



GitHub Copilot

Bootcathon 2024

Get an immersive learning experience on technology
and solve real energy business challenges

GitHub Copilot

Presented by
Kirati Srina (Bom), a web developer

“Unleash Your Coding Potential:
An Introduction to GitHub Copilot”



GitHub Copilot

What is GitHub Copilot?



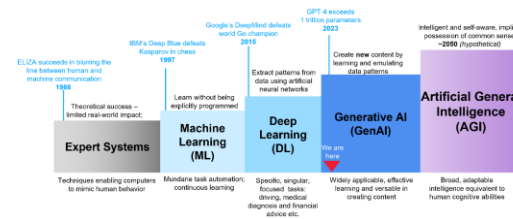
GitHub Copilot

- Help you write code faster
- Add-on feature of GitHub
- Integrate with IDEs e.g. VS Code, VS, JetBrains IDEs
- License cost by user per month



AI Coding Assistant

- General term
- An application of Generative AI technology
- Other tools: Codeium, Tabnine, AWS CodeWhisperer



Generative AI

- Capable of creating original content: text, image, codes
- Powered by Artificial Intelligence model aka. Large Language Model (LLM)



Microsoft Copilot

- AI Chatbot developed by Microsoft
- Summarizes documents, generates text and images
- Different product than GitHub Copilot

Cost

Copilot Individual

For individual developers, freelancers, students, and educators that want to code faster and happier.

\$ 10 USD

per month / \$100 USD per year

Start a free trial >

Free for verified students, teachers, and maintainers of popular open source projects.

What's included

Most popular

Copilot Business

For organizations ready to improve engineering velocity, code quality, and developer experience.

\$ 19 USD

per user / month

Buy now >

Contact sales >

What's included

New

Copilot Enterprise

For companies looking to customize GitHub Copilot to their organization and infuse AI across the developer workflow.

\$ 39 USD

per user / month

Contact sales >

What's included

<https://github.com/features/copilot/plans>

What Copilot Business Offer

Code Completion in your IDE

Autocomplete-style coding suggestions as you type in your IDE as “ghost text” where you can accept or ignore.

Chat in your IDE

Chat interface that lets you ask and receive answers to coding-related questions within supported IDEs.

Chat in the CLI

Chat-like interface in the terminal that allows you to ask questions about the command line.

This feature is currently disabled.

Chat in GitHub Mobile

Chat interface that lets you interact with GitHub Copilot within GitHub Mobile app.

This feature is currently disabled.

