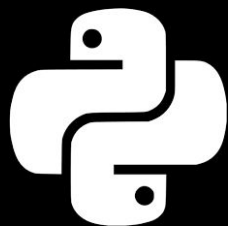


# LLMs for me



**Case Study in LLMs &  
Generative AI**

**llmsfor.me**



Myles Harrison,  
AI Consultant & Trainer



**February 11th, 2025**

*NLP from scratch* 

# Agenda

**01**

**Introduction - Looking Back**

**02**

**A Case Study in GenAI**

**03**

**Building a GenAI-powered App**

**04**

**Conclusion - Looking Forward**



**Introduction -  
Looking Back**

# TABLE OF CONTENTS

**01**

**INTRO TO LLMS**



**02**

**FINE-TUNING**



**03**

**OPENAI & GPT**



**04**

**LOCAL LLMS**



**05**

**MULTIMODAL LLMS**



**06**

**CASE STUDY FOR LLMS**





# A Case Study in GenAI

# GenAI Case Study – from the client

“

*Our healthcare network, **Devil May Care Health Clinics**, struggles with high call volumes for appointment scheduling, insurance inquiries, and general patient support, leading to long wait times and overburdened staff. We need an **AI-powered chatbot** for our website and patient portal to streamline communication, provide 24/7 assistance, and integrate with our EHR and scheduling systems while ensuring HIPAA compliance. The chatbot must **enhance patient experience**, reduce **administrative workload**, and **intelligently escalate complex issues to human staff**. Can you develop a secure, intuitive solution that meets these needs?* ”

