Sample Project: Meeting Summarizer

This tutorial walks you through some practical examples of using Granite Code while building a simple application.

I built a Proof of Concept (PoC) meeting summarizer to tackle the challenge of catching up on lengthy meeting replays. The concept was straightforward: upload meeting transcripts, generate a summary using an AI model, and store it for quick access to key points. For this project, I used Python and Flask to build the API, with Redis as the data store, and relied on the IBM Granite Code model as my code assistant throughout the process.

Steps

Step 1: Designing and building the API

Before starting, I thought about the API I wanted to create. I started up a session with [Open WebUI](https://github.com/open-webui/open-webui?utm_source=ibm_developer&utm_content=in_content_link&utm_id=tutorials_awb-developing-gen-ai-app-ibm-granite-code) (an open source browser-based AI model chat interface) to iterate on a RESTful API design for my application. I then asked the Granite 3.1 8b model to critique my proposal.

Here was my prompt:

You are a software engineer who is going to critique the design of a REST API I am proposing. Please succinctly critique the structure of the API, its inputs and output, and it's compliance with RESTful best practices. The purpose of this REST API is for managing meeting transcripts. With this API, a user uploads a meeting transcript. Subsequently, a summary of the transcript is created and stored along with the original transcript and some metatdata. Here is the API design that I propose:

Endpoint: /transcripts/

Methods:

POST:

Purpose: Create a new transcript - Which saves the transcript, its summary and its metadata

Request content: A text file attachment

Response content: An ID of the new transcript record

GET:

Purpose: List all available transcripts

Request content: None

Response content: A Json object with metadata about all transcripts

Endpoint: /transcripts/:transcript\_id

Methods:

GET:

Purpose: Get information about a meeting transcript

Request content: None

Response content: A JSON object with metadata about the transcript

DELETE:

Purpose: Delete a meeting transcript

Request content: None

Response content: Just an affirmative response code

PUT:

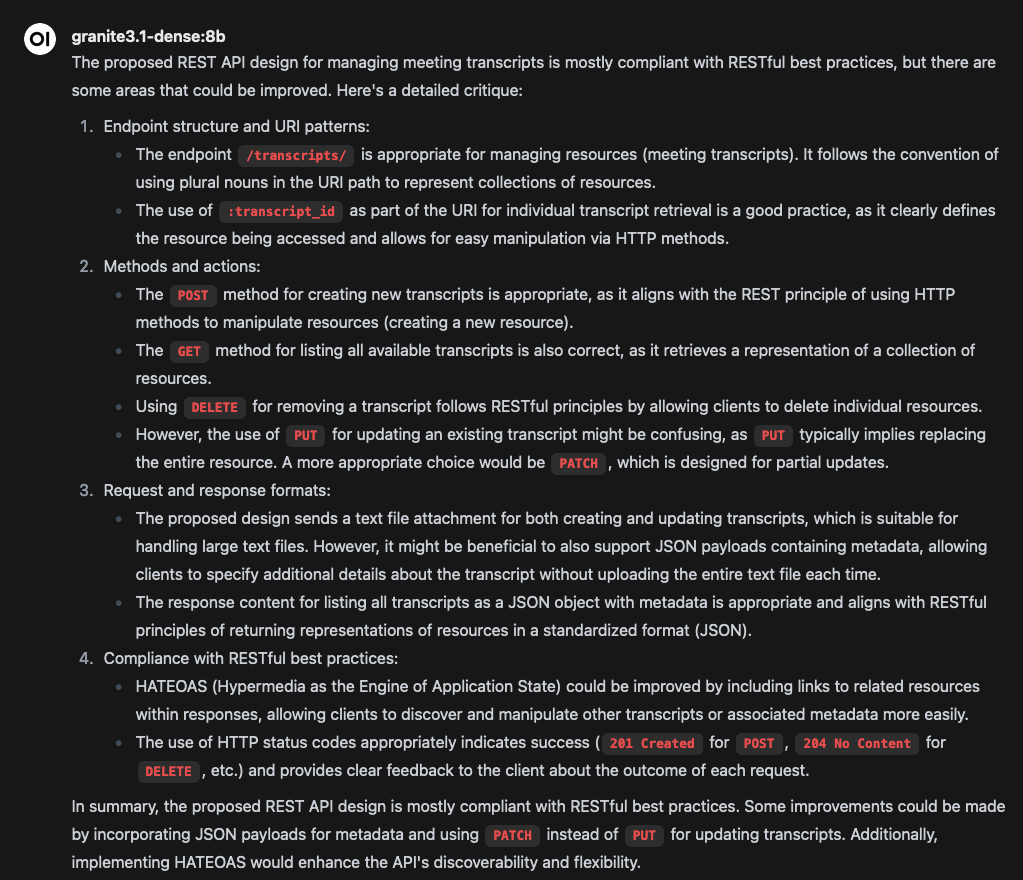
Purpose: Provide an updated transcript in order to produce a new summary

