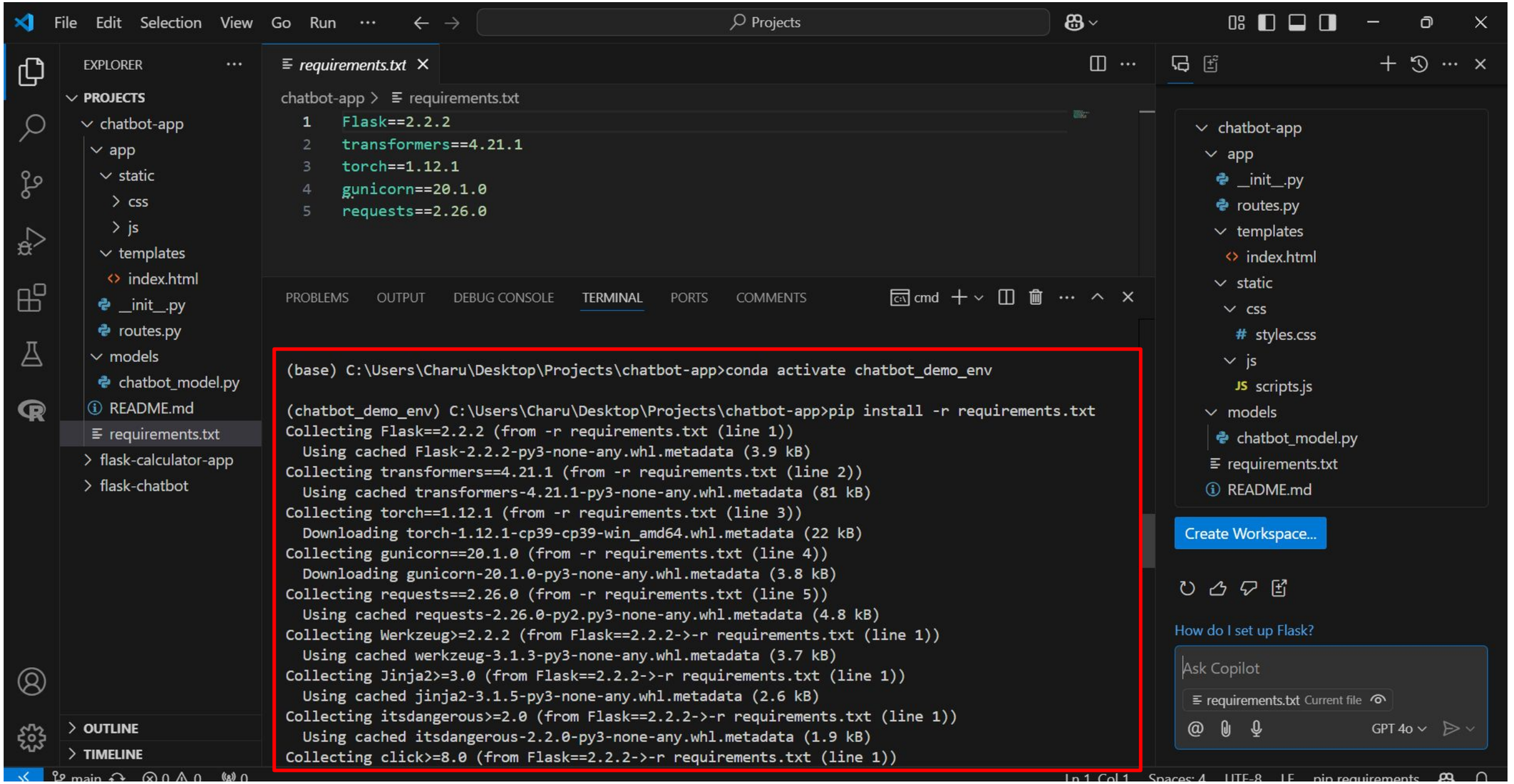
Use Case – Building a Chatbot Application with GitHub Copilot

Step 2 : Use Copilot to create workspace by defining your goal and file structure in copilot chat area



