Prompt Engineering Notes for Coders and Hackers

1. Ask the Model to Adopt a Role or Define a Persona

title

When you assign a specific role to the model (e.g., "Be an ethical hacker" or "Act as a Python tutor"), its responses become more targeted and relevant.

How: Start the prompt with `Act as` or `You are a [role/persona]`. **Example:**

```
You are a cybersecurity expert. Explain how to ethically test the security of a
```

title

The model provides ethical and professional advice specific to cybersecurity.

2. Include Details in Your Query to Get More Relevant Answers

title

The more context you provide, the more accurate and tailored the response.

How: Clearly state the language, framework, tools, or version you are working with. **Example:**

```
I am using Python 3.10 with Flask. \\
\vspace{0.5cm} % Adjust space as needed
How can I create an API endpoint to accept JSON data and store it in an SQLite d
```

title

A detailed response with Python 3.10 and Flask-specific code.

3. Specify the Steps Required to Complete a Task

title

Breaking down tasks into steps helps the model structure the response in a way that is easy to follow and implement.

How: Use phrases like `Step-by-step`, `List the steps`, or `Explain the process`. **Example:**

```
Provide step-by-step instructions to set up a Node.js server using Express and M
```

title

A step-by-step guide to set up the server, install dependencies, and connect to MongoDB.

4. Provide Examples to Enhance Understanding

title

Examples give the model context for the output you expect and improve response quality.

How: Include your example and specify the desired format. **Example:**

```
Create a Python function to hash a password using bcrypt. Example:
Input: "my_password"
Output: A hashed password
```

title

A Python function using bcrypt to hash the given password.

5. Request Explanations or Justifications

title

Understanding the reasoning helps validate or refine the solution.

How: Use phrases like `Explain why` or `Provide reasoning for`. **Example:**

```
Write a secure SQL query to prevent injection attacks and explain why it is secu
```

title

Code using parameterized queries and an explanation of how it prevents SQL injection.

6. Ask for Optimization or Alternatives

title

Sometimes you need better performance, simpler code, or different approaches.

How: Use phrases like `Optimize this code` or `Provide alternative methods`. **Example:**

```
Optimize the following Python code for readability and performance:
```

Followed by your code.

7. Define Specific Constraints or Rules

title

Constraints guide the model to provide solutions that meet your exact requirements.

How: Mention constraints like `Use no external libraries` or `Must be under 10 lines`. **Example:**

```
Create a Python function to sort a list without using the built-in ‘sort()‘ func
```

title

A sorting function that does not rely on `sort()`.

8. Combine Multiple Tasks in One Prompt

title

Saves time and gets a comprehensive answer.

How: Use conjunctions like `and` or `while` to connect tasks. **Example:**

```
Write a Python script to scrape data from a website and save it to a CSV file. A
```

title

A complete script that includes web scraping, data saving, and error handling.

9. Specify Output Format

title

Ensures the response matches your expected structure or style.

How: Use phrases like `Output as` or `Provide in [format]`. **Example:**

```
Generate a JSON schema for a to-do app with fields: id, title, description, and
```

title

A JSON schema in the desired format.

10. Iterate and Refine Prompts

title

If the initial response is not accurate, refine the prompt to include additional context or adjust instructions.

How: Use phrases like `Refine this` or `Based on this code, do the following`. **Example:**

```
Based on the Node.js server code you provided, add a route for deleting a user b
```

title

An updated code snippet with the delete route.

11. Avoid Ambiguity

title

Ambiguity in the prompt can lead to irrelevant or generic responses.

How: Be as specific as possible about your goals. **Example:**

```
Bad: Write a function in JavaScript.
Good: Write a JavaScript function to reverse a string without using built-in met
```

title

A targeted solution for reversing a string.

12. Ask for Debugging or Explanation of Errors

title

Useful for finding bugs and understanding issues in your code.

How: Share the problematic code and describe the issue. **Example:**

```
Here's my Python code:
```

Followed by the code and the error message.

```
Help me fix this and explain why the error occurs.
```

---

13. Experiment with Context Length

title

Adding background details or omitting irrelevant context affects the quality of the response.

How: Provide necessary details but avoid overloading the prompt. **Example:**

```
I am building a REST API in Django. How do I implement token-based authenticatio
```

title

Focused guidance on JWT implementation in Django.

