**Setting Up an AI Agent: Step-by-Step Guide**

This guide explains how to create a simple AI agent that can respond to questions using LLMs like OpenAI GPT, Anthropic Claude, or Groq LLM. It focuses on setup, workflow, and potential upgrades.

**Project Setup**

1. **Create a new folder** for your AI agent project.
2. Inside this folder, create the following files:
   * requirements.txt → to list all necessary Python libraries.
   * .env → to securely store your API keys.
   * tools.py → to define the tools your AI agent can use (search, Wikipedia, save content, etc.).
   * main.py → for the main application logic and GUI interface.

**Python Dependencies**

Add these dependencies to requirements.txt:

* langchain → framework for building LLM-based agents.
* wikipedia → access Wikipedia summaries.
* langchain-community → community utilities for LangChain.
* langchain-openai → connect to OpenAI GPT models.
* langchain-anthropic → connect to Claude models.
* python-dotenv → load API keys from .env.
* pydantic → input/output validation for tools.
* ttkbootstrap → modern GUI for Tkinter apps.

**Installation:** Use pip install -r requirements.txt in the project folder.

Optional but recommended: use a virtual environment for isolation and dependency management.

**API Keys Setup**

1. Create a .env file to store your API keys:
   * OPENAI\_API\_KEY → OpenAI GPT API key.
   * ANTHROPIC\_API\_KEY → Claude API key.
   * GROQ\_API\_KEY → optional Groq LLM key.
2. Get keys from:
   * OpenAI: [platform.openai.com/api-keys](https://platform.openai.com/api-keys)
   * Anthropic (Claude): [console.anthropic.com/settings/keys](https://console.anthropic.com/settings/keys)

Keep your API keys private and never share them publicly.

**Tools Setup**

1. Create functions to perform tasks:
   * Search online information.
   * Retrieve Wikipedia summaries.
   * Save research content.
2. Define input validation schemas to ensure your tools receive correct data.
3. Wrap tools in a safe-call mechanism to handle different types of input.

**Main Program Workflow**

1. Load environment variables (API keys).
2. Initialize the chosen LLM (OpenAI, Claude, or Groq).
3. Create an AI agent using LangChain that can call tools when needed.
4. Build a modern GUI using Tkinter with ttkbootstrap:
   * Chat area with scrollbars and color-coded user/AI messages.
   * Input box with automatic word wrapping.
   * Send button and **Enter key** to send messages.
   * Clear button to reset the conversation.
   * Timestamps for all messages.
5. Ensure the GUI is **responsive** and resizable, so all elements are visible on different window sizes.

**Testing the AI Agent**

1. Activate your virtual environment.
2. Run the main program.
3. Test by typing a question (e.g., "Why don’t people ride cows?").
4. Check the AI’s response and tool outputs.

**Skill Level Required**

* **Beginner/Intermediate Python:** You need basic understanding of Python syntax, virtual environments, and package management.
* **Optional Intermediate Knowledge:** Understanding APIs and environment variables.
* **Optional Advanced:** Knowledge of LangChain, LLMs, and GUI design for further customization.

**Upgrade Possibilities**

* Integrate real-time Wikipedia or web search APIs.
* Add more tools for math, code execution, or other knowledge domains.
* Switch between different LLMs dynamically (OpenAI GPT, Claude, Groq, etc.).
* Save chat history to a file or database.
* Add multi-theme support (dark/light mode).
* Create a web version using frameworks like Flask or FastAPI.
* Implement user authentication for multi-user support.

**Summary**

This AI agent project provides a **modular, expandable structure**:

* Tools are separated from the main program for flexibility.
* GUI is modern and responsive.
* API keys are securely managed in .env.
* Designed for upgrades and adding new LLMs or tools in the future.