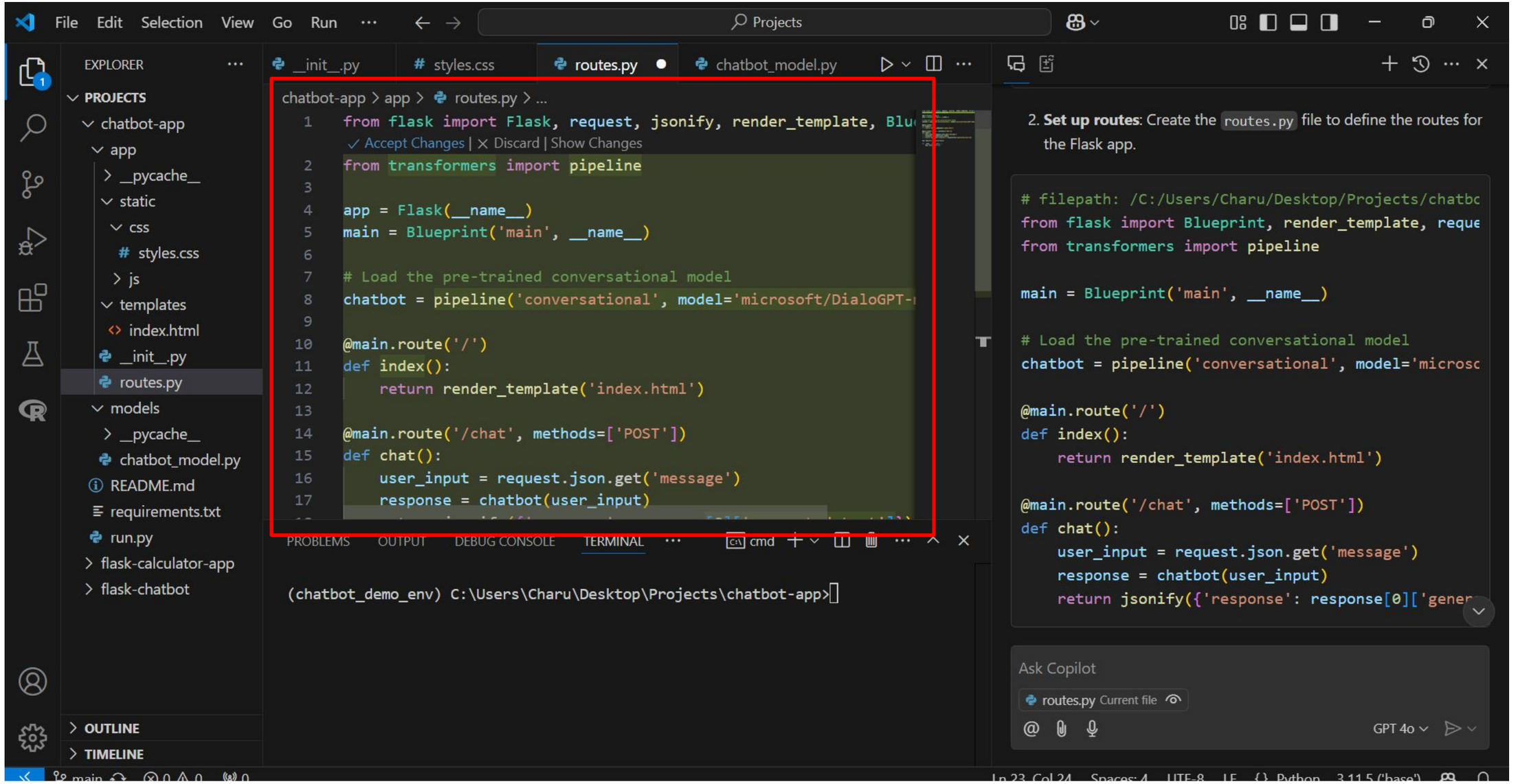
Use Case – Building a Chatbot Application with GitHub Copilot

Step 3 : Create Environment and install dependencies from requirements file generated by copilot