---

14. Test for Edge Cases

title

Verifies that the solution works in all scenarios.

How: Ask the model to include edge case handling in its response. **Example:**

```
Write a Python function to check if a string is a palindrome. Include edge cases
```

---

title

By mastering these techniques, you can maximize the value you derive from AI tools.

==========Done=================

udemy Ai for Devleopers with Github copilot, Cursor Ai chatGpt course...

Your Name

April 26, 2025

1 Introduction

This document provides a detailed guide on installing and setting up **GitHub Copilot**, a powerful AI-powered code completion tool that enhances your coding experience across various editors and IDEs. **Important steps** for installation and setup are highlighted in red, and **key concepts** are emphasized in green for clarity.

2 Extension Tab se Install Karna

This section explains how to install GitHub Copilot in Visual Studio Code (VS Code).

2.1 Step-by-Step Process (Visual Studio Code ke example ke saath)

- Open karo apna **VS Code**.
- Left sidebar mein jao **Extensions Tab** pe (ya shortcut `Ctrl+Shift+X` dabao).
- Search karo **"GitHub Copilot"**.
- Install karo **"GitHub Copilot"** extension.
- Phir search karo **"GitHub Copilot Chat"** aur isko bhi install karo.

Check: **Ab aapka Copilot aur Copilot Chat ready hai use karne ke liye.**

3 Copilot Availability in Different IDEs/Editors

GitHub Copilot is available for **alag-alag editors aur platforms**. Below is a table summarizing the supported platforms:

Platform	Support	Setup Steps
		Visual Studio Code
Supported	Install extensions directly from marketplace.	Supported
	Visual Studio (2022 aur aage ke versions)	
Visual Studio Extension Manager se install karo.	Supported	JetBrains Marketplace se ”GitHub Copilot” plugin install karo.
JetBrains IDEs (Jaise: IntelliJ IDEA, PyCharm, WebStorm)		Neovim
Supported	Plugin manager ke through GitHub Copilot ka plugin install karo.	Supported
	Web Browser (CodeSpaces, GitHub.dev)	
Browser ke andar hi extensions activate hote hain.	**Abhi Direct Official Support nahi hai**	JetBrains Plugin work kar sakta hai kuch workaround ke saath, but officially Android Studio ke liye Copilot fully supported nahi hai.
Android Studio		

Table 1: Copilot Availability Across Platforms

4 Detailed Installation Steps for Different Editors

This section provides detailed steps for installing GitHub Copilot in various editors.

4.1 Visual Studio Code

- Extension Marketplace khol ke install karo: **"GitHub Copilot"** aur **"GitHub Copilot Chat"**.
- GitHub Account se login karo jab prompt aaye.
- Setting mein jaake customize kar sakte ho (jaise: suggestions ka frequency, chat shortcuts, etc.).

4.2 Visual Studio (Windows ke liye)

- Visual Studio ko open karo.
- Extension Manager (`Extensions ¿ Manage Extensions`) mein jao.
- Search karo **"GitHub Copilot"**.
- Install karo aur Visual Studio ko restart karo.
- GitHub Account se login karo.

4.3 JetBrains IDEs (IntelliJ, PyCharm, etc.)

- IDE open karo.
- `Settings/Preferences ¿ Plugins ¿ Marketplace` mein jao.
- Search karo **"GitHub Copilot"**.
- Install karo aur IDE restart karo.
- GitHub login process complete karo.

4.4 Web Browser (CodeSpaces, github.dev)

- Browser mein jab aap **CodeSpaces** ya **github.dev** open karte ho,
- Agar Copilot enabled hai, to directly extensions ka support milta hai.
- **Manual installation ki zarurat nahi hoti.**

5 Tips for Best Usage

To maximize your experience with GitHub Copilot:

- Hammesha apne extensions ko **updated** rakho.
- Correct GitHub account (jispe Copilot subscription active hai) se login karo.
- Settings mein apne hisaab se Copilot behavior (jaise suggestions ki frequency) tweak kar sakte ho.
- Copilot Chat ka use **complex queries aur code explanation** ke liye karo.

