devfest

Implementing Generative AI FAQ Bot on own documents

Dr. Yogesh Haribhau Kulkarni (GDE – ML)

Google Developer Groups
Pune, India

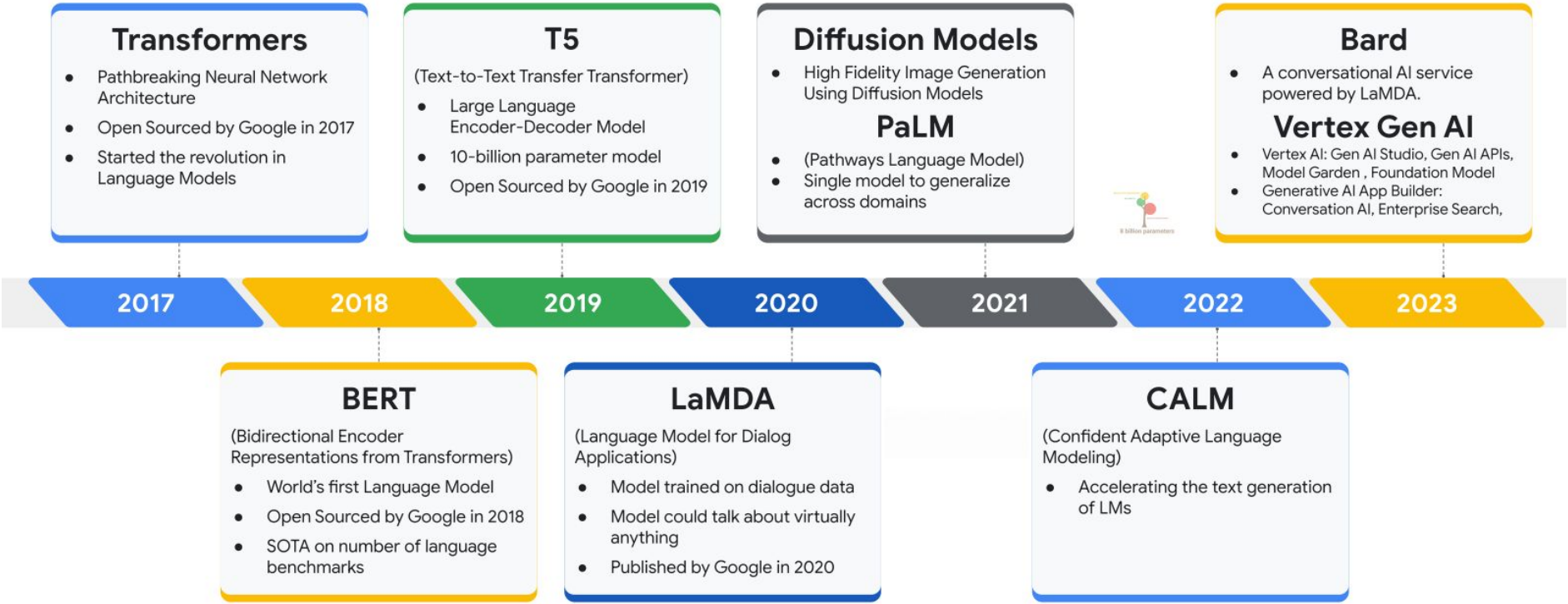
Google Developer Groups

Generative AI

Era of Generative AI

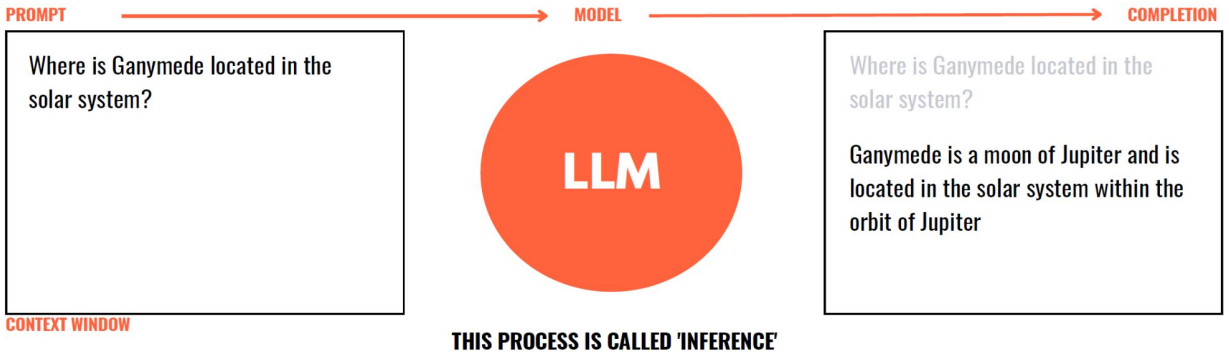
- Generative AI: Branch of AI for creating original content based on existing data patterns.
- Applications: Images, videos, text generation.
- Examples:
  - Text Generation: PaLM - articles, stories, poetry.
  - Image Generation: Stable Diffusion - images of people,animals.
  - Music Generation: MuseNet - original music in various genres.