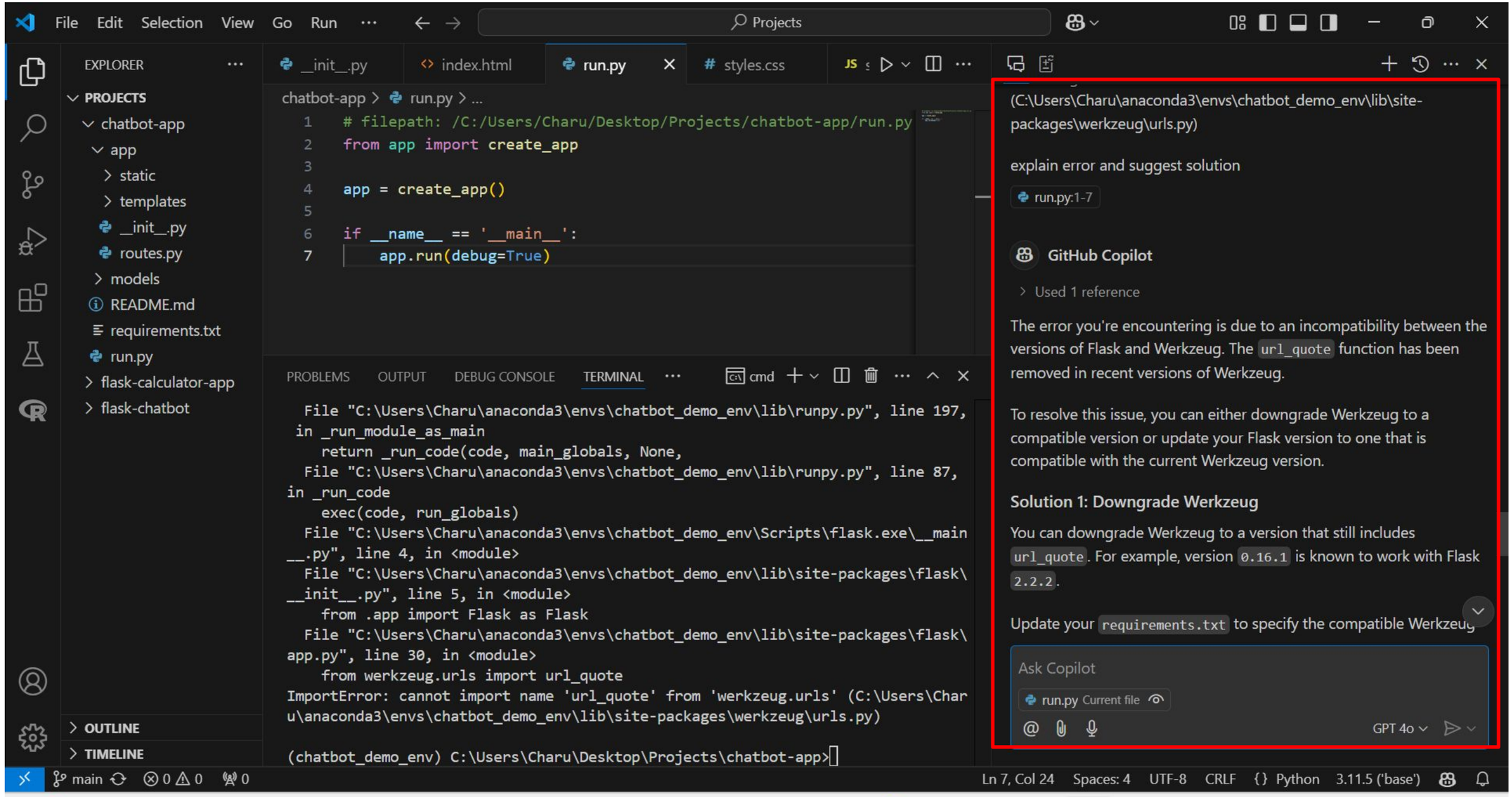
Use Case – Building a Chatbot Application with GitHub Copilot