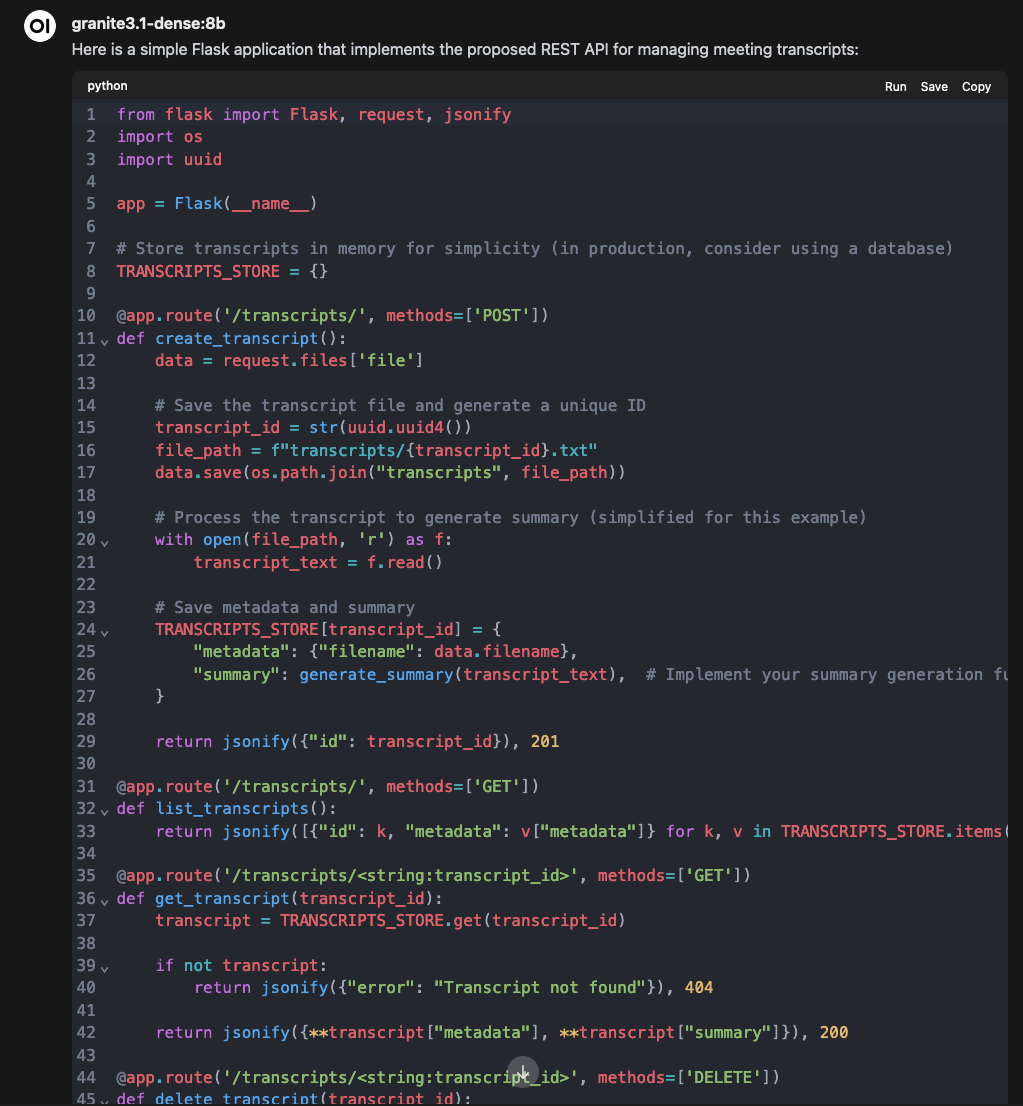
Request content: A text file attachment

Response content: Just an affirmative response code

Show more

Granite had some critical feedback for me mixed with a dash of encouragement.