This revolution started at Google …



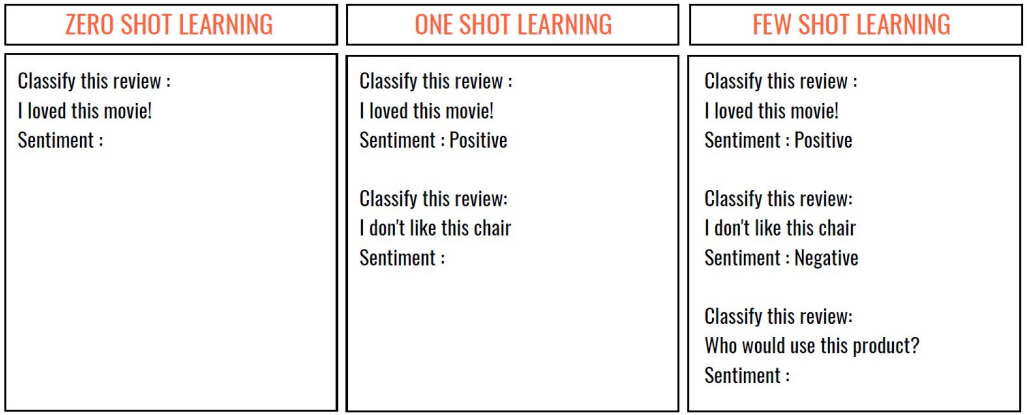
New Programming Language: English

- Prompt: The prompt is your text input that you pass to the model.
- Prompt Design: The art and science of figuring out what text to feed your language model to get it to take on the behavior you want.



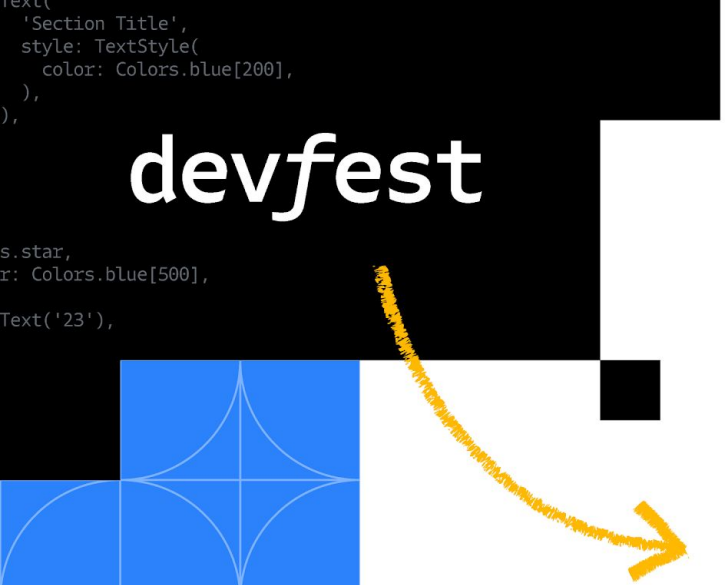
Shots

- Zero-shot prompt: The model is provided with no example when prompting for response.
- Few-shot prompt: Few-shot prompts are like one-shot prompts, but the model is given multiple labeled examples of the task.