Step 4 : Review code and make required changes



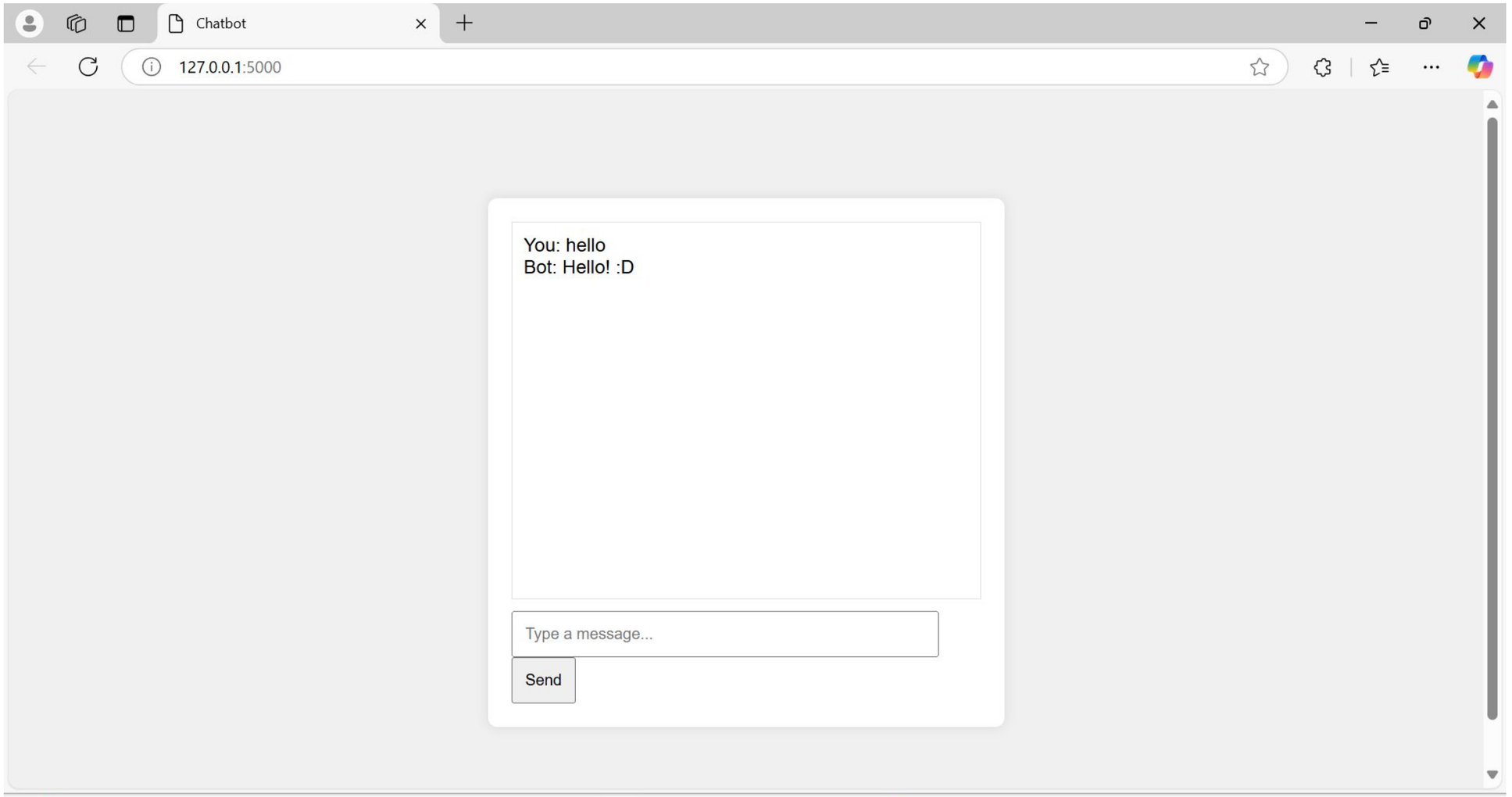
Use Case – Building a Chatbot Application with GitHub Copilot

Step 5 : Run code and Debug with help of copilot if any errors come



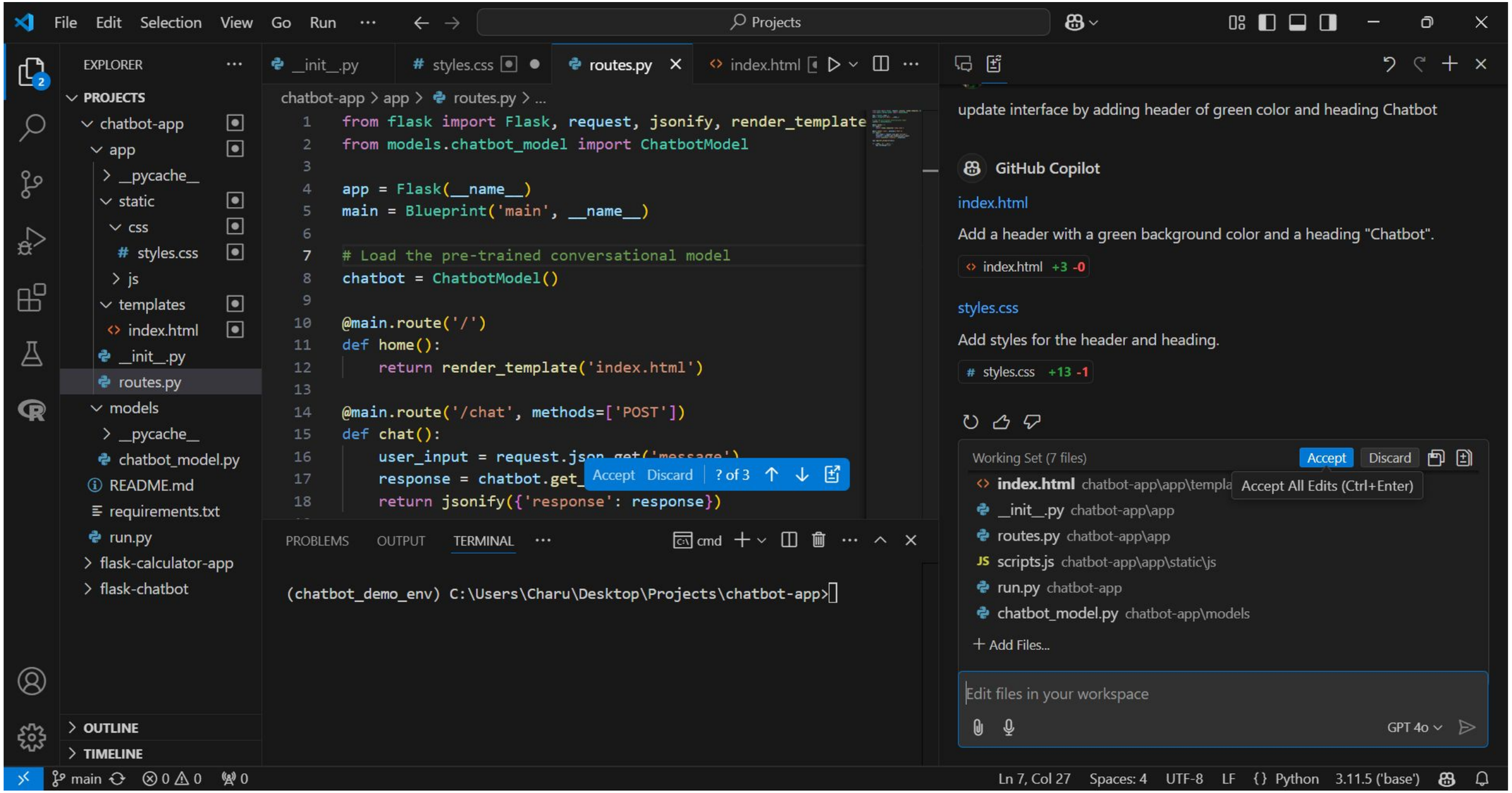
Use Case – Building a Chatbot Application with GitHub Copilot