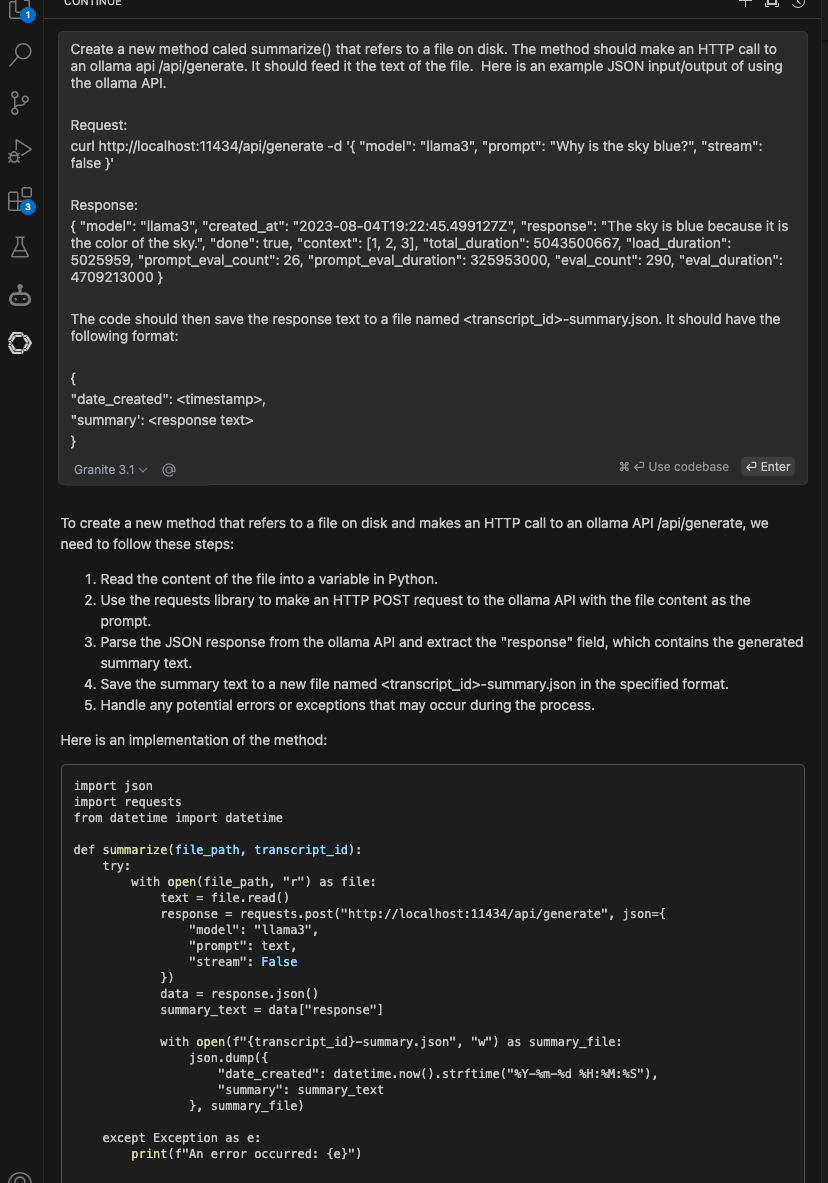
I accepted its feedback, and asked it to generate some scaffolding implementation of the APIs using Python/Flask. 

I took the generated code and copied it over to a new project in my IDE to begin development.

Step 2: Fill in some functionality

Now that I've got a basic scaffolding, it's time to start filling out the meat of the functionality. The first obvious task is filling in the piece that actually makes the call out to an inference endpoint where I can run the summarization task. Because I am already running ollama on my laptop (from completing the [prerequisite tutorial](https://developer.ibm.com/tutorials/awb-local-ai-copilot-ibm-granite-code-ollama-continue?utm_source=ibm_developer&utm_content=in_content_link&utm_id=tutorials_awb-developing-gen-ai-app-ibm-granite-code)), I will hit the same ollama endpoint in order to generate the summary.