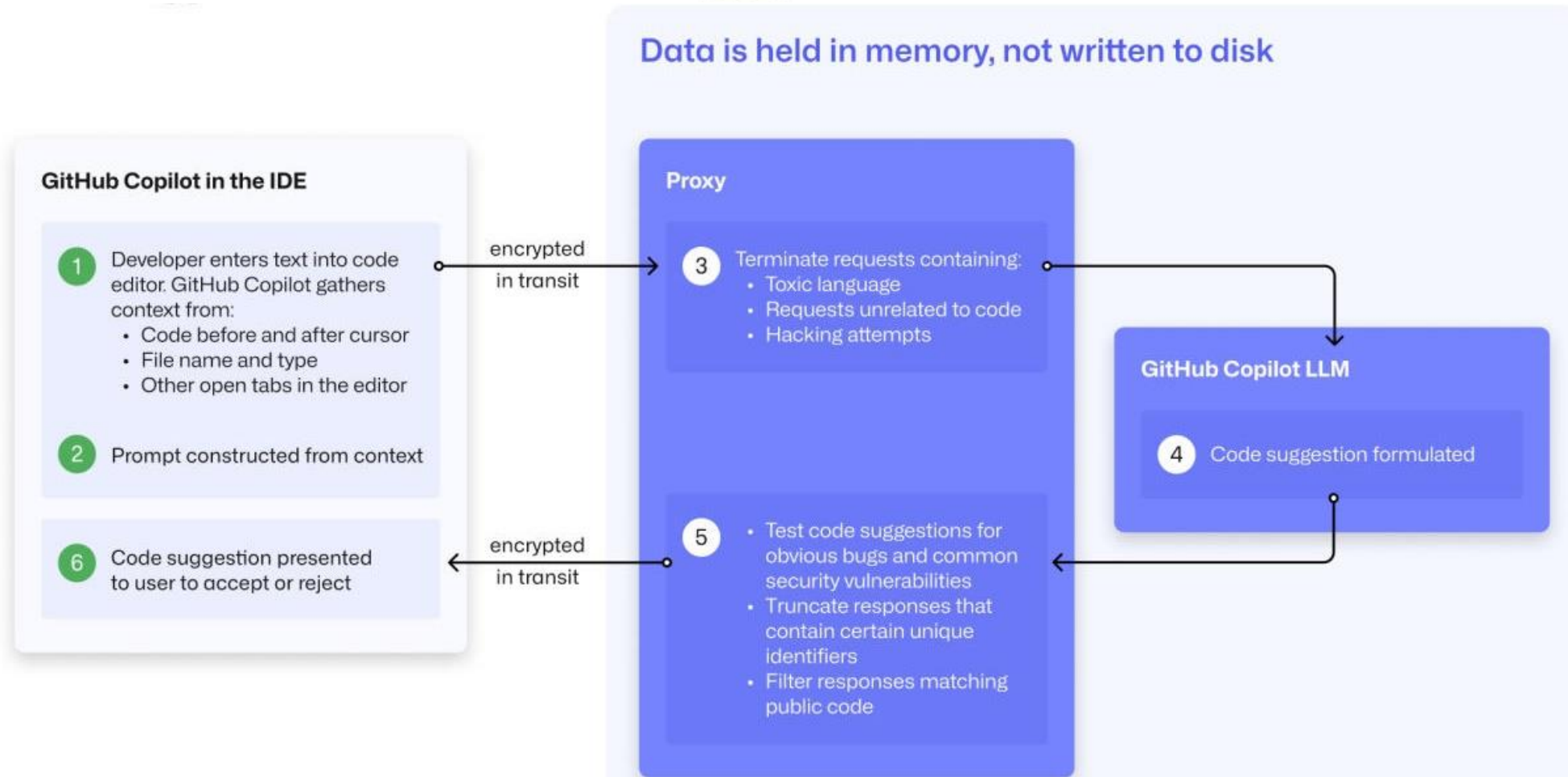
Content Exclusion

Enforce Copilot to ignore certain files or repositories.

Audit Logs

Review what actions have been taken by which users.

How GitHub Copilot Work



Security and Privacy



Security

- Copilot requires Internet.
- All data transfers are encrypted.
- AI-based vulnerability prevention
- Does not guarantee bug/vulnerability-free



Privacy

- Prompts and suggestions are not stored at rest.
- GitHub never use (our) data to train its model.
- GitHub collects and keeps User Engagement Data (UED) – reviewed by DPO and Law



IP and Open Source

- GitHub does not claim ownership.
- The models are trained on vast public data sets.
- Suggestions are not direct copy-paste.
- Suppress suggestions matching public code.

Benefits of Using GitHub Copilot

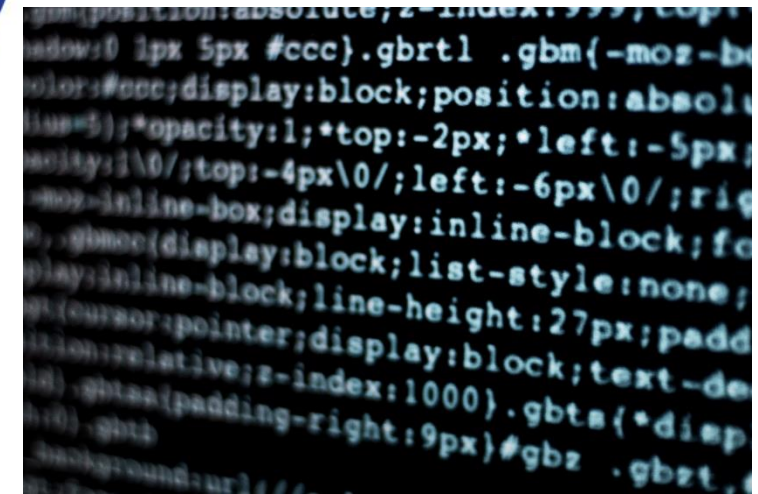
- **Enhanced Efficiency:** Accelerate coding by automating repetitive tasks and suggesting relevant completions.
- **Improved Code Quality:** Enforce best practices, maintain consistent style, and potentially reduce errors through suggestions.
- **Reduced Boilerplate:** Autogenerate common code patterns, freeing you to focus on core logic.
- **Exploration and Learning:** Discover new functions, libraries, and coding approaches through Copilot's suggestions.