- Dr. Susan Church, Chief Operating Officer at Devil May Care Health Clinics



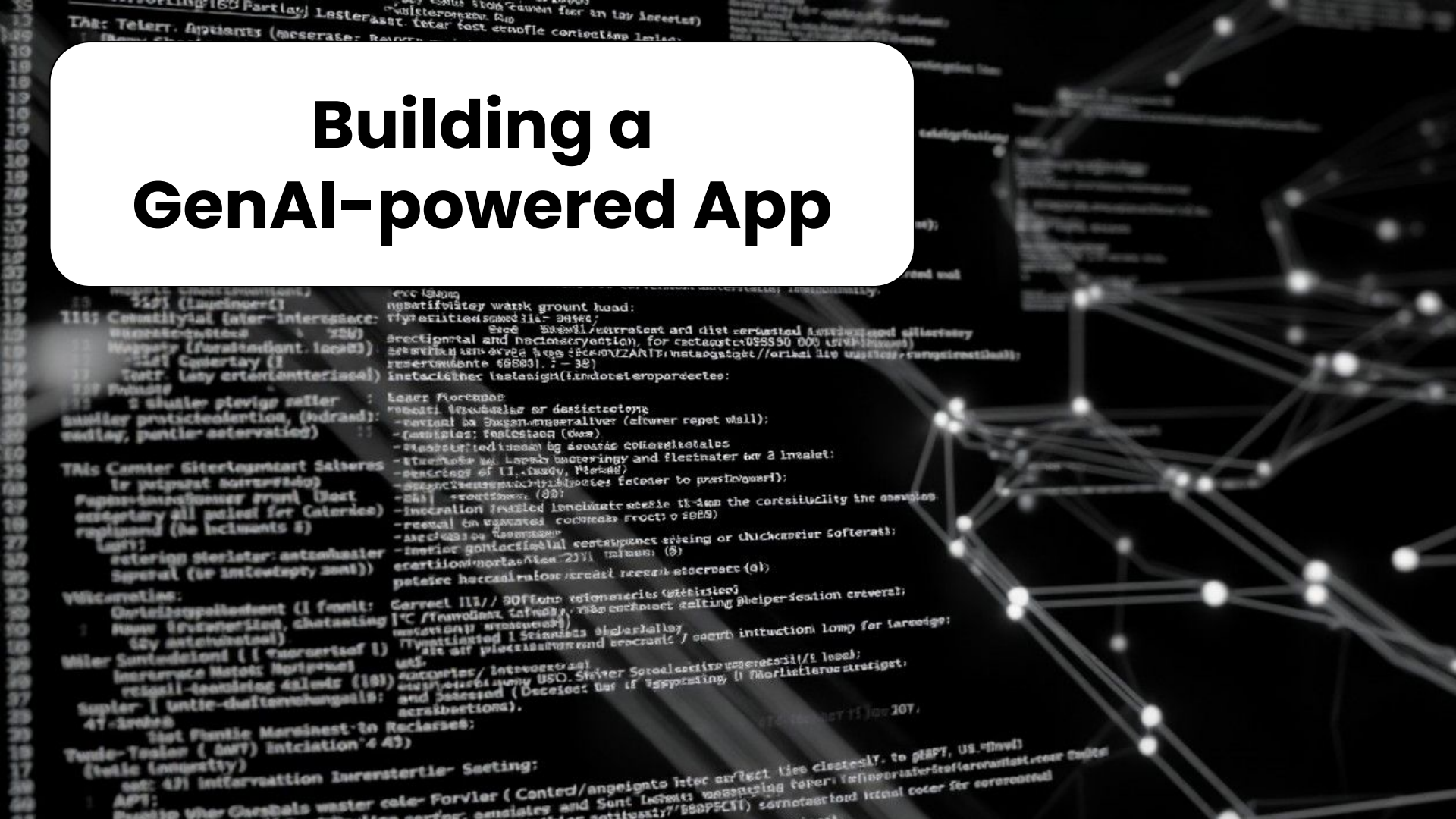
# Requirements Gathering

This is actually the hardest part.

Don't skip this step -  
or you'll wish you hadn't later.



# Building a GenAI-powered App

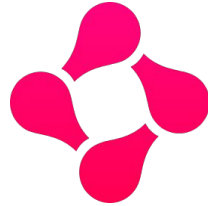




# Python (Web) App Frameworks for chat



Streamlit



Chainlit

# “Hello World” in Streamlit

```
import streamlit as st

# Title of the app
st.title("Hello, World! in Streamlit")

# Displaying text
st.write("Welcome to your first Streamlit app!")
```

```
streamlit run app.py
```

app.py

> terminal

[Cloud](#)[Gallery](#)[Components](#)[Generative AI](#)[Community](#)[Docs](#)[Blog](#)[Sign in](#)[Sign up](#)

## GenAI & LLMs in Streamlit

[streamlit.io/generative-ai](https://streamlit.io/generative-ai)

[docs.streamlit.io/develop/tutorials/chat-an-d-llm-apps/build-conversational-apps](https://docs.streamlit.io/develop/tutorials/chat-an-d-llm-apps/build-conversational-apps)

# Build powerful generative AI apps

Thousands of developers use Streamlit as their go-to platform to experiment and build generative AI apps. Create, deploy, and share LLM-powered apps as fast as ChatGPT can compute!

[Try example code](#)

[Deploy on Community Cloud](#)

# st.chat\_input

```
import streamlit as st

# Title of the app
st.title("Hello, World! in Streamlit")

# Displaying text
st.write("Welcome to your first Streamlit app!")

# Create a chat input
st.chat_input("Say something")
```



## Hello, World! in Streamlit

Welcome to your first Streamlit app!

Say something

(creates GUI element, but doesn't do anything...)

[docs.streamlit.io/develop/api-reference/chat/st.chat\\_input](https://docs.streamlit.io/develop/api-reference/chat/st.chat_input)

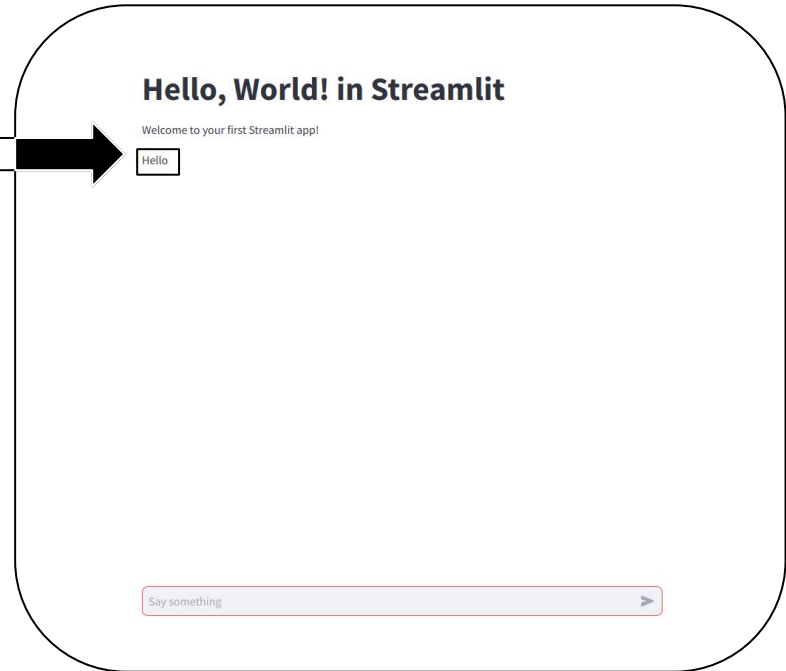
# Chat elements are methods

```
...  
# Create a chat input & print result to stdout  
prompt = st.chat_input("Say something")  
print(prompt)  
st.markdown(prompt)
```

Writes output once:

- `print()` sent to standard out (terminal)
- `st.markdown` writes to streamlit app GUI

