Step 1: Set up GitHub Copilot

Purpose: Catch issues and improve code quality during development.

Necessary items:

1. GitHub Copilot subscription
2. Supported IDE: VS Code, JetBrains, or Neovim with Copilot plugin
3. Copilot extension installed

Configuration options / best practices:

- Enable inline suggestions: Helps accept good code faster.
- Show multiple suggestions:
    - VS Code -> Settings -> GitHub Copilot: Suggestion Count → 3
    - Allows picking the clearest, most efficient snippet.
- Enable telemetry prompts: To improve suggestion quality.
- Use Copilot for tests and helper functions: Often suggests edge cases you might miss.

Workflow tip:

Accept suggestions selectively, always review for correctness.

Example: Copilot can auto-generate validation functions; check logic before commit.

Step 2: Integrate ChatGPT for PR review

Purpose: AI-assisted review for logic bugs, readability, security, and edge cases.

Necessary items:

1. OpenAI API key or ChatGPT account
2. Access to PR code
3. Optional**:** GitHub Action or script for automated submissions

Manual workflow:

1. Reviewer selects files or code snippets from PR (focus on core logic).
2. Use prompts like:
    - "Review this code for potential bugs or edge cases."
    - "Suggest improvements for readability and maintainability."
    - "Highlight any security, performance, or logic concerns."
    - "Provide example unit tests for this function."

Tips for efficiency:

- Break large files into functions or classes before sending to ChatGPT (avoids token limits).
- Include context: "This is a function in a web API endpoint."

Step 3: Optional semi-automated workflow

Purpose: Reduce manual copy-pasting by integrating ChatGPT API with GitHub Actions.

Necessary items:

1. GitHub Action triggered on PRs (pull_request)
2. Script to:
    - Extract PR diff or specific files
    - Send snippets to ChatGPT API
    - Post responses as PR comments

Recommended prompt in script:

"Please review the following Python code snippet for:
- Logic errors and edge cases
- Readability improvements
- Security and performance concerns
- Suggest concise refactors
Return results in a structured JSON format for GitHub comments."

Step 4: Practical review tips

- **Small, focused PRs:** AI works best on 100–300 lines at a time.
- **Structured prompts:** Always specify:
    - Language
    - Purpose of code
    - What to check (bugs, readability, security, performance)
- **Combine human judgment:** AI may suggest unnecessary changes—review before committing.