6 Final Summary

This document summarized the key steps for setting up GitHub Copilot:

- **Install karo GitHub Copilot aur Copilot Chat** apne editor ke hisaab se.
- **Har editor ke liye alag setup steps hain**, wo dhyan se follow karo.
- **Galat setup se productivity affect hoti hai**, isliye sahi installation important hai.
- **Copilot aur Chat dono milke coding experience next-level bana dete hain!**

================================ ====================================

Mastering Prompts using Comments with GitHub Copilot's AI

7 Introduction

Prompt likhne ka matlab sirf input dena nahi hai — **agar aap sahi comment likho**, toh GitHub Copilot ussi ke hisaab se **accurate code generate karta hai**. This document explains how to use comments as prompts and leverage Copilot's inline chat feature.

8 Kaise kaam karta hai?

Copilot aapke likhe gaye **comments ko prompt** ki tarah treat karta hai, aur uske base pe **smart code suggestions** deta hai.

9 Step-by-Step: Comment-based Prompting

9.1 Ek meaningful comment likho

Listing 1: Example Python Comment

```python
# function that takes two parameters a and b, adds them and returns the
    result
```

9.2 Cursor ko next line pe rakho

Copilot automatically suggestion dega:

Listing 2: Generated Python Code

```python
def add(a, b):
    return a + b
```

Check: **Ye comment hi prompt ban gaya AI ke liye.**

10 Example Prompts (Different Languages)

10.1 Python

Listing 3: Python Prompt

```python
# generate a fibonacci series of n numbers
```

10.2 JavaScript

Listing 4: JavaScript Prompt

```javascript
// create a function to check if a number is prime
```

10.3 Kotlin (for Android)

Listing 5: Kotlin Prompt

```
// function to validate email format using regex
```

Check: **Jitna zyada clear and specific comment likhoge, utni zyada useful suggestion milegi.**

11 Agar vague comment doge

Listing 6: Vague Comment

```
# do something
```

Copilot confused hoga aur **random ya generic code** dega.

12 Best Practices

- Comment mein **task clearly explain karo**.
- Agar possible ho to **inputs aur output** define karo.
- Use karo "how", "what", ya "return" jaise words.

13 Using the Inline Chat Feature (Copilot Chat inside Editor)

GitHub Copilot ke paas ek aur powerful tool hai: **Editor Inline Chat**. Iska use aap code ko contextually modify karne ke liye kar sakte ho.

14 Step-by-Step: Inline Chat ka Use

14.1 Function ya code block select karo

(Jis part mein change chahiye)

14.2 Right-click karo

Choose: **GitHub Copilot - Inline Chat**

14.3 Prompt likho

Listing 7: Inline Chat Prompt

```
Change this function to also return the difference between a and b.
```

Copilot will:

- Understand selected code
- Suggest changes **only to that block**

15 Why Select Code First?

Agar aap pura code select nahi karte, toh Copilot pura file ya nearby context mein change suggest karega. **Isliye select karna zaroori hai** taaki sirf wahi block modify ho jaye.

16 Special Trick: Use #filename in Prompt

Agar aap inline chat mein ye likhte ho:

Listing 8: Prompt with Filename

```
#main.py     Convert this function to async
```

Copilot samjhega ki aap **main.py ke context** mein kaam kar rahe ho, aur accordingly suggestions dega.

17 Tips for Effective Inline Chat Prompts

18 Agar Inline Chat ya Comments use nahi karte

- **Manual changes karne padenge**
- Copilot ki full power use nahi ho paayegi
- Repetitive code editing mein time waste hoga

Weak Prompt	**Better Prompt**
optimize this function for large inputs	make this better
change it	convert this function to use async/await
fix the off-by-one error in this loop	fix it

Table 2: Weak vs. Better Inline Chat Prompts

19 Summary

This document summarized the key points for mastering GitHub Copilot:

- **Comments = Prompts** — Clear likho, relevant likho.
- Use **Inline Chat** jab selected code block ko change karna ho.
- Prompt mein **#filename** likhne se Copilot context aur better samajhta hai.
- Ye tools **productivity boost** karte hain aur debugging/editing easy bana dete hain.

Using Inline Chat Feature in GitHub Copilot

20 Kaise use kare Inline Chat

- Jab aapko **sirf ek particular function** me code changes chahiye, to:
  - **Poora function select karo** (na ki sirf ek line), taki changes sirf usi function me ho.
  - **Right-click** karo aur **"Copilot: Edit with Inline Chat"** option choose karo.
- Ab ek **Inline prompt** open hoga.
  - Yahan pe **apna prompt likho** — jo changes chahiye woh batao.
  - Copilot sirf **selected code ke andar** hi changes karega.