[docs.streamlit.io/develop/api-reference/chat/st.chat\\_input](https://docs.streamlit.io/develop/api-reference/chat/st.chat_input)



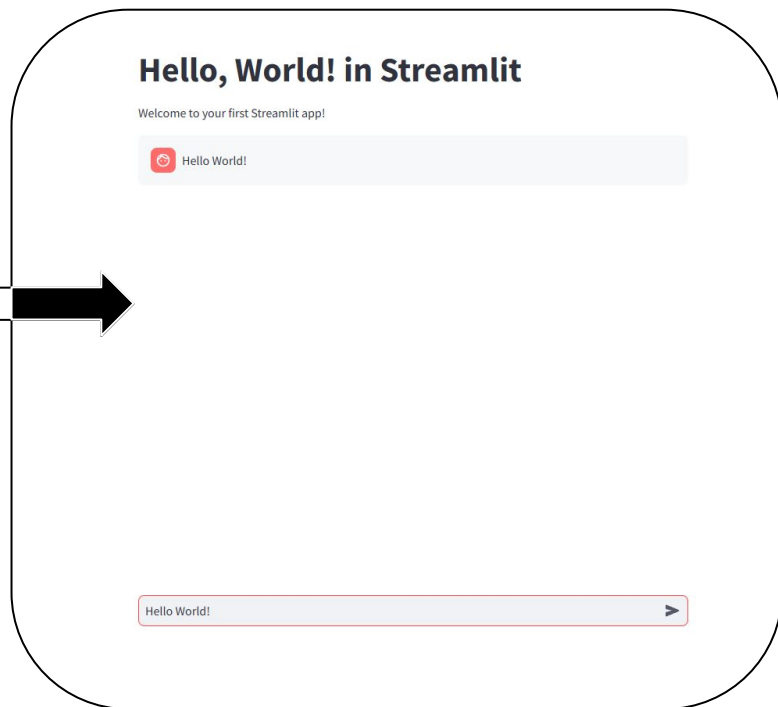
# Combine with `st.chat_message`

```
...  
# Create a chat input  
if prompt := st.chat_input("Say something"):  
    with st.chat_message("user"):  
        st.markdown(prompt)
```

Now we have “chat”, however this also overwrites the very first message each time.

Streamlit constantly reruns the application to change state continuously.

[docs.streamlit.io/develop/api-reference/chat/st.chat\\_input](https://docs.streamlit.io/develop/api-reference/chat/st.chat_input)



# Add LLMs 🚀 (1 of 2)

```
import streamlit as st
from openai import OpenAI
```

Import OpenAI client

```
# Instantiate the client
client = OpenAI()
```

Instantiate client

```
# Title of the app
st.title("Hello, World! in Streamlit")
```

```
# Displaying text
st.write("Welcome to your first Streamlit app!")
```

} Previous code ("user" role)

```
# Create a chat input
if prompt := st.chat_input("Say something"):
    with st.chat_message("user"):
        st.markdown(prompt)
```

# Add LLMs 🚀 (2 of 2)

```
# Get the response from OpenAI
response = client.chat.completions.create(
    messages=[{"role": "user", "content":
prompt}],
    model="gpt-4o-mini"
)
response_text = response.choices[0].message.content

# Write out
with st.chat_message("assistant"):
    st.markdown(response_text)
```



Get response from GPT-4o-mini



“assistant” response



# Customizing our app

- Add system prompt to messages array
- Changing user and assistant avatars:  
<https://discuss.streamlit.io/t/change-default-avatars-in-st-chat/49142>
- Changing CSS and styling:  
[https://github.com/BugzTheBunny/streamlit\\_custom\\_gui](https://github.com/BugzTheBunny/streamlit_custom_gui)

## Devil May Care Clinic Chatbot

Welcome to the Devil May Care Clinic Chatbot! Ask me anything about your health and I'll do my best to help you out.



Can you tell me about the clinic?



The Devil May Care Clinic is dedicated to providing comprehensive healthcare services with a focus on patient well-being and accessibility. Our mission is to ensure that every patient receives personalized care tailored to their unique needs. We offer a wide range of services including primary care, mental health support, preventive care, and specialized treatments.

Our team of healthcare professionals is committed to maintaining a welcoming and friendly environment, ensuring that patients feel comfortable and supported during their visits. Please let me know if you have any specific questions about services, appointments, or anything else!

say something





**Conclusion -  
Looking Forward**

# So much to learn...



LangChain



LangGraph



DSPy *Programming—not prompting—LMs*



LlamaIndex



Weights & Biases

DeepEval.