Gen AI Products/Services on GCP

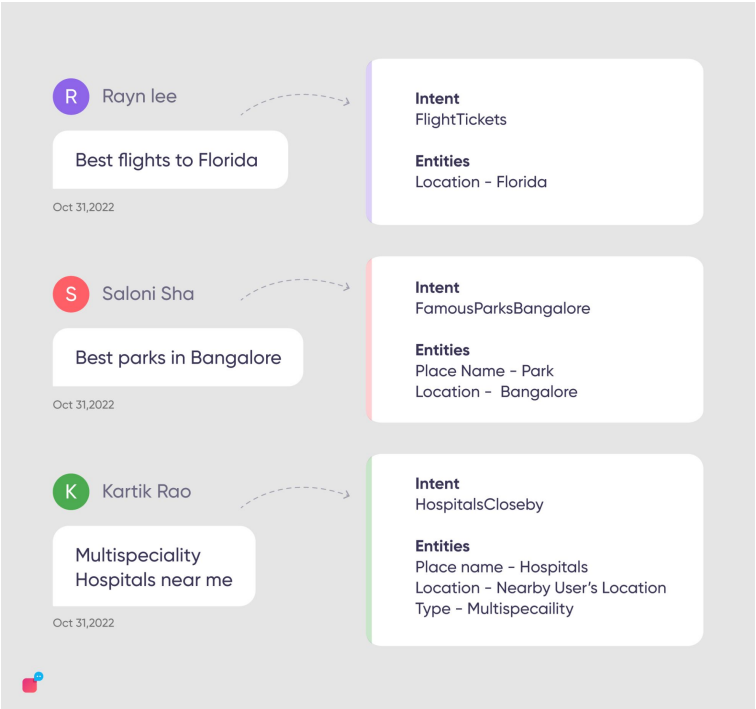
- **Generative App Builder**: Offers out-of-the-box solutions for search and conversational experiences.
- **Vertex AI Integration**: GenAI services integrated with Vertex AI for end-to-end ML platform usage
- **Vertex AI Palm APIs via Langchain**



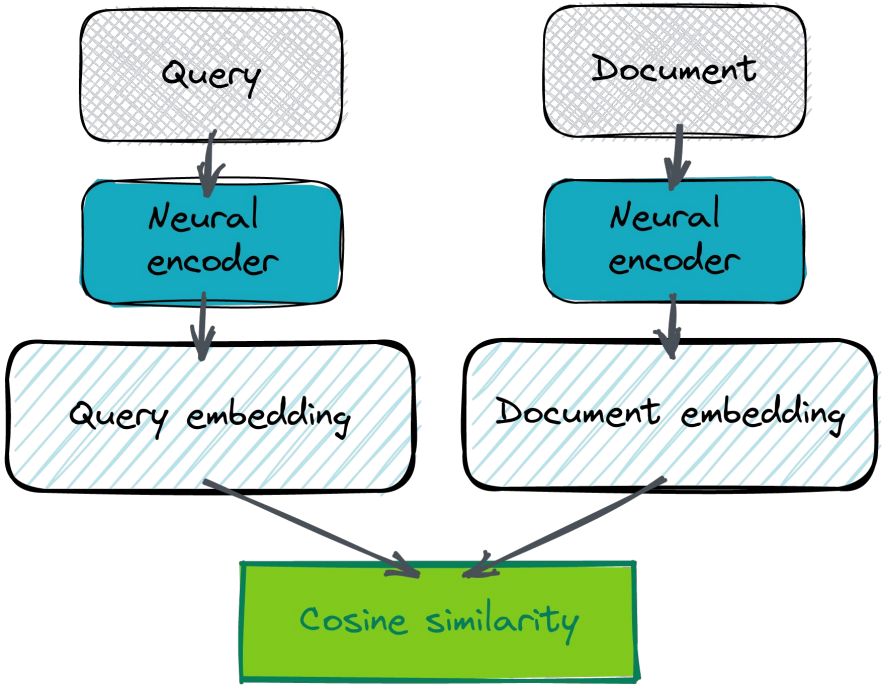
Google Developer Groups

QnA :: Bot

By Intents and Entities



By Similarity



Google Developer Groups

Retrieval Augmented Generation (RAG)