21 Note

Agar aap **Inline Chat** ke prompt me `# filename` mention karte ho,
to Copilot **ussi file ke context** ke hisaab se code generate karega.
Example: Agar aap `# utils.js` likhte ho, to Copilot sochta hai ki yeh code `utils.js` file ke according hona chahiye.

22 Kab GitHub Copilot ka "Chat" aur "Inline Chat" use karna chahiye?

Situation	**Use Chat Feature**	**Use Inline Chat Feature**
		Jab aapko **general coding help**, **concept explain** karwana ho (jaise "what is debounce function?")
	Jab aapko **poori file me improvement** ya **naya code generate** karwana ho	
Jab aapko **sirf ek specific chhoti jagah** (jaise ek function ya ek block of code) me **change** karwana ho		Jab aapko **bigger refactoring** ideas chahiye (file ka structure change karna, best practices ke hisaab se)
	Jab aapko ek chhoti **specific cheez** fix ya modify karni ho **within selected code**	

Table 3: Chat vs. Inline Chat Usage

23 Example

23.1 Chat Feature ka Example

Mujhe React me lazy loading ke concept samjhao aur ek example code do.

23.2 Inline Chat Feature ka Example

Ek function select karke prompt dena:

Listing 9: Inline Chat Prompt

```
Convert this function to use async/await instead of .then() chaining.
```

24 Quick Tip

- **Chat** zyada useful hota hai jab aapko **idea generation** ya **general help** chahiye.
- **Inline Chat** best hota hai jab aapko **precision targeting** chahiye — sirf selected code me changes karne ke liye.

================================= ==============================

Configuring GitHub Copilot – Tips for Efficient Use (VS Code)

25 Introduction

Agar aap chahte ho ki GitHub Copilot **aapke coding style ko samjhe aur usi ke according suggestions de**, toh kuch **important settings** ko configure karna zaroori hai.

26 Step-by-Step: Open Copilot Settings in VS Code

1. Open karo **VS Code**.
2. Go to: `File > Preferences > Settings`
   Ya shortcut: `Ctrl + ,`
3. Search karo: **"Copilot"**
   (Aapko GitHub Copilot se related saari settings dikhegi)

27 Enable "Use Instruction File" Option

27.1 Setting: Use instruction file

Check: Ye setting GitHub Copilot ko **aapki likhi hui instructions follow karne ke liye bolti hai**, jisme aap define kar sakte ho:

- Naming convention
- Function structure
- Preferred patterns

27.2 File: .github/copilot-instructions.md

Check: Ye file aapke project root me hoti hai, agar nahi hai to manually bana sakte ho.

27.3 Isme likho instructions like

Listing 10: Example copilot-instructions.md

```
Prefer snake_case variable names.
Split long functions into smaller ones.
Avoid using nested ternary operators.
Use comments before complex logic.
```

Ab Copilot in instructions ko **base banake suggestions dega**.

28 Example

28.1 Instruction diya

Listing 11: Instruction Example

```
Use snake_case for variables.
```

28.2 Copilot Suggestion

Listing 12: Generated Python Code

```
def get_user_data(user_id):
```

Check: **Aapki style ka automatically dhyan rakha jaata hai!**

29 Enable "Temporal Context" Option

29.1 Setting: Enable temporal context

Check: Is setting ka kaam hai:
Copilot ko allow karna ki wo aapke last edited files ka context use kare
aur uske base pe naye file mein suggestion de.

30 What exactly it does?

- Recent file (jo aapne edit kiya hai) ka code context mein liya jaata hai.
- Us file ke functions ko **suggested file mein import karke** use karta hai.
- Helps in **cross-file suggestions** aur **project-level awareness**.

31 Ye Feature Sirf Inline Chat ke liye hota hai kya?

Nahi!
'Temporal Context' feature sirf **inline chat tak limited nahi hai.**
Check: Ye feature:

- **Inline Suggestions** (jo normal typing ke dauraan milti hain)
- **Inline Chat** (selected block par prompt dete ho)

dono mein kaam karta hai — especially jab aap ek bade project pe ho jahan kai files **interlinked** hoti hain.

32 Quick Summary Table

33 Agar ye settings enable nahi ki

- Copilot aapke **style** ko samjhe bina generic suggestion dega.
- Recent file context ka use nahi hoga → suggestions **incomplete ya irrelevant** ho sakte hain.
- AI personalization ka faida nahi milega.

