Table of Contents

[Gen AI Hackathon: Cheat Sheet 2](#_Toc180320806)

[Open AI Fine-tuning vs Prompt + RAG vs Function Calling 2](#_Toc180320807)

# Gen AI Hackathon: Cheat Sheet

## Open AI Fine-tuning vs Prompt + RAG vs Function Calling

1. **Fine-Tuning**

* **What It Is**: Fine-tuning involves taking a pre-trained model like GPT and training it further on specific data to adjust its outputs to be more suited to your use case.
* **Use Cases**:
  + When you have a large amount of domain-specific data.
  + When you need very precise outputs (e.g., legal document generation, industry-specific language, etc.).
  + When your application needs to maintain consistency and accuracy with highly specialized tasks.
* **Advantages**:
  + Custom responses based on your domain or task.
  + Reduced need for elaborate prompts since the model "knows" the data context.
  + Potentially faster responses since the model has been pre-trained on your specific needs.
* **Disadvantages**:
  + Requires time and effort to gather, clean, and process training data.
  + Fine-tuning can be computationally expensive and time-consuming.
  + General applicability may reduce if too specialized.

1. **Prompt + RAG (Retrieval-Augmented Generation)**

* **What It Is**: RAG involves using a retrieval system that fetches relevant documents or data from a knowledge base in real-time and feeds it into the model along with a prompt. The model generates outputs based on both the prompt and the retrieved information.
* **Use Cases**:
  + When you need real-time or dynamically updated information (e.g., customer support, KYC, or address verification).
  + When your application relies on large, up-to-date knowledge bases (e.g., search engine-like tasks).
  + When you don’t want to fine-tune a model but still need domain-specific, accurate responses.
* **Advantages**:
  + Access to vast amounts of information without needing to train the model.
  + Flexible, as you can swap or update the data source without retraining.
  + Works well with changing or evolving datasets.
* **Disadvantages**:
  + More complex architecture as it involves both retrieval systems and language models.
  + Responses can be slower since it includes both retrieval and generation steps.
  + Requires a well-maintained and organized knowledge base for optimal performance.

1. **Function Calling**

* **What It Is**: Function calling allows the model to interact with your system's functions and APIs to retrieve structured data or perform actions. You define specific functions that the model can "call" based on the input it receives.
* **Use Cases**:
  + When you need structured outputs like database queries, API calls, or custom workflows.
  + When combining natural language interaction with specific system behaviors (e.g., ordering items, retrieving account balances, triggering actions in applications).
* **Advantages**:
  + Produces structured, predictable results.
  + Highly customizable to interact with existing systems (e.g., databases, APIs).
  + Reduces complexity in prompt crafting as the model learns to call the right functions.
* **Disadvantages**:
  + Requires setting up functions and making sure they align with the use case.
  + Limited to structured outputs and predefined tasks, so it can’t adapt as well to open-ended tasks.
  + Can add complexity to the system in terms of handling multiple function calls.

**Comparison:**

| Feature | Fine-Tuning | Prompt + RAG | Function Calling |
| --- | --- | --- | --- |
| Customization | High, tailored to specific data | Flexible, based on external data | Custom, based on system functions |
| Setup Complexity | High (data prep, training) | Medium (setup retrieval system) | Medium (define and expose functions) |
| Response Flexibility | Medium (trained for specific task) | High (can fetch new data in real-time) | Medium (structured outputs) |
| Response Speed | Fast (pre-trained model) | Slower (retrieval + generation) | Fast (calls functions directly) |
| Use Case Examples | Legal text generation, content moderation | Address verification, search tasks | API integration, database queries |
| Cost & Resource Intensive | High (compute and time for training) | Medium (depends on retrieval scale) | Low-Medium (depends on function complexity) |

**Summary**

* **Fine-tuning**: Best if you need the model to have deep, consistent knowledge of a specific domain or context.
* **Prompt + RAG**: Ideal when working with dynamic information that needs real-time retrieval, like verification systems.
* **Function Calling**: Best when you need precise, structured outputs (like querying a database or interacting with APIs) and want to combine natural language inputs with system-specific actions.