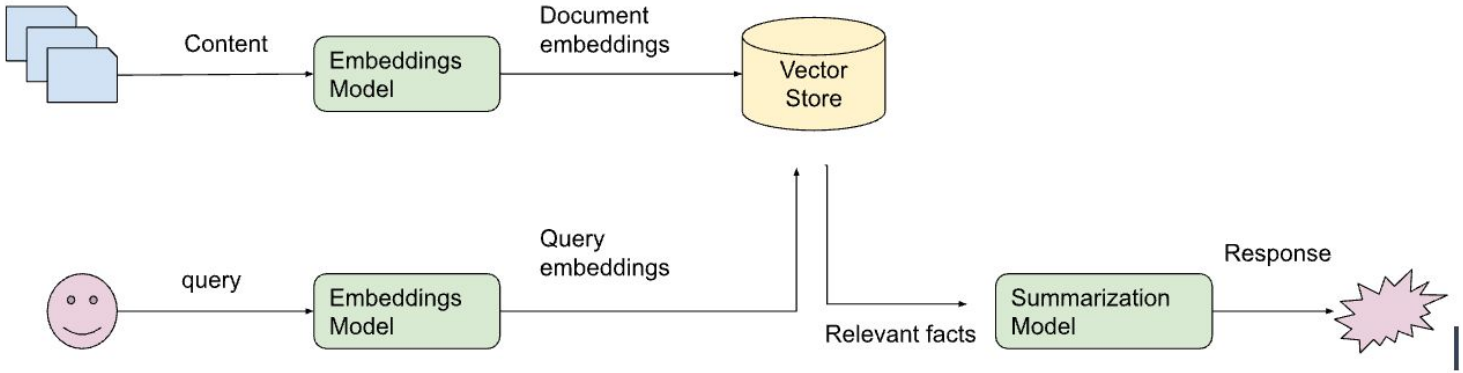
Need for RAG

- Token limit on prompt: 2k, 10k, … (not 100 books!!)
- Domain Knowledge: Generic, Wikipedia, … (not Medical, Legal)
- Old data: Sept 2021? Or a day old… (not latest for sure)
- So, RAG is needed for Custom Data