How to use

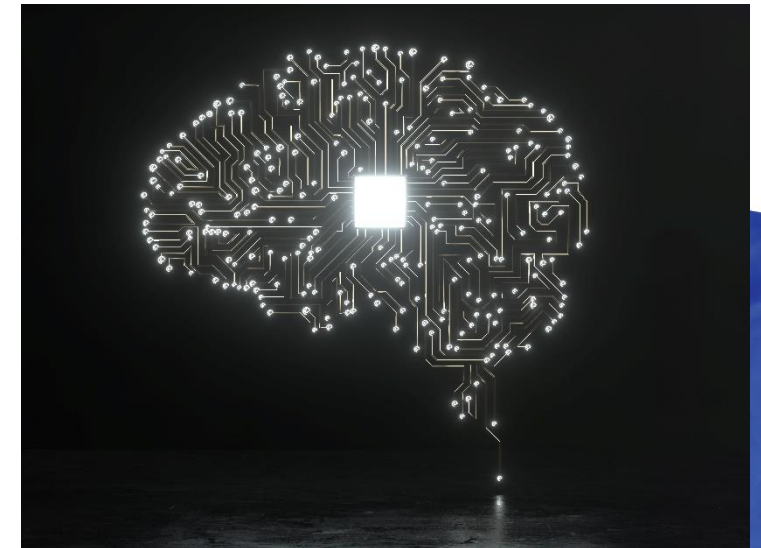
Code Completion Suggestions:

- Analyzes your code context (variables, function calls, surrounding code) to provide relevant suggestions in real-time.
- Suggestions appear as grayed-out text, allowing for easy acceptance (Tab key) or dismissal (Escape key).



Natural Language Code Generation:

- Understands natural language descriptions of desired functionality.
- Generates code snippets or functions based on your comments or prompts.



How to use

289



< 2/3 > Accept Tab Accept Word Ctrl + RightArrow ...

290

```
function onReset() {  
    document.getElementById('eventSearchInput').value = "";  
    tags = [true, true, true, true];  
    document.getElementById('tag-0').classList.add('em-c-tag--selected');  
    document.getElementById('tag-1').classList.add('em-c-tag--selected');  
    document.getElementById('tag-2').classList.add('em-c-tag--selected');  
    document.getElementById('tag-3').classList.add('em-c-tag--selected');  
    isShowEmptySlot = true;  
    document.getElementById('showEmptySlotCb').checked = true;  
    onSearch();  
}
```

How to use

```
289
290 function onReset() {
291     document.getElementById('eventSearchInput').value = "";
292     tags = [true, true, true, true];
293     document.getElementById('tag-0').classList.add('em-c-tag--selected');
294     document.getElementById('tag-1').classList.add('em-c-tag--selected');
295     document.getElementById('tag-2').classList.add('em-c-tag--selected');
296     document.getElementById('tag-3').classList.add('em-c-tag--selected');
297     isShowEmptySlot = true;
298     document.getElementById('showEmptySlotCb').checked = true;
299     onSearch();
300 }
```

How to use

```
290 // Convert timezone offset (minutes) to string, Example: -420 => '-0700'
```

```
291
```

```
290 ✨ Convert timezone offset (minutes) to string, Example: -420 => '-0700'
```

```
291 function getOffsetString(offset) {  
    const prefix = offset < 0 ? '-' : '+';  
    const hours = Math.floor(Math.abs(offset / 60)).toString().padStart(2, '0');  
    const minutes = Math.abs(offset % 60).toString().padStart(2, '0');  
    return `${prefix}${hours}${minutes}`;  
}
```

```
292
```

```
293
```

How to use

```
290 // Convert timezone offset (minutes) to string, Example: -420 => '-0700'
```

```
291
```

```
290 // Convert timezone offset (minutes) to string, Example: -420 => '-0700'
```

```
291 function getOffsetString(offset) {
```

```
292   const prefix = offset < 0 ? '-' : '+';
```

```
293   const hours = Math.floor(Math.abs(offset / 60)).toString().padStart(2, '0');
```

```
294   const minutes = Math.abs(offset % 60).toString().padStart(2, '0');
```

```
295   return `${prefix}${hours}${minutes}`;
```

```
296 }
```

```
297
```

```
298
```


How to use

```
290 Ask Copilot or type / for commands
291
292
```

- **Ctrl + I**

```
290
291
292 function convertBangkokToUtcTime(dateString, timeString) {
293     const dateTime = new Date(`${dateString} ${timeString} UTC`);
294     dateTime.setHours(dateTime.getHours() - 7);
295     return dateTime;
296 }
297
```

Convert Bangkok time to UTC time

GitHub Copilot

Made changes.