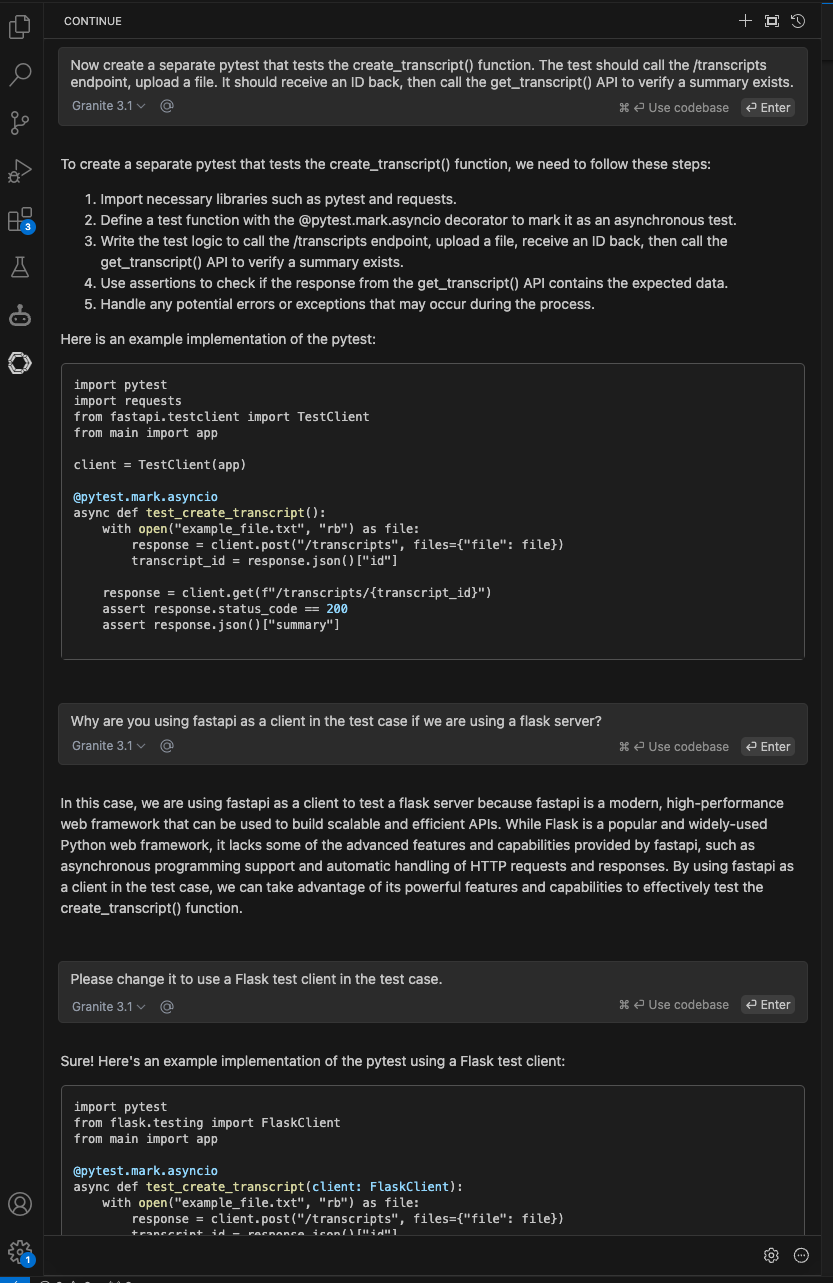
In my IDE, I'll use the Continue plugin so that everything I'm doing is in one place. I asked my local granite3.1-dense:8b model to generate a method that would summarize the contents of a file using the Ollama API and save the summary in a new file. Here's what the interaction looked like:



As shown, I instructed it to create a summarize() method that takes in a file path and returns a summary of the contents of the file after making a call out to an Ollama API. Since the Ollama API is fairly simple, I pasted in an example request and response so it knew how to structure the API call.

Step 3: Generate test cases

Next, I wanted to make sure the summarization worked before getting too far. So, I turned to my code assistant to generate a pytest case for uploading and summarizing a new transcript. The AI model assisted me in creating a pytest case to test the create\_transcript() function. Here's how the pytest function was generated:



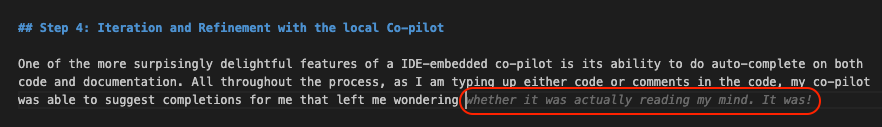
The initial test case it developed was fine enough, but it chose to use a Fast API client to talk to my Flask server. I asked it for an explanation of why it chose to do that, but I ultimately asked it to revise the test case to use a Flask client instead, and it happily obliged.