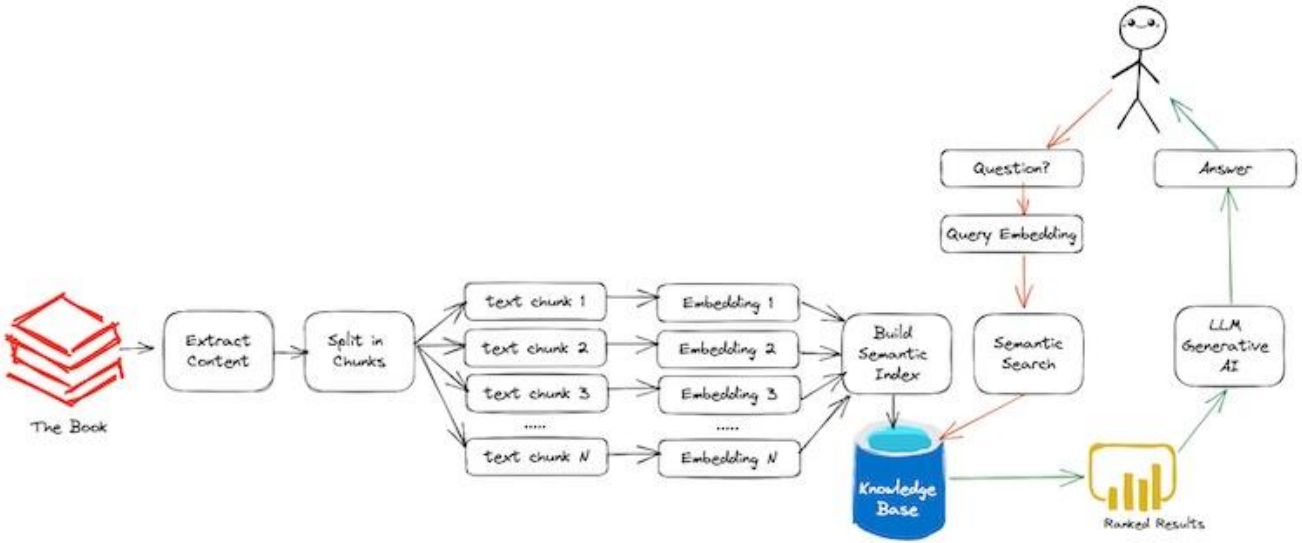
RAG

- The Retrieval Augmented Generation (RAG) framework overcomes these issues by connecting LLMs to external data sources and applications.
- RAG provides LLMs access to data they did not see during training, improving relevance and accuracy of completions.
- Brings custom context to the prompt via vector database.
- Implementing RAG involves considerations such as the size of the context window and the need for data retrieval and storage in appropriate formats
- Chunking + guardrails are key.

RAG Workflow



QnA by RAG


The Book
Extract Content
Split in Chunks
text chunk 1
text chunk 2
text chunk 3
.....
text chunk N
Embedding 1
Embedding 2
Embedding 3
.....
Embedding N
Build Semantic Index
Knowledge Base
Question?
Query Embedding
Semantic Search
Ranked Results
LLM Generative AI
Answer

Google Developer Groups

FAQ Bot

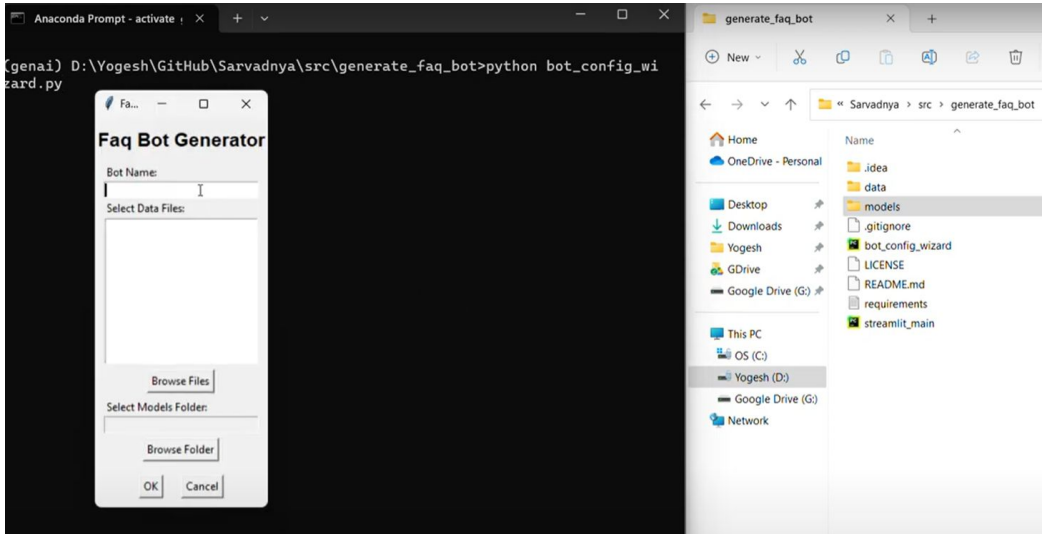
Components

- Streamlit for UI
- RAG using Langchain
- LLM by Vertex Palm APIs or Hugging Face models

Customization :: Config

- bot_config_wizard.py
- Configuration wizard UI using the tkinter library
- Output bot_config.json
- Can manually create also

```
{
  "APP_NAME": "MyApp",
  "DOCS_INDEX":"/fullpath/to/docs.index",
  "FAISS_STORE_PKL":"/fullpath/to/faiss_store.pkl",
  "FILES_PATHS": [
    "/fullpath/to/file1.csv",
    "/fullpath/to/file2.txt",
    "/fullpath/to/file3.pdf"
  ]
}
```



RAG:: Langchain

- Generates a retrieval and language model chain.
- If the bot’s docs index exists, it’s loaded; otherwise, data is loaded from various file types (CSV, PDF, TXT, HTML)
- Embeddings are generated, and the FAISS index is created.
- The chain is then constructed using the LLM and retriever.

