



Current Trends in Software Engineering - SE4010
Sri Lanka Institute of Information Technology
Year 4 Semester 2

Madusanka G.K.I
IT21189944

Table of Content

Introduction	3
Justification of LLM Choice	4
Justification of Development Approach.....	5
Challenges and lessons Learned	7
Use of GenAI Tools	8

Table of Figures

Figure 1: LLM Architecture Diagram.....	5
Figure 2: Jupyter Notebook	6
Figure 3: Sample chat with created ChatBot	6
Figure 4: Create a sample UI for chat bot Question	8
Figure 5: Create a sample UI for chat bot Answer	8
Figure 6: Resolve error chroma import Question.....	9
Figure 7: Resolve error chroma import Answer	9

Introduction

This report outlines the development of an intelligent academic chatbot designed to answer questions based on lecture notes from the **Current Trends in Software Engineering (CTSE)** module. The system leverages a state-of-the-art **Large Language Model (LLM)** from **Google Gemini**, combined with the **LangChain** framework to implement a **Retrieval-Augmented Generation (RAG)** pipeline. The entire solution was developed and deployed inside a **Jupyter Notebook environment**, making it interactive, modular, and suitable for demonstration and academic experimentation.

At its core, the chatbot is capable of accepting natural language questions from users and generating accurate, contextually relevant answers derived from PDF-based CTSE lecture materials. The PDF content is first split into manageable textual chunks, converted into vector embeddings, and stored in a **Chroma vector database**. When a user poses a query, the system retrieves the most semantically relevant chunks and uses the LLM to generate a coherent and informative response.

This approach demonstrates how modern LLMs can be fine-tuned to support academic learning by turning static documents into interactive knowledge tools. It also showcases the practical application of cutting-edge AI technologies in solving real-world problems in the field of education and knowledge retrieval.

Justification of LLM Choice

For this Assignment, Google Gemini (Model: Gemini 2.0 flash) was selected as the LLM over alternatives like OpenAI's GPT – 3.5/4 and Anthropic's Claude.

Gemini Offers the following advantages:

- Native Integration with Google's Ecosystem and LangChain modules
- Fast response time suitable for real – time Q&A tasks
- Accurate embeddings model (Embedding - 001) for vector similarity search
- Cost efficiency and high-quality answers for academic applications

While GPT models also provide high performance, Gemini's seamless API experience, reduce latency, and cost-effectiveness made it a better fit for the educational scope of this project.

Justification of Development Approach

Workflow Overview

The system follows a classic RAG architecture

1. Merged PDF Lecture notes are loaded and parsed
2. Text is split into overlapping chunks
3. Chunks are converted into embeddings using Gemini Embedding-001
4. These embeddings are stored in a local vector database (Chroma)
5. When a user submits a query, the most relevant chunks are retrieved
6. The LLM uses those chunks to generate an accurate answer

Tools Used

- **LangChain:** Framework for chaining LLM and retrieval Logic
- **Google Gemini API:** Used for both embedding generation and text generation
- **Chroma vector Store:** Local database used for fast vector similarity search
- **Colab + Jupyter Notebook:** development environment

