Generate project documentation using GitHub Copilot

6 minutes

Project documentation describes the purpose, goals, and requirements of a project. To create project documentation, you need to understand the project structure, its components, and how the components interact with each other.

GitHub Copilot's Chat view is an ideal tool for generating project documentation, because it can analyze the entire project structure and provide high-level overviews of the project. The Chat view can also be used to generate specific types of documentation, such as README files, API references, and other project-related documents.

You can use each of the Chat view modes (Ask, Edit, or Agent) to generate project documentation. Each mode has its own strengths and weaknesses. Project specifications and other constraints affect which mode should be used.

> **Important**
>
> When you use the Chat view in agent mode, GitHub Copilot may make multiple premium requests to complete a single task. Premium requests can be used by user-initiated prompts and follow-up actions Copilot takes on your behalf. The total premium requests used depends on the complexity of the task, the number of steps involved, and the model selected.

Establish project documentation requirements

Documentation types and content requirements depend on the project, the intended consumers, and the standards adopted by the organization.

For example, the README.md could include the following sections:

- Project Title: The name of your project.
- Description: A brief overview of what the project does and why it exists.
- Table of Contents: Optional, but helpful for longer READMEs.
- Installation: Instructions on how to install and set up the project.
- Usage: Examples of how to use the project, including code snippets or screenshots.

- Features: A list of key features or functionality.
- Configuration: Details on any configuration options or environment variables.
- Contributing: Guidelines for contributing to the project.
- License: Any licenses used by the distributed project.
- Credits and acknowledgments: Recognition of contributors, libraries, or resources used.
- Contact: How to reach the maintainers or project team.
- Changelog: A history of changes and updates (sometimes linked to a separate file).

GitHub Copilot Chat can help you generate project documentation that meets the specific needs of your project and its stakeholders.

Use the ask mode to generate project documentation

The ask mode can be used to analyze a workspace and then generate documentation.

Use the following process to generate project documentation using the ask mode:

1. Identify the documentation requirements and supporting resources.

    - Identify the documentation requirements for your project. Identify the types of documentation and the required document sections.

    - Identify the resources required to generate the documentation. Your code workspace could be the only required resource. However, you may need to add context to the chat for sections like "Contributing", "Credits", and "Contact".

2. Open the Chat view and start a new chat session using the ask mode.

3. Add context to the chat session.

    - You can add context to the chat session by dragging and dropping files from Visual Studio Code's EXPLORER view into the Chat view. You can also use the **Add Context** button.
    - You can open external files in the code editor to include resources that aren't part of the workspace and use them to provide more context. For example, you can open markdown files that contain contributor guidelines or contact information and then use the **Add Context** button to add them to the Chat view context.

4. Enter a series of prompts that investigate your documentation requirements.

    You can use the ask mode to analyze the workspace and build a chat session history that supports your documentation requirements. Describing your goals can help establish

context for the chat session. Asking questions that address your requirements helps GitHub Copilot identify the information needed to generate the documentation.

Refresh the added context as needed.

5. Enter a prompt that asks for suggested project documentation, listing the required sections that you identified in the first step.

   For example: "`@workspace /explain I need help creating a README file that can be used in the GitHub repository for this workspace. The file should be formatted as markdown. The README file needs to include the following sections: Project Title, Description, Table of Contents, Installation, Usage, Features, Configuration, and License.`"

6. Review the suggested project documentation, and refine the results using new prompts if necessary.

7. Move the suggested project documentation into a project documentation file.

   For example, create a README.md file at the root of the workspace, and the insert the suggested content into the file.

   You can use the ask mode to suggest updates for specific sections of your project after creating the document, or use other GitHub Copilot tools to help with updates.

Use the edit mode to generate project documentation

Edit mode is best for making changes to specific files in the project. The edit mode can be used to generate project documentation by adding context files to the chat and then creating or updating documentation files.

Use the following process to generate project documentation, such as README.md file, using the edit mode:

1. Identify the documentation requirements and supporting resources.

2. Open the Chat view and start a new chat session using the edit mode.

3. Add context to the chat session.

   Chat participants aren't available in edit mode, so you can't specify `@workspace` as part of your prompt. However, you can add context to the chat session using `#codebase` and by dragging and dropping files or folders from Visual Studio Code's EXPLORER view into

the Chat view. Use Visual Studio Code to open external files, such markdown files that contain contributor guidelines, and then use the **Add Context** button to add them to the chat context.

4. Enter a prompt to create the intended project documentation.

   For example: "I need to create a README file that can be used for the GitHub repository. The file should be formatted as markdown. Create a README.md file in the root workspace folder. The README file needs to include the following sections: Project Title, Description, Table of Contents, Installation, Usage, Features, Configuration, and License."

5. Review the README.md file created using edit mode, and then save or discard the file.

   You can update the file using prompts to correct or enhance specific sections if necessary.

Use the agent mode to generate project documentation

Agent mode is best for generating project documentation that requires an in-depth understanding of the project. Agent mode analyzes the entire project structure before it generates project documentation. By gathering information from multiple files and folders, agent mode can describe complex relationships and include links between documents.

Use the following process to generate project documentation, such as README.md file, using the edit mode:

1. Identify the documentation requirements and supporting resources.

2. Open the Chat view and start a new chat session using the agent mode.

3. Add context to the chat session.

   Chat participants aren't available in agent mode, so you can't specify `@workspace` as part of your prompt. However, you can add context to the chat session using `#codebase` and by adding workspace files and folders to the chat context. External files can be opened in Visual Studio Code and then added to the chat context using the **Add Context** button.

4. Enter a prompt to create the intended project documentation.

   For example: "Generate a collection of project documentation files. Create or update the workspace README.md file for this repository. Create or update the UsageExamples.md file. Create or update the ChangeLog.md file. Include links between the documentation

files, cross-reference classes and methods, and ensure consistency across the documents."

5. Review the document files and then save or discard the updates.

   Update the file using prompts to correct or enhance specific sections if necessary.

Agent mode capabilities

There are several documentation tasks where agent mode is the best choice.

1. Multi-file and cross-file documentation generation.
   - Agent mode can analyze the entire project structure, gather information from multiple files and folders, and generate documentation that links and summarizes content across the codebase. Ror example, generating a full API reference or a README that describes all major components.
2. Automated project analysis and summarization.
   - Agent mode can perform tasks like summarizing the architecture, identifying main classes/services, and producing diagrams or tables that require understanding relationships between files and components.
3. Dynamic content generation (for example, usage examples, class tables)
   - Agent mode can scan the project to generate usage examples, class responsibility tables, or lists of public APIs.
4. Batch documentation tasks.
   - Agent mode can execute a sequence of documentation tasks (for example, update README, create CONTRIBUTING.md, generate API docs, update changelog) in one workflow.
5. Intelligent linking and navigation.
   - Agent mode can create links between documentation files, cross-reference classes and methods, and ensure consistency across docs.

Agent mode is ideal for project-wide, multi-file, and context-aware documentation tasks that require analysis, synthesis, and coordination.

Summary

GitHub Copilot can help you generate project documentation that meets the specific needs of your project and its stakeholders. The Chat view can be used to generate project documentation in three different modes: Ask, Edit, and Agent. Each mode has its own strengths and weaknesses, and the best mode to use depends on the specific task at hand. The ask mode is best for asking questions about your codebase or technology concepts. The edit mode is best for making changes to specific files in the project. The agent mode is best for generating project documentation that requires a more in-depth understanding of the project.

Next unit: Generate inline code documentation using GitHub Copilot

Previous Next