Step 6 : Your basic chatbot is ready now you can iterate and make changes and update with copilot



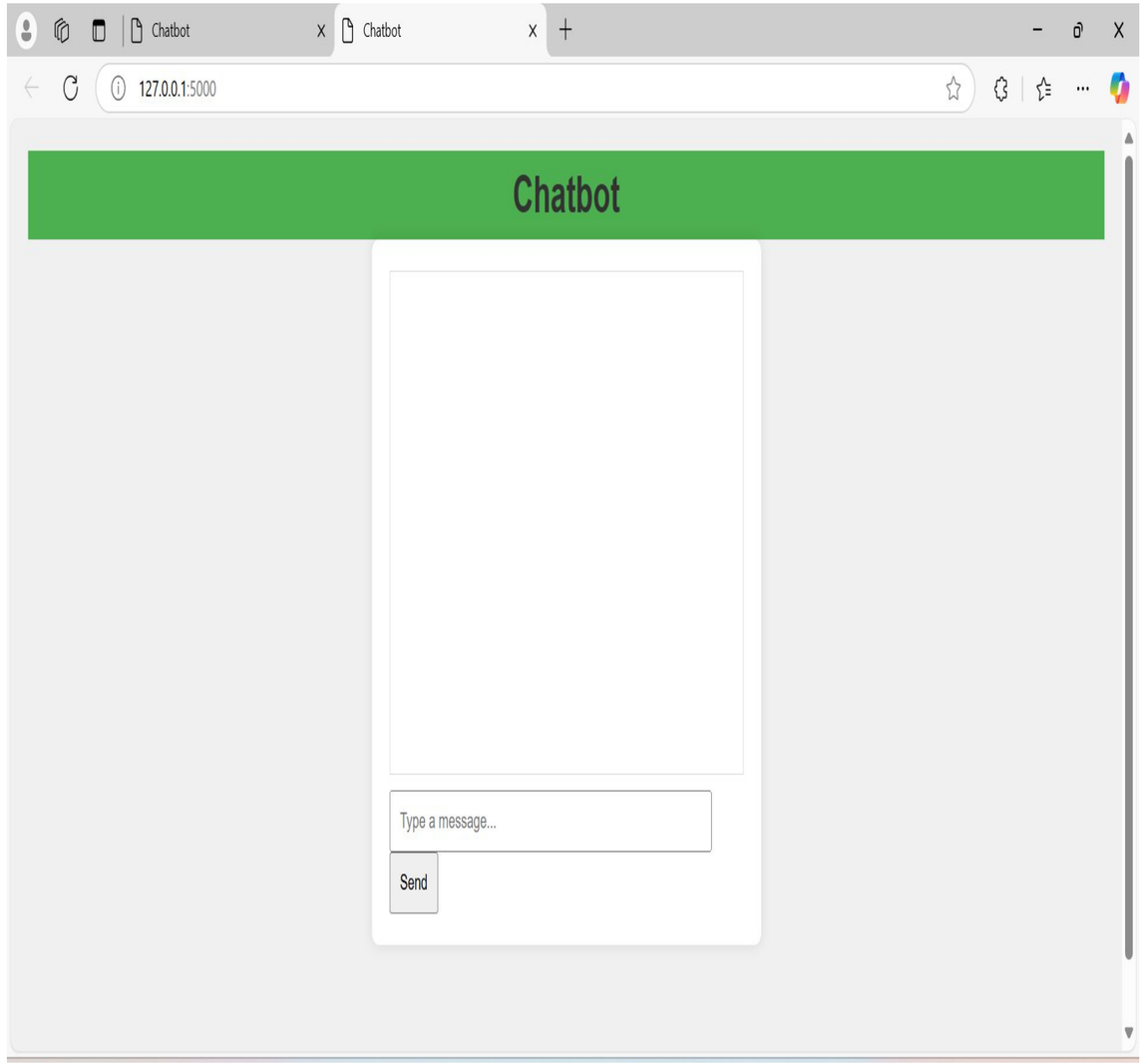
Use Case – Building a Chatbot Application with GitHub Copilot