Setting	Purpose	File Used	Where it works
Suggest code based on your coding style	.github/copilot-instructions.md	Inline Suggestions & Chat	Check: Use instruction file
No file, automatic	Inline Suggestions & Chat	Check: Temporal context	Use recent file edits to influence code suggestions

Table 4: Summary of Copilot Settings

34 Best Practices

- Har project ke liye ek **.github/copilot-instructions.md** file zarur banao.
- Har new project start karte hi ye 2 settings enable karo:
  - **Use Instruction File**
  - **Temporal Context**
- Ye settings aapko **project-level smartness** aur **consistent coding pattern** denge.

35 Summary

This document summarized key tips for configuring GitHub Copilot in VS Code:

- **Use instruction file** to enforce coding style.
- **Temporal context** for cross-file suggestions.
- Create **.github/copilot-instructions.md** for every project.
- Enable both settings for **smart and consistent** coding experience.

================================ ================================

Taking Advantage of Code Actions (GitHub Copilot)

36 What are Code Actions?

Jab aap **code ke kuch lines select karte ho** VS Code mein, toh ek **small star icon** dikhta hai — ye **Code Action** ko trigger karne ka symbol hota hai.

37 Step-by-Step: How to Use Code Actions

1. Code ke kuch lines **select karo**.
2. Dekho **ek star type ka icon** left side margin mein ya line ke end pe aayega.
3. Click karo star icon pe.

Aapko **2 important options** milte hain:

Option	**What it Does**
	Modify using Copilot
Aapke selected code ko rewrite karta hai ya improve karta hai based on AI understanding.	Code ko analyze karta hai aur suggestions ya feedback deta hai (jaise performance improve karne ke liye tips).
Review using Copilot	

Table 5: Code Action Options

38 Modify using Copilot

38.1 Kya karta hai?

- Aapke existing code ko **modify ya refactor** karta hai.
- Better readability, error fixing, ya optimization suggest karta hai.

38.2 Kab Use Karein?

Jab aap chahte ho ki aapka code **aur efficient, clean ya modern** ban jaye.

38.3 Example

Suppose aapne ye likha hai:

Listing 13: Original Python Code

```
def add(a, b): return a+b
```

Select karo, Modify using Copilot choose karo → Suggestion aa sakta hai:

Listing 14: Modified Python Code

```
def add(a, b):
    """Returns the sum of two numbers."""
    return a + b
```

Check: **More readable and documented**

39 Review using Copilot

39.1 Kya karta hai?

- Aapke code ko **audit karta hai**.
- Performance, best practices, ya hidden bugs ka feedback deta hai.

39.2 Kab Use Karein?

- Jab aapko apne likhe code ka **AI based review** chahiye.
- Especially jab code complex ho ya production ready karna ho.

39.3 Example

Suppose aapne likha:

Listing 15: Original Python Code

```
def getData():
  pass
```

Select karo, Review using Copilot → Suggest karega:

- Better naming: `get_data()`
- Add docstring
- Raise `NotImplementedError` inside empty functions

40 Sidebar Chat vs Inline Chat — Kab use karein?

Feature	**Inline Chat**	**Sidebar Chat**
Sirf selected code block pe focus karta hai	Pure file ya project-level broader context pe kaam karta hai	Context Quick fixes ya small modifications ke liye
Detailed discussions ya large feature additions ke liye	Speed Ek function ko async mein convert karna	Pura app ke structure pe suggestion lena (like: How to implement auth system?)
Example Chhoti chhoti code improvements ke liye	Jab aapko detailed, research-level ya big feature guidance chahiye	When to Use

Table 6: Inline Chat vs. Sidebar Chat

41 Quick Examples

41.1 Inline Chat Example

Listing 16: Inline Chat Prompt

```
Select a function    Inline Chat    Prompt: "Convert this to async."
```

Check: **Sirf wahi function async mein convert hoga.**

41.2 Sidebar Chat Example

Listing 17: Sidebar Chat Prompt

```
Prompt: "How to implement Google Login in React Native app?"
```

Check: **Ye aapko pura structure aur libraries batayega.**

42 How to Open Inline Chat in VS Code Terminal?

Normally Inline Chat sirf Editor pe hota hai.
Direct Terminal mein Inline Chat open nahi hota.
Check: Use: `Ctrl + I` (or right-click in editor ¿ GitHub Copilot - Inline Chat)

Terminal se directly nahi, lekin **keyboard shortcut** ya editor ke shortcut se aap inline chat instantly trigger kar sakte ho.

