# **Building a Generative AI App with Cohere**

## **Part 1: Getting Started and Building the Application**

My goal for this project was to build a working text generation tool using a real AI model. After exploring options like OpenAI and Hugging Face, I decided to use **Cohere**, partly because I wanted to try something newer, and partly because I was curious how it compared to the more mainstream providers.

The initial setup was straightforward. I created a Cohere account, retrieved an API key, installed the cohere Python package, and began coding.

My first milestone was building a basic loop: the user enters a prompt, the script sends it to the Cohere API via generate(), and the response is printed. I wrapped the input in a loop to allow continuous interaction and added input validation so the app wouldn't crash on empty strings or strange inputs.

It was satisfying to test it with fun prompts like:

* “Once upon a time, there was a robot who…”
* “Explain photosynthesis to a 10-year-old.”
* “Write a haiku about the ocean.”

Even in this raw form, the model’s responses were surprisingly fluent. The haiku, for example, came out nicely formed and short.

## **Part 2: Debugging and Improving the Application**

Once the basics were in place, I started running into questions about control; how do I influence the tone of the responses? Can I make the model more creative? More concise? That’s where I began experimenting with generation parameters.

### **API Errors and Learning Moments**

Initially, I mistakenly assumed I could pass top\_p to the generate() method, only to be greeted by a helpful but firm error message. It turns out that Cohere’s generate() endpoint supports fewer parameters than some other models. So I focused on the supported ones: temperature (for creativity) and max\_tokens (for output length).

Another mistake was using the ClientV2 object instead of Client. The generate() function is only available through Client, so I adjusted my code and everything worked smoothly after that.

### **Making It Interactive and Configurable**

To make the tool more flexible, I built in a quick setup prompt at the beginning of the session where users can set their own values for temperature and max\_tokens. It’s a small touch, but it totally changes the experience. It feels like tuning the personality of the AI.

* With temperature = 0.3, the bot becomes concise and factual.
* At temperature = 0.9, it gets playful, poetic, even a little weird in the best way.

The final version also includes clearer error messages and better handling for invalid input. The code now feels clean, robust, and genuinely fun to interact with.

## **Part 3: Testing and Evaluating the AI**

I ran the app through a range of test prompts, some creative, some educational, and some factual, to see how well it handled different styles and topics.

| **Prompt** | **Temperature** | **Max Tokens** | **Response Summary** | **Evaluation** |
| --- | --- | --- | --- | --- |
| “Write a haiku about the ocean.” | 0.7 | 50 | Poetic, vivid imagery | 🟢 Excellent |
| “Explain recursion like I’m five.” | 0.5 | 100 | Used a ‘mirror’ metaphor | 🟢 Simple & effective |
| “What is the capital of France?” | 0.3 | 20 | “The capital of France is Paris.” | ✅ Accurate |
| “Summarize: Photosynthesis is the process by which plants, algae, and some bacteria convert light energy into chemical energy, specifically in the form of sugars like glucose. This process uses carbon dioxide and water to produce sugars and oxygen as a byproduct. It's a vital process for life on Earth, as it provides the foundation of most food chains and releases oxygen into the atmosphere” | 0.5 | 80 | Gave a brief explanation | ✅ Informative |
| “What would happen if gravity stopped?” | 0.9 | 150 | Speculative, creative chaos | 🟡 Fun, less scientific |

### **What I Learned**

**Relevance**: The model is extremely good at keeping responses on-topic, even with vague or imaginative prompts.

**Creativity**: Higher temperatures unlock some really fun outputs that are great for poetry or storytelling.

**Limits**: The model doesn’t “think” deeply. Logical reasoning and fact-checking are still weak points. It also forgets everything between prompts.

I also realized that generate() isn’t ideal for a real chatbot, since it doesn’t remember prior messages. If I wanted multi-turn conversation, I’d need to manually store and stitch together chat history as part of the prompt.

## **Final Thoughts**

This project helped me connect all the dots: from installing a package to calling an API, validating inputs, tuning generation settings, and evaluating real outputs. More importantly, it made generative AI feel accessible, not some abstract concept, but a tool I can shape and use.

If I keep working on this, I’d like to:

* Add a logging system to track all prompts and responses
* Build a web front-end using Streamlit or Gradio
* Try building a chat-like memory loop with appended histories