Accept Discard ↺ ↻

Accept or discard 1 change

How to use

290

291

292

293



Run: Generate using Copilot

```
29 @Component({
30   selectors: 'app-bulk',
31
32   Quick Fix
33   ✨ Fix using Copilot
34   ✨ Explain using Copilot
35   Move
36   🔑 Move to a new file
37   🔑 Move to file
38   SanitizePipe, HeadersPipe
39
40   1
```

You, yesterday • Uncommitted changes

plateOutlet,
Pipe,
BlockModule, StdToolbarModule,
dSortModule, StdPaginatorModule,
, StdSelectModule,
tdIconModule, StdTooltipModule,

Key Features

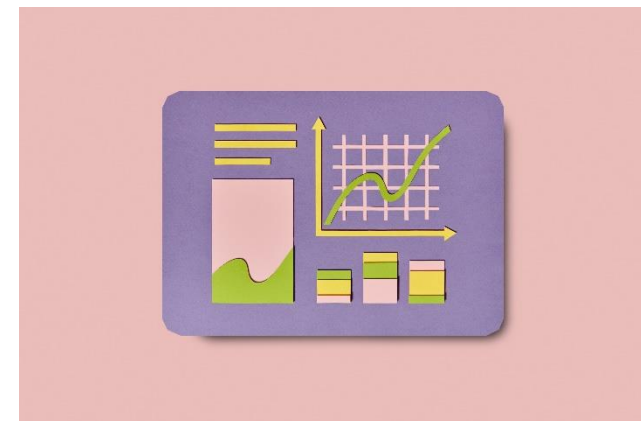
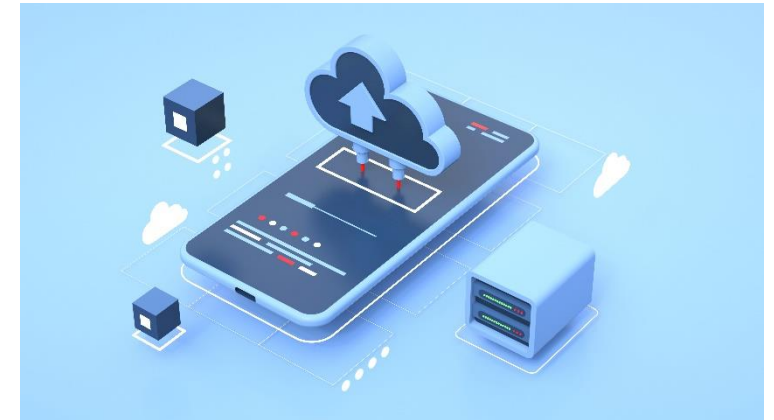
- **Increased Productivity:** Code completion, function suggestions, and automation of repetitive tasks significantly reduce coding time.
- **Improved Code Quality:** Enforces best practices, maintains consistent style, and can help identify potential errors through suggestions.
- **Reduced Boilerplate Code:** Autogenerates common code patterns (e.g., getters/setters, loop structures) for various functionalities.
- **Natural Language Processing:** Translates natural language descriptions into code, aiding developers of all experience levels.



Use Cases

Diverse Programming Scenarios

- **Web Development (JavaScript, HTML, CSS):** Autocomplete code for DOM manipulation, event handling, asynchronous operations, and more.
- **Data Science (Python, R):** Generate code for data analysis, visualization, machine learning tasks, and interacting with data science libraries.
- **Mobile Development (Java, Kotlin):** Simplify UI creation, data handling, asynchronous tasks, and working with mobile device features.



Limitations and Considerations

- **Potential for Errors or Bugs:** While Copilot suggests code, human review and testing are crucial to ensure correctness.
- **Overreliance Can Hinder Learning:** Don't blindly accept suggestions; strive to understand the underlying code and logic.
- **Subscription Costs (Consider Free Trial):** Explore the free trial to evaluate Copilot's value before committing to a paid subscription.