43 Agar Inline Chat aur Sidebar Chat ka galat use kiya

- Inline Chat mein bade complex features banwane ki koshish karoge toh **context lost hoga**.
- Sidebar Chat mein chhoti chhoti cheeze poochoge toh **time waste hoga**.

44 Best Practices

- **Chhote code improvements ke liye** → Inline Chat use karo.
- **Large feature development ke liye** → Sidebar Chat use karo.
- Jab kuch specific lines ko refactor/modify karwana ho → **Code Actions** use karo.

45 Final Summary

This document summarized key points for using Code Actions with GitHub Copilot:

- Check: **Code Action = Quick AI help on selected code**
- Check: **Modify using Copilot = Code ko improve/refactor karo**
- Check: **Review using Copilot = Code ka AI based review aur feedback lo**
- Check: **Inline Chat = Focused, small edits**
- Check: **Sidebar Chat = Big, full-featured help**
- Check: **Terminal se direct Inline Chat nahi, editor shortcut se karo.**

==

Multiple Edits with Copilot Editors

46 Kya hai Copilot Edits Mode?

- **Copilot Edit Mode** ek feature hai jo **GitHub Copilot Chat** me aata hai.
- Jab aap chat kholte ho, upar **+ button ke paas** ek option milta hai: **"Copilot: Edit"**.
- Is mode ka use hota hai jab aap **apne existing codebase me multiple jagah edits karwana chahte ho**, ek single prompt ke through.

Samajhne ke liye: Jaise ek command doge aur Copilot poore project me ya specific file me changes suggest karega.

47 Copilot Edits Mode Kya Karta Hai?

- Aap ek **natural language prompt** dete ho (jaise normal English me).
- Copilot **poori file**, **selected files**, ya **poore project (workspace)** ke context me dekhta hai.
- Fir **code updates, improvements, ya naya code add** karta hai jaha zarurat ho.
- Ye **inline suggestions** deta hai, jisko aap **accept ya reject** kar sakte ho.

48 Kab Use Karna Chahiye Copilot Edits Mode?

49 Important Concepts: `workspace`, `#filename`, and Common Decorators

49.1 1. `workspace`

- **Meaning:**
  Iska matlab hota hai **poore project (workspace)** ke context me kaam karna.
- **Use kab kare:**
  Jab aapko **project ke kai files** me changes chahiye based on your prompt.

Situation	**Use Copilot Edits Mode**
	Jab aapko **ek type ka change multiple files** me karwana hai
Jab aapko **bohot bade codebase me ek improvement** karni hai (e.g., har function ko async banana)	Jab aapko **small refactoring** chahiye across project (e.g., add error handling)
Jab aapko **general edits ek hi file me** karne ho	Jab aapko **new feature add karna ho specific file me**

Table 7: When to Use Copilot Edits Mode

- **Example:**
  `workspace Create a login form inside #login.html`
  Yani, Copilot pura workspace dekhega aur `login.html` file ke andar login form banayega.

49.2 2. `#filename`

- **Meaning:**
  Isse aap Copilot ko **specifically ek file** ka context dete ho.
- **Use kab kare:**
  Jab aapko sirf ek file me changes chahiye.
- **Example:**
  `#dashboard.js Add a new function to fetch user analytics`
  Yani, Copilot sirf `dashboard.js` file ke liye kaam karega.

49.3 3. Commonly Used Decorators in Copilot Edit Prompts

50 Kaise Likhe Achha Prompt for Copilot Edits?

1. **Clear and short likho:**
   `workspace Change all var to let and const`
2. **Specify file agar specific ho:**
   `#profile.html Add user avatar upload field`
3. **Focus on one task at a time:**
   `openEditors Add console error logging wherever a fetch fails`

Decorator	Meaning	Example
Project ke sare files ke context me kaam karega	`workspace` `Add input validation to all login forms`	`workspace` Specific file me changes karega
`#userModel.js` `Add email verification method`	`#filename` Jo files currently open hai editor me, unke liye kaam karega	`openEditors` `Refactor all functions to arrow functions`
`openEditors` Sirf currently active file me changes karega	`activeFile` `Improve error handling in API calls`	`activeFile`

Table 8: Commonly Used Decorators

4. **Use decorators properly:**
   Jaise project wide ho to `workspace`,
   sirf ek file ke liye ho to `#filename`.