My FAQs Bot

```
class MyFAQsBot:
    Yogesh Kulkarni
    def __init__(self, config_json):
        # print("in __init__")
        self.app_name = config_json['APP_NAME']
        self.files_paths = config_json['FILES_PATHS']
        self.docs_index = config_json['DOCS_INDEX']
        self.faiss_store_pkl = config_json['FAISS_STORE_PKL']
        self.model_name = config_json['MODEL_NAME']
```

```
def get_model(self):
    llm = None
    if self.model_name == "VertexAI":
        llm = VertexAI()  # need GCP account, project, own config set under ENV variable, refer README
    elif self.model_name == "Llama2":
        llm = LlamaCpp(model_path=self.model_path)
    return llm
```

Data Loaders and Embeddings

```
data = []
for p in self.files_paths:
    if p.lower().endswith('.csv'):
        loader = CSVLoader(file_path=p)
        data += loader.load()
    elif p.lower().endswith('.pdf'):
        loader = PyPDFLoader(file_path=p)
        data += loader.load()
    elif p.lower().endswith('.txt'):
        loader = TextLoader(file_path=p)
        data += loader.load()
    elif p.lower().endswith('.html'):
        loader = UnstructuredHTMLLoader(file_path=p)
        data += loader.load()
    else:
        st.write("Selected file extension not supported")
print(f"data has {len(data)} documents")
```

```
embeddings = HuggingFaceHubEmbeddings()
store = FAISS.from_documents(data, embeddings)
faiss.write_index(store.index, self.docs_index)
with open(self.faiss_store_pkl, "wb") as f:
    pickle.dump(store, f)
```

The Chain

```
db_as_retriver = store.as_retriever()
llm = self.get_model()
chain = RetrievalQA.from_chain_type(llm=llm, retriever=db_as_retriver, verbose=False, chain_type="stuff")
return chain
```

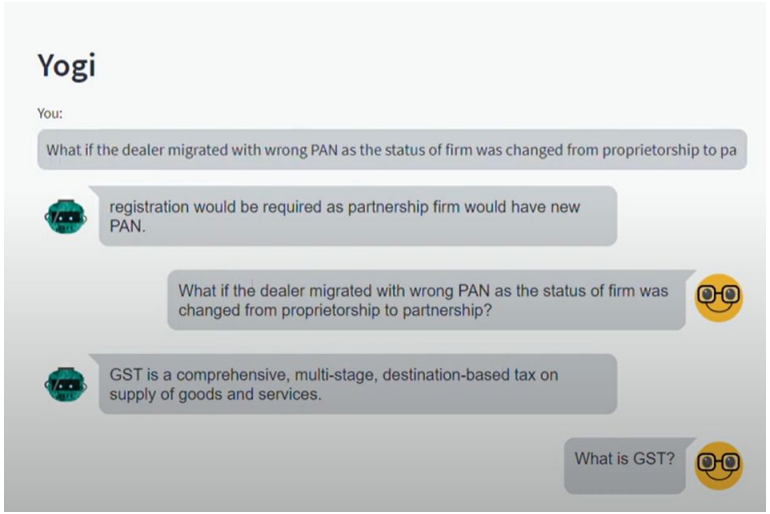
ChatBot Session

```python
if __name__ == "__main__":
    config = read_config()
    bot = MyFAQsBot(config)
    bot.run_ui()
```

```python
def run_ui(self):
    if "chain" not in st.session_state:
        st.session_state["chain"] = self.generate_chain()

    chain = st.session_state["chain"]

    st.set_page_config(page_title=self.app_name, page_icon=":robot:")
    st.header(self.app_name)

    if "generated" not in st.session_state:
        st.session_state["generated"] = []

    if "past" not in st.session_state:
        st.session_state["past"] = []

    user_input = st.text_input("You: ", "<type here>", key="input")
    prev_input = "<type here>"

    if prev_input != user_input:
        prev_input = user_input

        result = chain(user_input)  # ({"question": user_input})

        st.session_state.past.append(user_input)
        st.session_state.generated.append(result['result'])

    if st.session_state["generated"]:
        for i in range(len(st.session_state["generated"]) - 1, -1, -1):
            message(st.session_state["generated"][i], key=str(i))
            message(st.session_state["past"][i], is_user=True, key=str(i) + "_user")
```