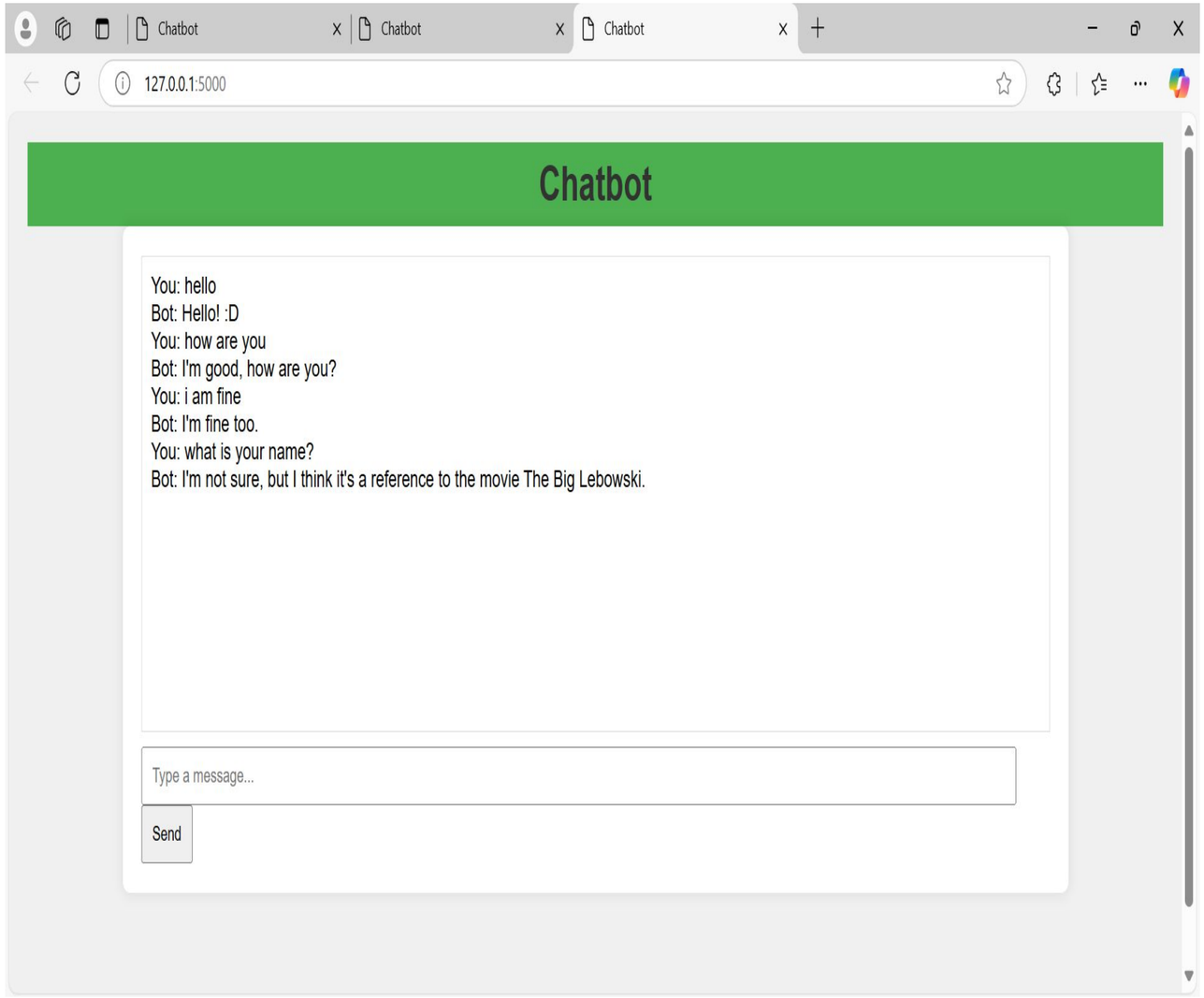
Step 6 : Iterate and make changes and upgrade your application with copilot