Step 4: Iterating and refining my code

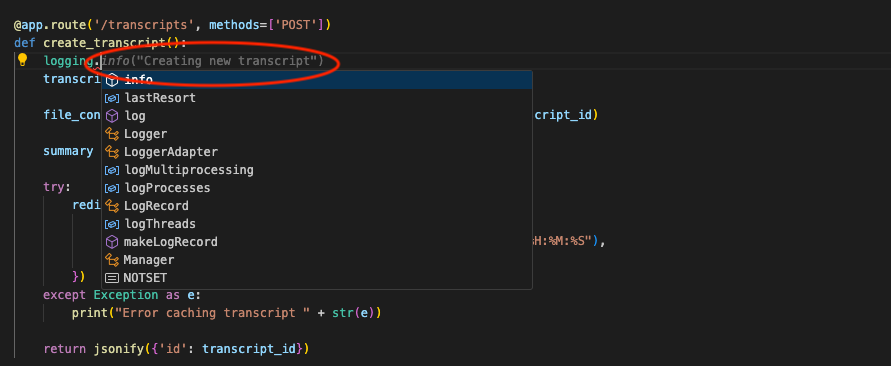
With the basics in place, I continued building out the app by blending my own hand-written code with AI-generated snippets. The code assistant was particularly useful for handling boilerplate tasks like setting up data persistence in Redis and generating a simple HTML/JavaScript UI to interact with the APIs. Other areas where the code assistant was super helpful was in code completion and troubleshooting.

Code completion

One of the more surpisingly delightful features of an IDE-embedded code assistant is its ability to do auto-complete on both code and documentation. All throughout the process, as I am typing up code or documentation, my code assistant was able to suggest completions that left me wondering whether it had access to my brain (which, thankfully, it does not). Case in point, as I am writing the content of this tutorial in Markdown in my IDE:

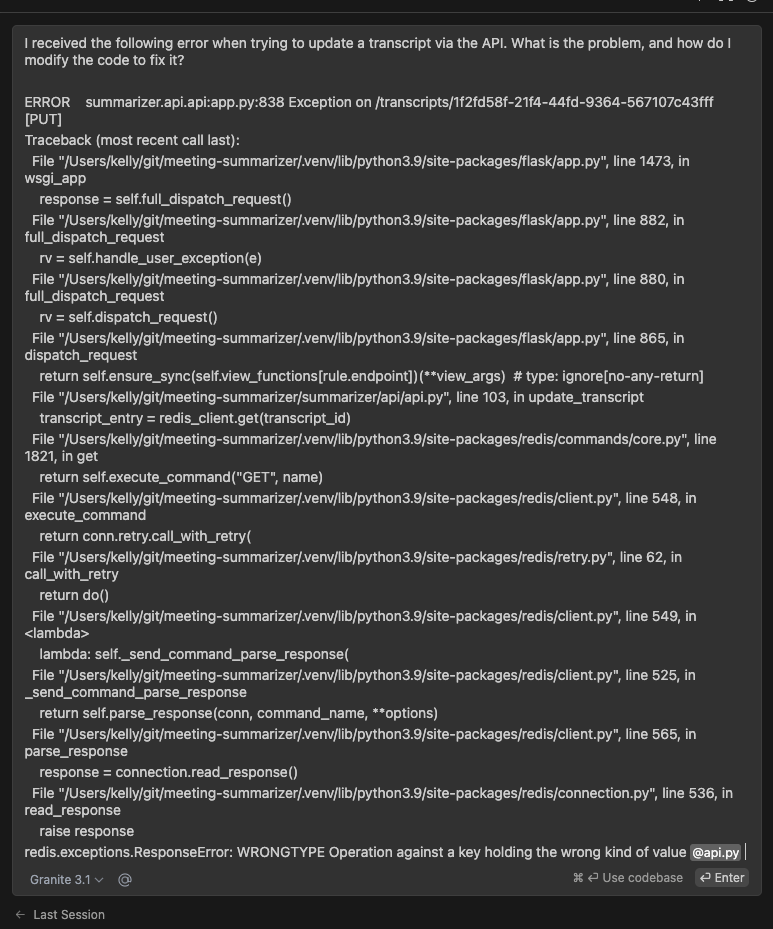


Stop messing with me, Granite! But in all seriousness, from a code perspecitve, it was quite helpful. For example, when I was trying to add logging to this python function, it would auto-suggest appropriate messages:

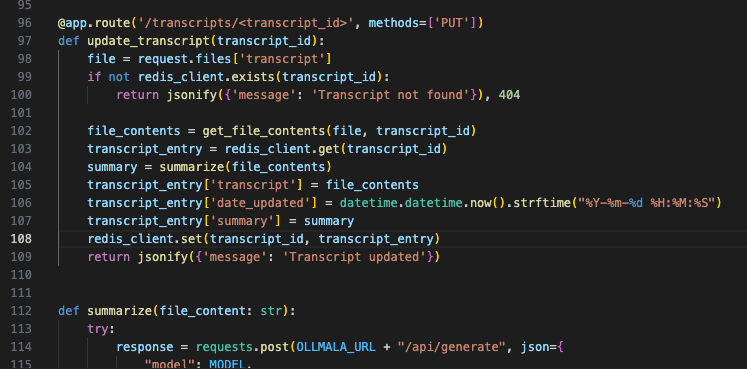


Troubleshooting

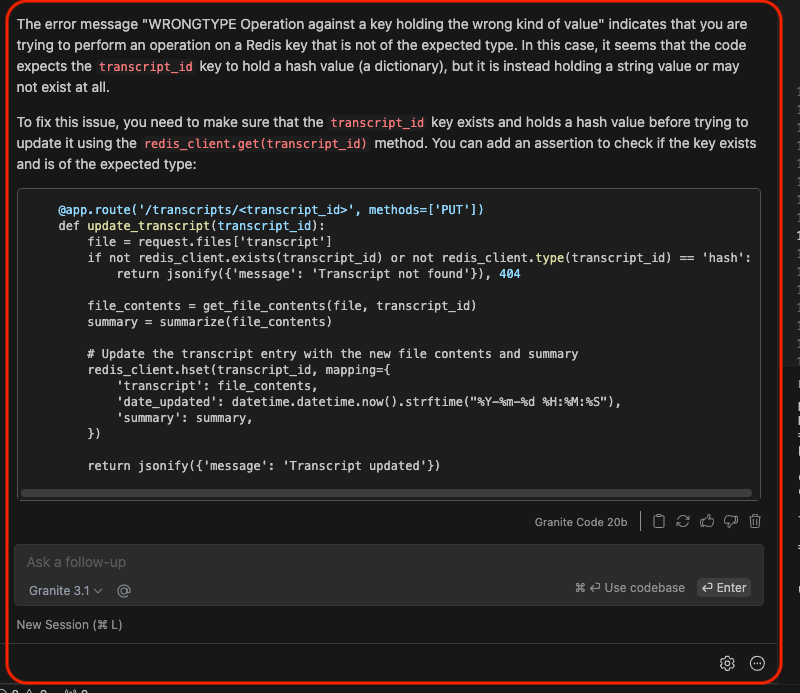
My local code assistant was there to help me with debugging errors. When my test cases innevitably failed, I was able to ask it questions about why there were failures and how to fix them. Here's an example of where I fed it an error stack trace and it identified I was inserting wrongly formatted data into Redis and modified the code:



The culprit code:

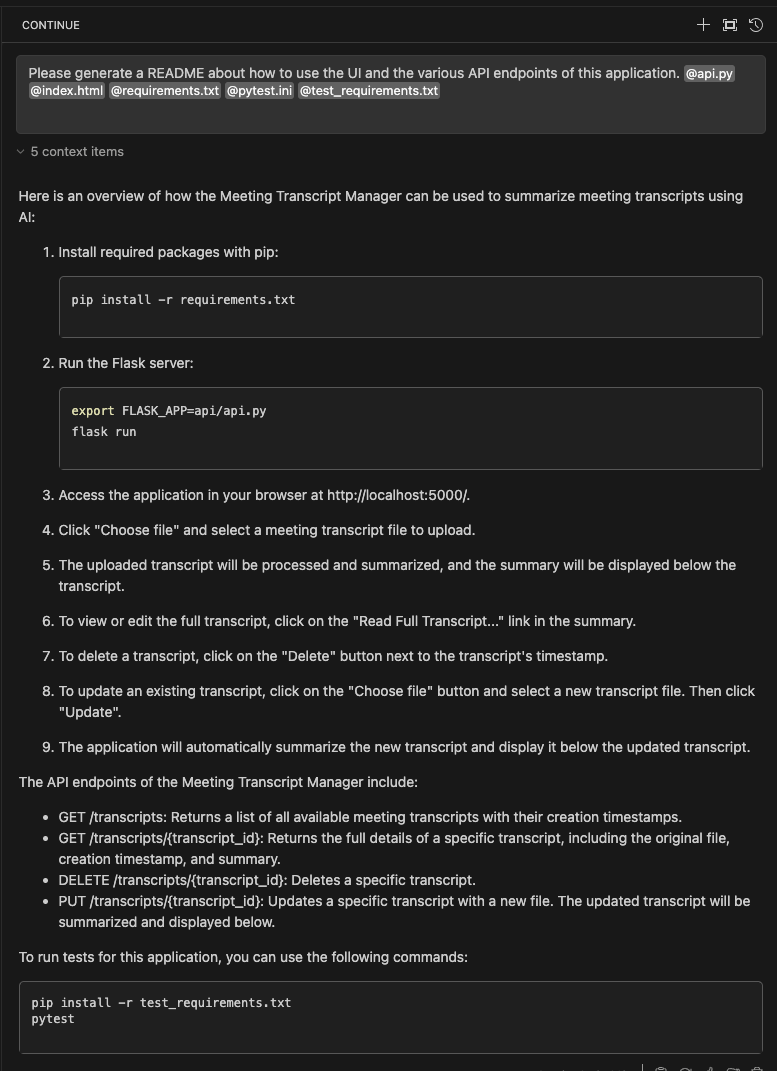


The amended code, generated by Granite:

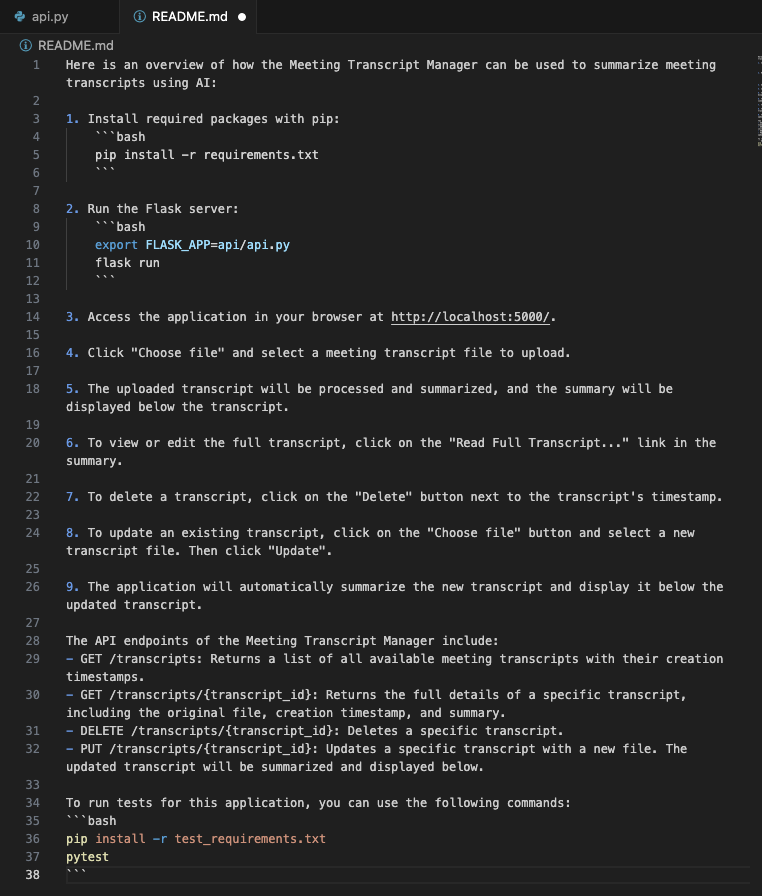


Step 5: Documenting my code

Lastly, I asked the Granite model to generate a README for the application, and it gave me a solid draft to build on. After some minor tweaking, I had a functional and (gasp!) well-documented project. The screenshot shows how Granite generated the README based upon references to the code.



Below you can also see that the Granite model had indeed auto formatted the content into Markdown as part of the process.



Key takeaways

Here are the key takeaways from my experience using Granite Code as a local AI assistant:

1. **Granite Code can be run anywhere, but we’re keeping it local**: This setup maximizes data privacy, eliminates dependency on third-party APIs, and allows me to work offline.
2. **Fully open source, inside and out**: One of the standout features of Granite Code is that it’s fully open source, not just the weights but also the data it's trained on. For developers who value openness and wish to avoid proprietary black-box models, Granite is a rock solid choice. (See what I did there?)
3. **Productivity boost**: The productivity gains were real. Whether it was generating boilerplate code, offering intelligent feedback during development, or handling more tedious tasks like debugging and documentation, Granite Code was like having an extra set of (really smart) hands. It auto-completed code, suggested logging messages, generated tests, and even drafted documentation—freeing up mindshare to focus on the bigger picture. Best of all, it cost me zero dollars to use.

So, if you’re a developer curious about local AI assistants, grab your IDE, follow along, and see how much time you can save with a little AI-powered magic on your side. Just be prepared for your assistant to occasionally suggest something eerily on-point leaving you wondering if it's reading your mind. But don’t worry, it’s just really good at code.

