### How to quickly build a GPT chatbot with data retrieval and other complex functionalities — and some best practices

### TL;DR

* Over the last 2 months, I embarked on a journey to build a GenAI chatbot for Innovation. I learned a lot and share my learnings below.
* No deep technical knowledge required; designed to help anyone build a unique chatbot in a few hours
* Article covers tech stack, key terminology, comprehensive code snippets, additional functionalities and many learnings/tips
* Streamlit is used for the UI/UX, OpenAI is used for chat completions, and various other libraries to support additional functionalities
* My goal is to encourage you to bring your ideas to life — it’s truly never been easier than today

### ****Key Concepts for Understanding Chatbots****

* **Generative AI:** The type of AI under the hood of the most cutting-edge chatbots, and the technology we’ll be mastering in this article
* **LLM (Large Language Model):** A type of deep learning model architecture trained on vast amounts of text data to understand and generate human language output.
* **Prompt:** The input given to a language model, based on which it generates a response.
* **Prompt Engineering:** The art of crafting effective prompts to guide the AI’s responses in a desired direction.
* **Token:** The smallest unit of text that a language model can understand. A token can be as short as one character or as long as one word.
* **Temperature:** A parameter that controls the randomness of the AI’s output. Higher temperatures result in more random answers.
* **Knowledge Base:** A collection of information that a chatbot can access to provide relevant facts, data, and context to conversations.
* **Embedding:** The mathematical process of converting text into numerical representations that capture semantic meaning, allowing them to be compared and analyzed by an LLM.
* **Vectorization:** Transforming text into vector embeddings using machine learning algorithms. This numerical representation of text is used when indexing and querying knowledge bases.
* **Indexing:** Analyzing and organizing a knowledge base into an efficient lookup structure to enable quick retrieval of relevant information.
* **Querying:** Searching a knowledge base for specific information related to the current conversation and returning relevant excerpts.

### ****Recommended Chatbot Tech Stack****

In-depth details on on the tech stack used to build the chatbot are listed below. Largely, the chatbot will be coded in Python and will utilize Streamlit for UI/UX and hosting, OpenAI for response generation, and a few other libraries for more complex chatbot functionalities. These are purely recommendations, and many other alternatives exist.

* [**Streamlit**](https://streamlit.io/)**:** A Python library used for creating interactive web applications. It’s used in this project to build the chatbot’s user interface.
* [**Streamlit Cloud**](https://streamlit.io/cloud)**:** A platform for deploying, managing, and sharing Streamlit apps. It’s used to host the chatbot.
* **Python:** The programming language used to write the chatbot’s code.
* **Langchain:** A Python library for natural language processing tasks.
* **NLTK:** The Natural Language Toolkit is a leading platform for building Python programs to work with human language data.
* **Sklearn:** A machine learning library for Python.
* **Stopwords Library:** A library used to remove common words (like ‘the’, ‘a’, ‘in’) that do not carry much meaning and are often removed from texts.
* **GPTSimpleVectorIndex, LLMPredictor, PromptHelper:** These classes are part of the gpt\_index module and are used for indexing, predicting, and crafting prompts.