Use Case – Building a Chatbot Application with GitHub Copilot

Step 6 : Iterate and make changes and update with copilot and update your code to Github side by side

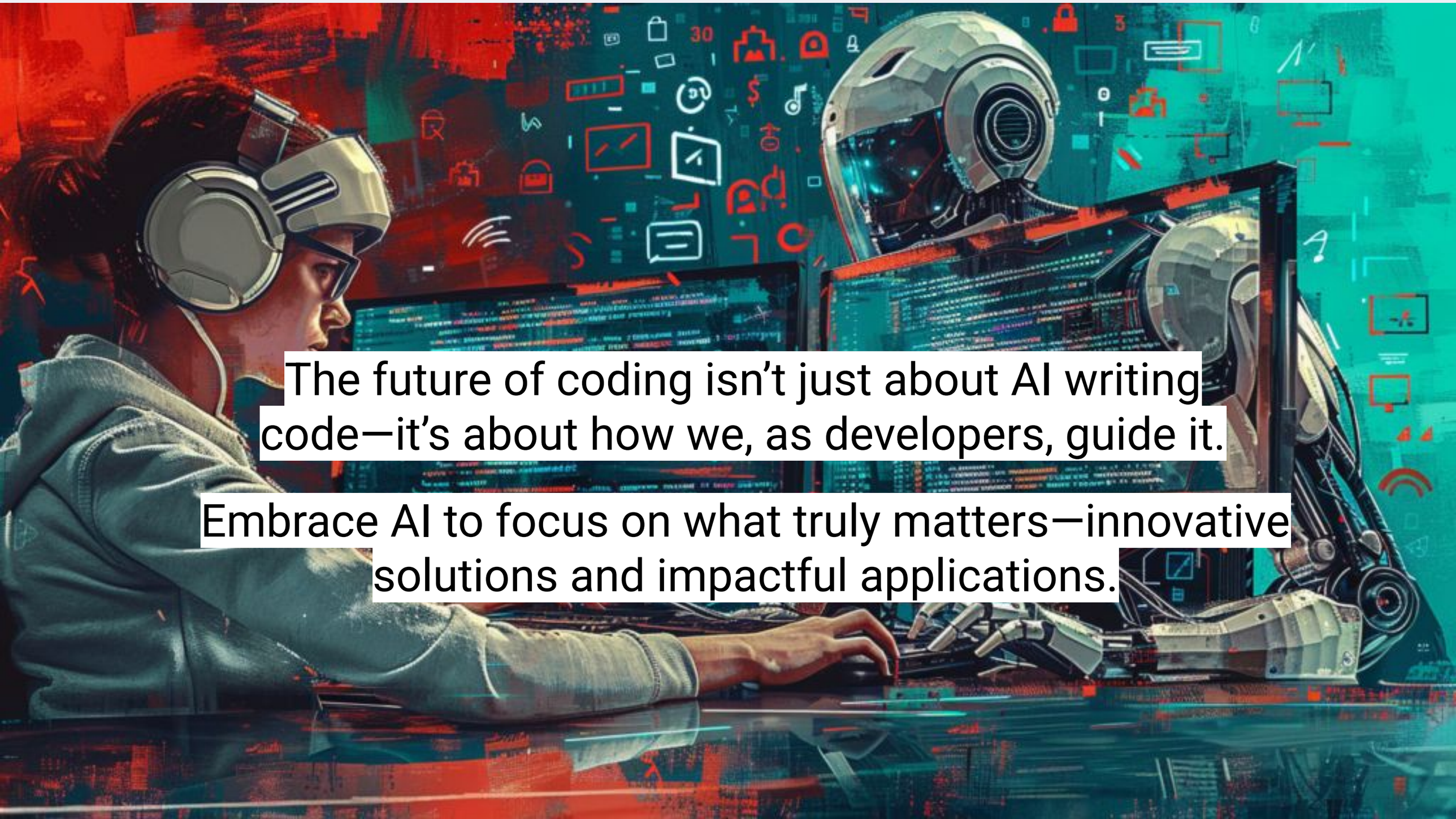


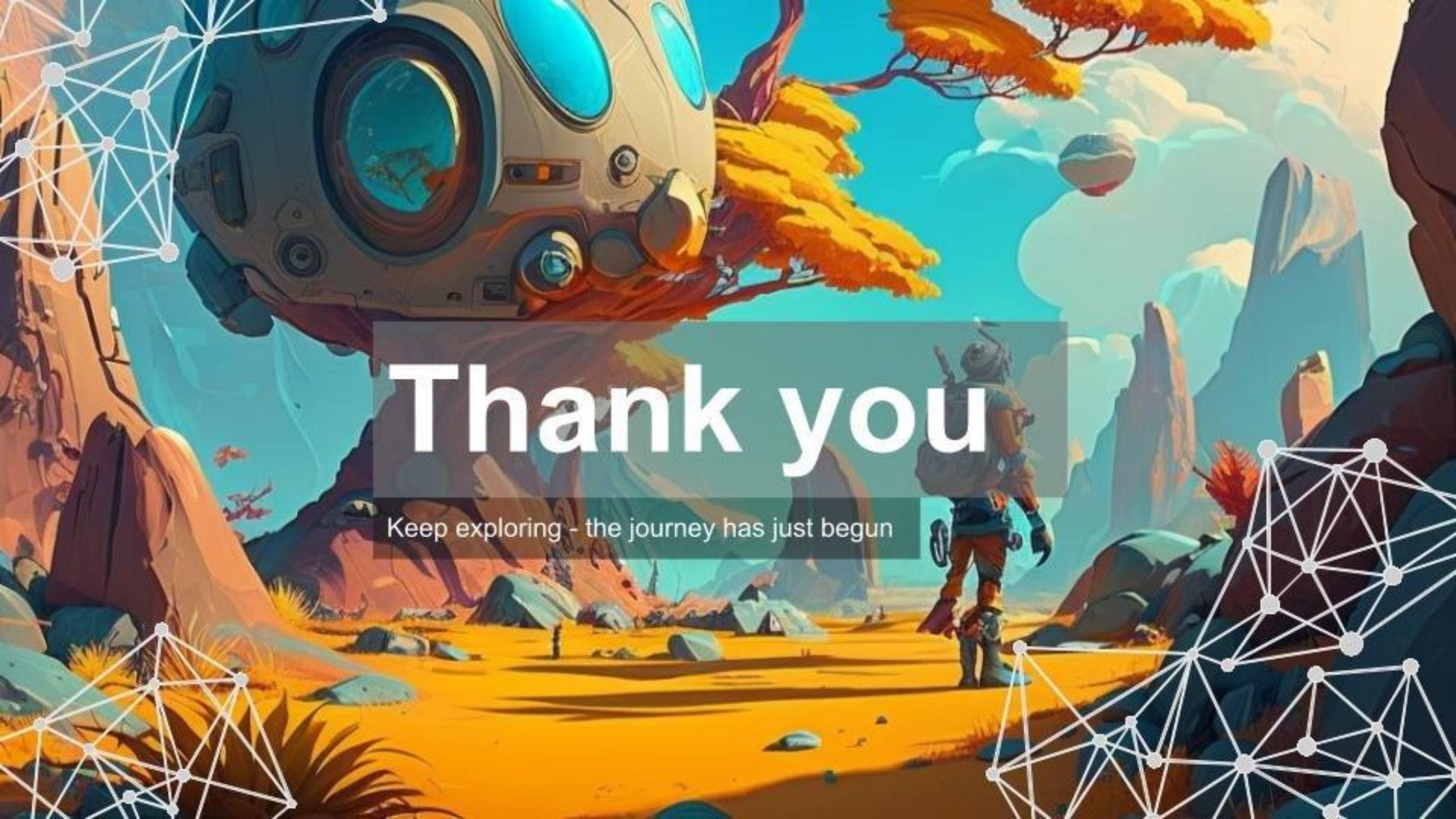


An application which might have taken 1-2 Hrs for development can be developed in half an hour with help of AI

Conclusion: AI Empowering Developers

- **Time-Saving**: AI automates repetitive tasks like code generation, debugging, and documentation, reducing development time significantly.
- **Focus on Creativity**: With routine tasks handled by AI, developers can dedicate more time to innovative and complex problem-solving.
- **Enhanced Productivity**: By streamlining workflows, AI helps developers build faster, more reliable, and scalable software solutions.
- **Future of Coding**: AI tools are transforming development into a more creative and efficient process.


The future of coding isn’t just about AI writing code—it’s about how we, as developers, guide it.
Embrace AI to focus on what truly matters—innovative solutions and impactful applications.


Thank you
Keep exploring - the journey has just begun