51 Quick Tips

- Agar bada project hai to **workspace** powerful hota hai, but dhyan rahe bohot saare unwanted changes ho sakte hai, to carefully review karo.
- Agar sirf ek file ya open file me chhote changes chahiye to **activeFile** ya **#filename** perfect hai.
- Always **read and review suggestions** before applying edits!

52 Example Real-World Use Case

52.1 Case 1: Add Comment Headers to All API Files

Listing 18: Workspace Prompt

```
@workspace Add function header comments to all API related files.
```

52.2 Case 2: Modify Specific File Only

Listing 19: File-Specific Prompt

```
#authService.js Change password hashing algorithm to bcrypt.
```

52.3 Case 3: Refactor All Open Files

Listing 20: Open Editors Prompt

```
@openEditors Replace all ‘var‘ declarations with ‘let‘ or ‘const‘.
```

===

GitHub Copilot Extensions – Boost Your Copilot’s IQ!

53 What are Copilot Extensions?

GitHub Copilot Extensions ka matlab hai:
Aise **official add-ons** jo GitHub Copilot ko **naye domains** mein **smart banate hain**, jaise Docker, Kubernetes, Python Testing, etc.

54 Why Use Copilot Extensions?

Normal Copilot sirf general coding suggestions deta hai.
Lekin Extensions se aap usko **domain-specific knowledge** de sakte ho, jisse wo:

- Domain-specific files (like Dockerfile) generate kare
- `@` syntax ke through direct help de
- Better and accurate responses de in specific tech areas

55 Example: Docker for GitHub Copilot

Ye extension Copilot ko **Docker aur Docker Compose** ke baare mein sikhata hai.

55.1 Aap kya kar sakte ho isse?

- Generate `Dockerfile`, `docker-compose.yml`
- Docker container setup

- Multi-stage builds
- Docker commands

55.2 Copilot Chat mein likho

Listing 21: Docker Extension Prompt

```
@docker create a Dockerfile for Node.js app with port 3000 exposed
```

Check: **Copilot smartly Dockerfile bana dega!**

56 How to Add Copilot Extensions (These are NOT Regular VS Code Extensions)

56.1 Step-by-Step Guide

1. Open GitHub Copilot Chat panel in VS Code
   (Shortcut: `Ctrl+I` or sidebar icon)
2. Top-right corner pe dekho: Settings icon → Click karo
3. Waha milega: **Extensions tab**
4. Aapko list milegi available Copilot extensions ki, jaise:
   Docker, Kubernetes, Python Test Generator, YAML, Terraform, etc.
5. Click on **Install** button next to the extension you want.
6. Bas ho gaya! Copilot ab uss domain mein **aur intelligent** ho gaya.

57 What Happens After Installation?

- Copilot aapke installed extensions ke hisaab se suggestions dena start karega.
- Chat mein `@extensionname` ka use karke direct prompt kar sakte ho.

58 Example Prompts Using Installed Extensions

Extension	**Prompt**
`@docker generate a multi-stage Dockerfile for a Python Flask app`	`@docker`
`@kubernetes`	`@kubernetes create deployment yaml for my react app`
`@python-tests write pytest for this function`	`@python-tests`
`@terraform`	`@terraform write config for creating an EC2 instance`

Table 9: Example Prompts for Copilot Extensions

59 Important Notes

- Ye **VS Code Marketplace extensions nahi hote** (na hi Extensions tab mein dikhte hain)
- Sirf **GitHub Copilot Chat** ke settings mein available hote hain
- Extensions **per-user basis pe install** hote hain (per project nahi)

60 When to Use Copilot Extensions?

Situation	**Use Extension?**
Check: `@docker`	Aapko Dockerfile banana hai?
Infra-as-code likhna hai?	Check: `@terraform`
Check: `@kubernetes`	Kubernetes YAML banana hai?
Test cases likhwaane hain?	Check: `@python-tests`

Table 10: When to Use Copilot Extensions

61 Agar Extensions Install nahi kiye

- Copilot general suggestions dega, **domain-specific deep knowledge nahi milegi**.
- `@docker`, `@kubernetes` jaise prompts samjhega hi nahi.
- **Productivity kam aur manual search jyada**.