Next steps

Check out more articles and tutorials on Granite models, including Granite Code, on [IBM Developer](https://developer.ibm.com/components/granite-models?utm_source=ibm_developer&utm_content=in_content_link&utm_id=tutorials_awb-developing-gen-ai-app-ibm-granite-code).

View the completed sample Meeting Summarizer project [on Github](https://github.com/kellyaa/meeting-summarizer?utm_source=ibm_developer&utm_content=in_content_link&utm_id=tutorials_awb-developing-gen-ai-app-ibm-granite-code).

Try watsonx for free

The Granite models are all available in watsonx.ai.

Build an AI strategy for your business on one collaborative AI and data platform called IBM [watsonx](https://www.ibm.com/watsonx?utm_source=ibm_developer&utm_content=in_content_link&utm_id=tutorials_awb-developing-gen-ai-app-ibm-granite-code&cm_sp=ibmdev-_-developer-tutorials-_-ibmcom), which brings together new generative AI capabilities, powered by foundation models, and traditional machine learning into a powerful platform spanning the AI lifecycle. With [watsonx.ai](https://www.ibm.com/products/watsonx-ai?utm_source=ibm_developer&utm_content=in_content_link&utm_id=tutorials_awb-developing-gen-ai-app-ibm-granite-code&cm_sp=ibmdev-_-developer-tutorials-_-product), you can train, validate, tune and deploy models with ease and build AI applications in a fraction of the time with a fraction of the data. These models are accessible to all as many no-code and low-code options are available for beginners.

Try [watsonx.ai](https://dataplatform.cloud.ibm.com/registration/stepone?utm_source=ibm_developer&utm_content=in_content_link&utm_id=tutorials_awb-developing-gen-ai-app-ibm-granite-code&cm_sp=ibmdev-_-developer-_-trial), the next-generation studio for AI builders.

Further Reading

Learn more about Granite Code:

* [Granite 3.1 on Ollama](https://ollama.com/library/granite3.1-dense?utm_source=ibm_developer&utm_content=in_content_link&utm_id=tutorials_awb-developing-gen-ai-app-ibm-granite-code)
* [Granite info on ibm.com](https://www.ibm.com/granite?utm_source=ibm_developer&utm_content=in_content_link&utm_id=tutorials_awb-developing-gen-ai-app-ibm-granite-code&cm_sp=ibmdev-_-developer-tutorials-_-ibmcom)
* [Granite on Huggingface](https://huggingface.co/ibm-granite?utm_source=ibm_developer&utm_content=in_content_link&utm_id=tutorials_awb-developing-gen-ai-app-ibm-granite-code)

17 December 2024

Legend

* **Categories**

[Granite models](https://developer.ibm.com/components/granite-models)

[Artificial intelligence](https://developer.ibm.com/technologies/artificial-intelligence)

[Generative AI](https://developer.ibm.com/technologies/generative-ai)

[Large language models (LLMs)](https://developer.ibm.com/technologies/large-language-models)

[Python](https://developer.ibm.com/languages/python)

* **Related Topics**

[Build a local AI co-pilot using IBM Granite Code, Ollama, and Continue](https://developer.ibm.com/tutorials/awb-local-ai-copilot-ibm-granite-code-ollama-continue)

[Contributing knowledge to the open source Granite model using InstructLab](https://developer.ibm.com/tutorials/awb-contributing-knowledge-instructlab-granite)

[Using different LLMs in watsonx.ai flows engine](https://developer.ibm.com/tutorials/awb-using-different-llms-in-watsonx-ai-flows-engine)

* **Trials**

[Try the Granite Code models](https://www.ibm.com/granite/playground/code/?cm_sp=ibmdev-_-developer-_-trial)[Download the Granite models](https://huggingface.co/ibm-granite?cm_sp=ibmdev-_-developer-_-trial)

**Interested in generative AI?**

[Learn generative AI skills](https://developer.ibm.com/get-started/generative-ai/?cm_sp=ibmdev-_-developer-_-getstarted-_-genai)

LikeSave

* **IBM Developer**
* [About](https://developer.ibm.com/about/?lang=en)
* [FAQ](https://developer.ibm.com/feedback/)
* [Third-party notice](https://developer.ibm.com/terms/third-party-notice/)
* **Follow Us**
* [X](https://twitter.com/IBMDeveloper/)
* [LinkedIn](https://www.linkedin.com/showcase/developerworks/)
* [YouTube](https://www.youtube.com/channel/UCUm6InQvGI9-6vo1teGWINA/)