Architecture Diagram

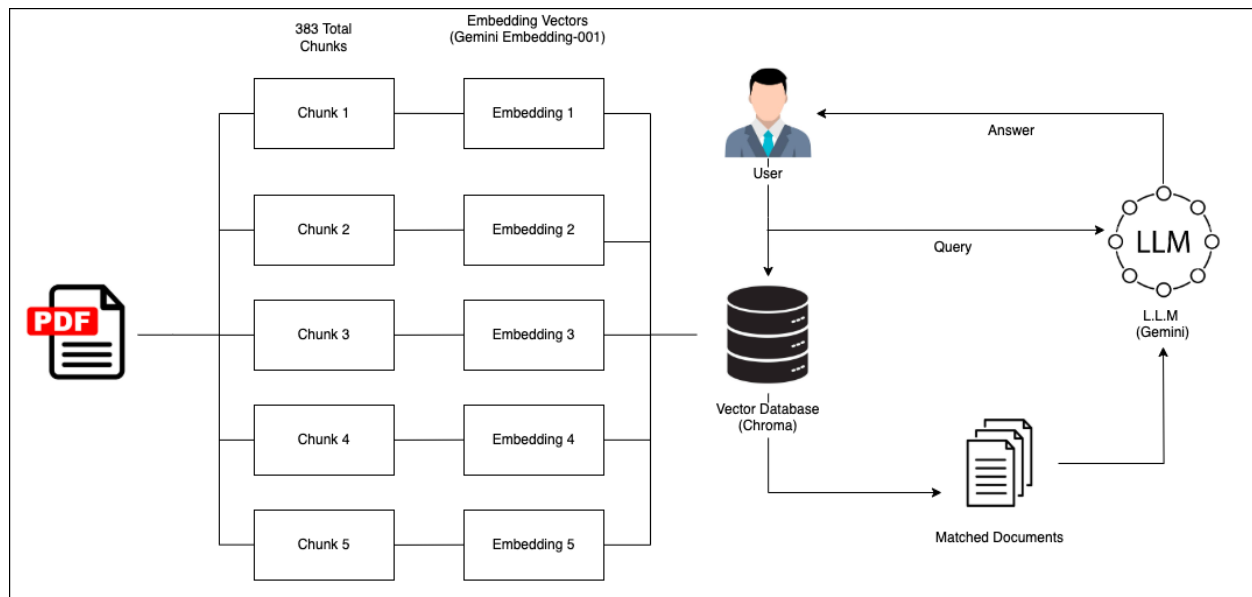
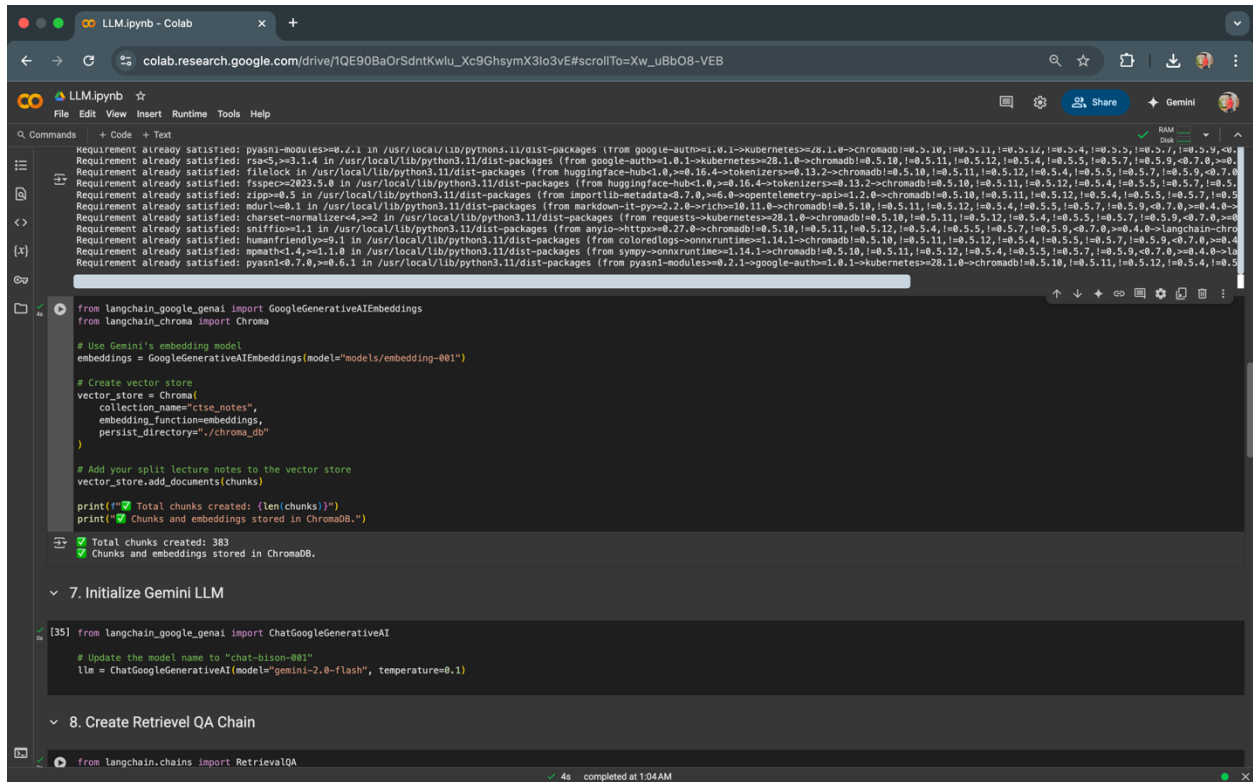