\$ the open-source LLM evaluation framework

Lumigator



Milvus



haystack  
by deepset



Pinecone



Chroma



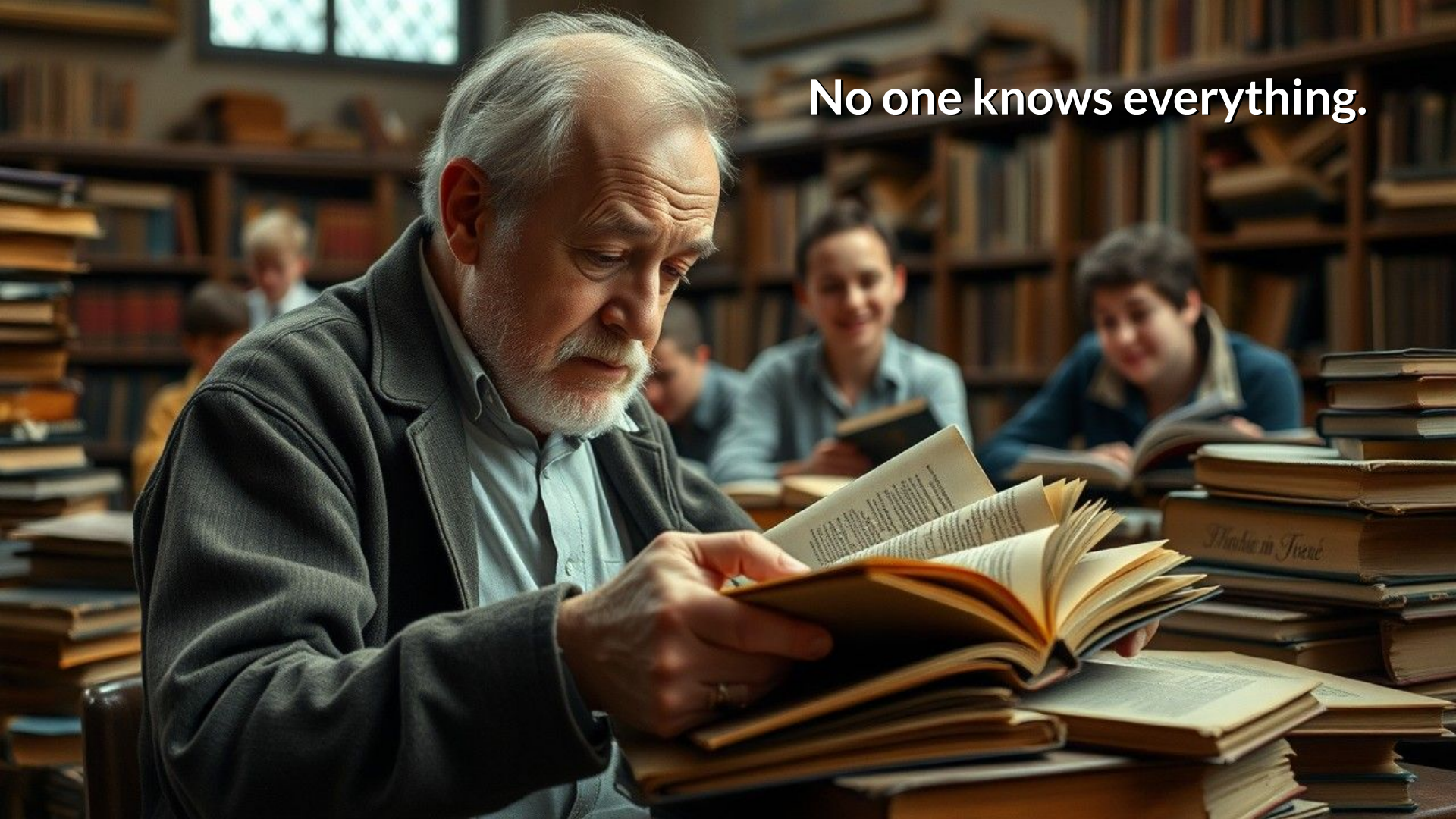
Weaviate



DuckDB

*NLP from scratch* 

No one knows everything.



# Questions?

That's it - for now 😊

I hope you enjoyed the course. If you would like to contribute to allow me to develop more content, please consider contributing at:

[nlpfromscratch.com/pwyc](https://nlpfromscratch.com/pwyc)

I will also send out a feedback form, as the course has concluded - let me know what could be improved for future courses and what you are most interested in learning next.

Thanks for coming!

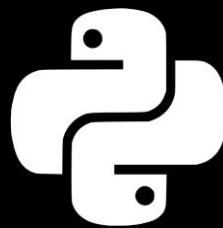


# End of Part 6

[LLMsfor.me](https://llmsfor.me)

PWYC Microcourse in LLMs and Generative AI  
January 2025

**Part 6 – Multimodal LLMs and Frameworks**  
**Tuesday, February 11th, 2025**



[llmsfor.me](https://llmsfor.me)

*NLP from scratch*