62 Final Summary

This document summarized key points for using GitHub Copilot Extensions:

- Check: **Copilot Extensions = Domain-specific AI upgrade**
- Check: Add via Copilot Chat settings → Extensions tab
- Check: Use `@docker`, `@kubernetes`, etc. in prompts
- Check: Ye **VS Code extensions nahi hote** — only Copilot-specific
- Check: Boost karta hai aapki **coding + DevOps + infra writing speed**

==

Introducing Cursor AI – Smart Suggestions, Chat, and Composer Mode

63 What is Cursor AI?

Cursor ek modern **AI-first code editor** hai — jo GitHub Copilot jaise tools se **aur bhi smart aur context-aware** hai.
Cursor ka goal hai:
Let AI understand and change your entire codebase like a real pair programmer.

64 Why to Use Cursor?

65 Cursor Chat – The Smart Sidekick

65.1 What is Cursor Chat?

Cursor editor ke right sidebar mein ek AI Chat hota hai, jo:

- Aapke code ka **pura structure samajhta hai**
- Real file references ke saath answer deta hai
- Functions/Classes pe jump bhi kara sakta hai
- Errors fix, refactoring, aur explanations de sakta hai

Feature	Why It Matters
Full project context	Cursor AI pura codebase samajhta hai — isolated function nahi
Smart inline suggestions	Regular autocomplete se zyada intelligent, based on your actual code structure
Built-in AI chat	ChatGPT jaisa interface with real code awareness
Composer mode	Multiple files edit kar sakta hai ek single prompt se!

Table 11: Why Use Cursor?

65.2 When to Use Cursor Chat?

Use-Case	Prompt Example
Error ka explanation	Why am I getting this null reference error?
Code understanding	Explain how the auth middleware works
Optimization	Optimize the database query in userController
Refactoring	Split this function into smaller parts

Table 12: When to Use Cursor Chat?

66 Cursor Composer Mode – The Real Magic

66.1 What is Cursor's Composer?

Composer ek **AI assistant mode** hai jo:
Aapke **entire codebase mein changes suggest + implement** kar sakta hai, based on a single prompt.
Ye GitHub Copilot se alag isliye hai, kyunki:

- Ye **multi-file editing** karta hai
- Pure project ka context samajhkar **intelligent, interconnected changes** karta hai
- AI ke saath milke **high-level feature development** mein madad karta hai

66.2 How to Use Cursor Composer?

66.2.1 Step-by-Step:

1. Cursor editor open karo
2. Left bottom mein click karo: `Ask AI` → Select: **Use Composer**
3. Type your prompt, jaise:

Listing 22: Composer Prompt

```
Add a password reset feature to the app.
```

4. Cursor:
   - Check karega kaunsi files change karni hongi (e.g. `routes.js`, `resetController.js`, `emailService.js`)
   - Aapko ek **list of proposed changes** dikhayega
   - Aap accept/reject kar sakte ho
   - Then apply all changes in one go!

66.3 Example Prompts for Composer

Prompt	What Composer Does
Creates new route, controller, view	Add user profile page with editable bio
Migrate from Mongoose to Prisma	Refactor models and DB connections
Add route, controller, email sender code	Add forgot password functionality

Table 13: Example Prompts for Composer

66.4 When to Use Composer?

66.5 When NOT to Use Composer?

- Tiny, one-line changes → **Inline Chat ya suggestion better**
- Code understanding / debugging → **Use Sidebar AI Chat**

Situation	**Use Composer?**
	Complex feature add karna hai
Check:	Check:
Multiple files mein interconnected changes chahiye	Manual refactor boring hai
Check:	Check:
Time save karna hai with AI superpowers	

Table 14: When to Use Composer?

66.6 Combo Use: Chat + Composer

- Chat se poochho: How should I implement forgot password?
- Fir Composer mode use karo: Add forgot password to the app
- AI + You = **Supercharged productivity**

67 Final Summary

Feature	**Use it For**	**Why it Rocks**
		Cursor Chat
Understand, explain, debug code	Real-time, context-aware help	Multi-file edits, big feature changes
	Cursor Composer	
Saves hours of manual work	Small quick changes	Copilot-style speed boost
Inline Suggestions		

Table 15: Summary of Cursor AI Features

==