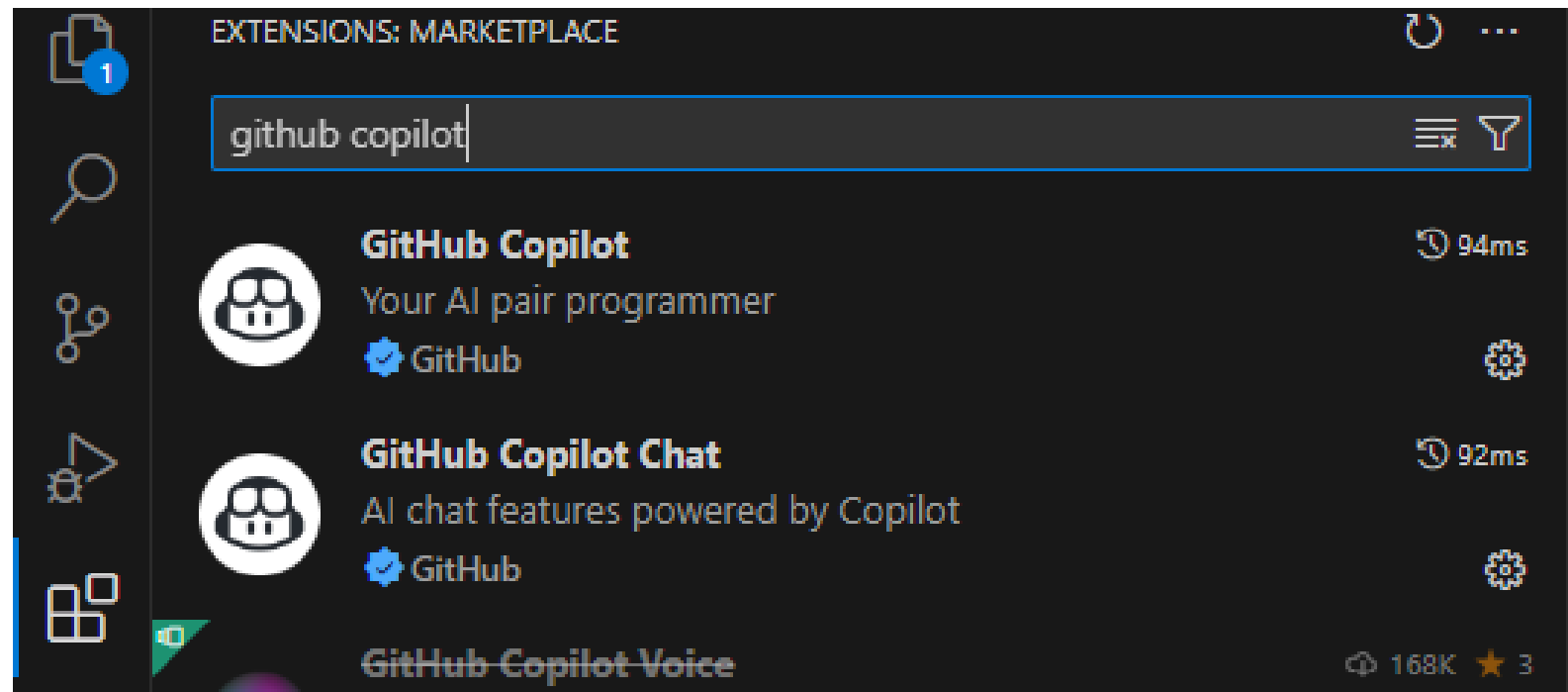
Getting Started

Installation Process

- Search for and install the "GitHub Copilot" extension in your compatible editor/IDE (VS Code, JetBrains IDEs, etc.).
- Sign in to your GitHub account with an active Copilot subscription (free trial or paid plan).

Compatibility

- GitHub Copilot works with a wide range of programming languages, including JavaScript, Python, Java, C++, and many more.