Bot UI :: Streamlit

- Handles user inputs,
- Processes them through the chain,
- Displays messages using the streamlit_chat module.



```
Text(
  'Section Title',
  style: TextStyle(
    color: Colors.red[200],
  ),
),
```

devfest

```
s.star,
r: Colors.red[500],

Text('23'),
```

Google Developer Groups

Demo

References

- Github:yogeshhk/Sarvadnya, src/generate_faq_bot
- Blog: “Bot making Bot: Data to Dialog” (Medium)

Photo by Vidar Nordli-Mathisen on Unsplash

```
Text(
  'Section Title',
  style: TextStyle(
    color: Colors.blue[200],
  ),
),
```

devfest

```
s.star,
r: Colors.blue[500],

Text('23'),
```

Google Developer Groups

Generative AI GitHub Repository

Sample code and notebooks for GenAI on Google Cloud

goo.gle/gen-ai-github

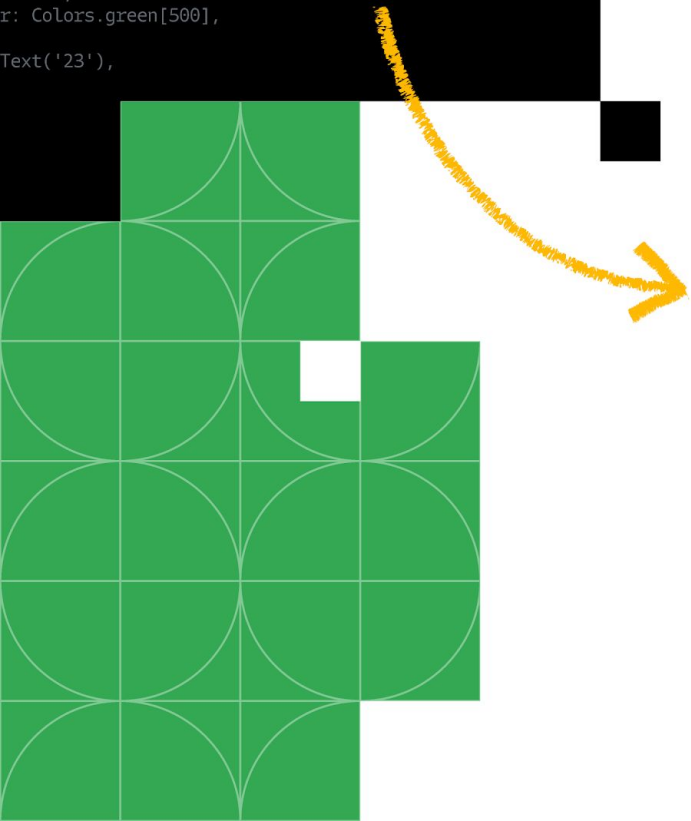
Table of Contents

Google Cloud

```
Text(
  'Section Title',
  style: TextStyle(
    color: Colors.green[200],
  ),
),
```

devfest

```
s.star,
r: Colors.green[500],

Text('23'),
```



Google Developer Groups

Learn more about Generative AI at
goo.gle/generativeai

```
Text(
  'Section Title',
  style: TextStyle(
    color: Colors.red[200],
  ),
),
```

devfest

```
s.star,
r: Colors.red[500],

Text('23'),
```

Google Developer Groups

THANK YOU

yogeshkulkarni@yahoo.com