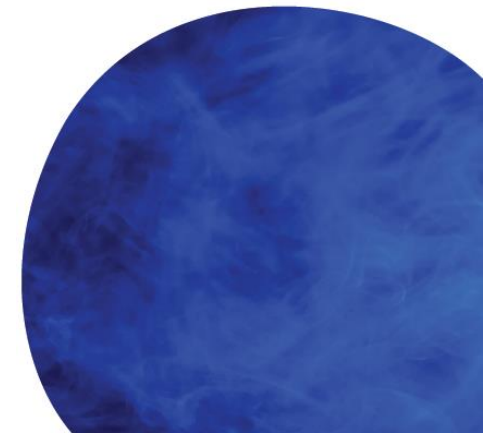


Get free trial of GitHub Copilot

- 1. Visit the GitHub Copilot Page:** Go to the official GitHub Copilot page (<https://copilot.github.com/>).
- 2. Sign In:** You need to sign in with your GitHub account. If you don't have one, you'll need to create a GitHub account first.
- 3. Start Trial:** Look for a button or link that says "Start your free trial" or similar. The availability of a free trial and its duration may vary based on current promotions or offers by GitHub.
- 4. Install GitHub Copilot:** After starting your trial, follow the instructions to install GitHub Copilot in your IDE. GitHub Copilot is available for Visual Studio Code, Neovim, and JetBrains IDEs.
- 5. Activation:** Once installed, GitHub Copilot should automatically activate, and you can start using it right away.

How to write efficiency prompt (guidelines)

- **Be Specific:** Clearly state what you need. Include details like programming languages, frameworks, and any specific requirements.
- **Include Context:** If your request is related to existing code, provide relevant context or code snippets.
- **State the Goal:** Explain what you aim to achieve with the request.
- **Mention Constraints:** If there are any constraints (e.g., performance, memory usage), mention them.
- **Ask One Thing at a Time:** Focus your prompt on a single request to get a more accurate response.



How to write efficiency prompt (continue)

Pseudocode:

- Start with a brief description of the task.
- Specify the programming language or technology.
- Provide any relevant code snippet or context.
- Clearly state the expected outcome.
- Mention any specific constraints or requirements.

How to write efficiency prompt (continue)

Example prompt:


I'm working on a Python project using Flask.

I need to write an endpoint that accepts JSON data with a user's name and age, then returns a message saying if the user is an adult or not.


The endpoint should respond with a 400 status code if the required fields are missing.

Can you help me scaffold this?

How to write efficiency prompt (continue)

 kirati-exxonmobil

"I'm working on a Python project using Flask. I need to write an endpoint that accepts JSON data with a user's name and age, then returns a message saying if the user is an adult or not. The endpoint should respond with a 400 status code if the required fields are missing. Can you help me scaffold this?"

 GitHub Copilot

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/check_age', methods=['POST'])
def check_age():
    data = request.get_json()
    # Check if both name and age are provided in the JSON data
    if not data or 'name' not in data or 'age' not in data:
        return jsonify({'error': 'Missing name or age'}), 400

    name = data['name']
    age = data['age']

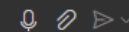
    # Check if the user is an adult
    if age >= 18:
        return jsonify({'message': f'{name} is an adult.'})
    else:
        return jsonify({'message': f'{name} is not an adult.'})

if __name__ == '__main__':
    app.run(debug=True)
```

This Flask application defines an endpoint `/check_age` that accepts POST requests. It expects JSON data containing a user's `name` and `age`. The endpoint checks if the user is an adult (18 years or older) and returns an appropriate message. If the required fields are missing, it responds with a 400 status code and an error message.

How can I handle errors and exceptions in Flask?

Ask Copilot or type / for commands



How to write efficiency prompt (continue)

Example prompt:

Task: Create a REST API endpoint for registering users

Language: Python, Flask

Context: This endpoint should be part of an existing Flask application for managing user data.

Requirements: Accepts a username and password in JSON format. Validate that both fields are present. Respond with a 200 status code and a welcome message if successful, or a 400 status code with an error message if any field is missing.

Example: Expected JSON response on success: {"message": "Welcome, [username]!"}

How to write efficiency prompt (continue)



kirati-exxonmobil

Task: Create a REST API endpoint for registering users Language: Python, Flask Context: This endpoint should be part of an existing Flask application for managing user data. Requirements: Accepts a username and password in JSON format. Validate that both fields are present. Respond with a 200 status code and a welcome message if successful, or a 400 status code with an error message if any field is missing. Example: Expected JSON response on success: {"message": "Welcome, [username]!"}



GitHub Copilot

> Used 1 reference

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/register', methods=['POST'])
def register_user():
    data = request.get_json()
    username = data.get('username')
    password = data.get('password')

    if not username or not password:
        return jsonify({'error': 'Username and password are required'}), 400

    return jsonify({'message': f'Welcome, {username}!'}), 200

if __name__ == '__main__':
    app.run(debug=True)
```



Q&A