Figure 1: LLM Architecture Diagram



```

Requirement already satisfied: pyasn1-modules==0.2.1 in /usr/local/lib/python3.11/dist-packages (from google-auth==1.0.1->kubernetes==28.1.0->chromadb==0.5.10,!=0.5.11,!=0.5.12,!=0.5.4,!=0.5.5,!=0.5.7,!=0.5.9,<0.6.0)
Requirement already satisfied: rsa<5,=>3.14 in /usr/local/lib/python3.11/dist-packages (from google-auth==1.0.1->kubernetes==28.1.0->chromadb==0.5.10,!=0.5.11,!=0.5.12,!=0.5.4,!=0.5.5,!=0.5.7,!=0.5.9,<0.6.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,=>0.16.4->tokenizers==0.13.2->chromadb==0.5.10,!=0.5.11,!=0.5.12,!=0.5.4,!=0.5.5,!=0.5.7,!=0.5.9,<0.6.0)
Requirement already satisfied: fsspec>2023.5.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,=>0.16.4->tokenizers==0.13.2->chromadb==0.5.10,!=0.5.11,!=0.5.12,!=0.5.4,!=0.5.5,!=0.5.7,!=0.5.9,<0.6.0)
Requirement already satisfied: zipp>0.5 in /usr/local/lib/python3.11/dist-packages (from importlib-metadata==0.7.0,=>6.0->opentelemetry-api==1.2.0->chromadb==0.5.10,!=0.5.11,!=0.5.12,!=0.5.4,!=0.5.5,!=0.5.7,!=0.5.9,<0.6.0)
Requirement already satisfied: mdurl==0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py==2.2.0->rich==10.11.0->chromadb==0.5.10,!=0.5.11,!=0.5.12,!=0.5.4,!=0.5.5,!=0.5.7,!=0.5.9,<0.6.0)
Requirement already satisfied: charset-normalizer<4,=>2 in /usr/local/lib/python3.11/dist-packages (from requests->kubernetes==28.1.0->chromadb==0.5.10,!=0.5.11,!=0.5.12,!=0.5.4,!=0.5.5,!=0.5.7,!=0.5.9,<0.6.0)
Requirement already satisfied: sniffio==1.1 in /usr/local/lib/python3.11/dist-packages (from anyio->httpx==0.27.0->chromadb==0.5.10,!=0.5.11,!=0.5.12,!=0.5.4,!=0.5.5,!=0.5.7,!=0.5.9,<0.6.0)
Requirement already satisfied: humanfriendly==9.1 in /usr/local/lib/python3.11/dist-packages (from coloredlogs->opentelemetry==1.14.1->chromadb==0.5.10,!=0.5.11,!=0.5.12,!=0.5.4,!=0.5.5,!=0.5.7,!=0.5.9,<0.6.0)
Requirement already satisfied: pypath<1.4,=>1.0 in /usr/local/lib/python3.11/dist-packages (from sympy->onnxruntime==1.14.1->chromadb==0.5.10,!=0.5.11,!=0.5.12,!=0.5.4,!=0.5.5,!=0.5.7,!=0.5.9,<0.6.0)
Requirement already satisfied: pyasn1<0.7.0,=>0.6.1 in /usr/local/lib/python3.11/dist-packages (from pyasn1-modules==0.2.1->google-auth==1.0.1->kubernetes==28.1.0->chromadb==0.5.10,!=0.5.11,!=0.5.12,!=0.5.4,!=0.5.5,!=0.5.7,!=0.5.9,<0.6.0)

from langchain_google_genai import GoogleGenerativeAIEmbeddings
from langchain_chroma import Chroma

# Use Gemini's embedding model
embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001")

# Create vector store
vector_store = Chroma(
    collection_name="tse_notes",
    embedding_function=embeddings,
    persist_directory="./chroma_db"
)

# Add your split lecture notes to the vector store
vector_store.add_documents(chunks)

print(f"Total chunks created: {len(chunks)}")
print(f"Chunks and embeddings stored in ChromaDB.")

✓ Total chunks created: 383
✓ Chunks and embeddings stored in ChromaDB.

7. Initialize Gemini LLM

[35] from langchain_google_genai import ChatGoogleGenerativeAI

# Update the model name to "chat-bison-001"
llm = ChatGoogleGenerativeAI(model="gemini-2.0-flash", temperature=0.1)

8. Create Retrieval QA Chain

from langchain.chains import RetrievalQA
  
```

Figure 2: Jupyter Notebook

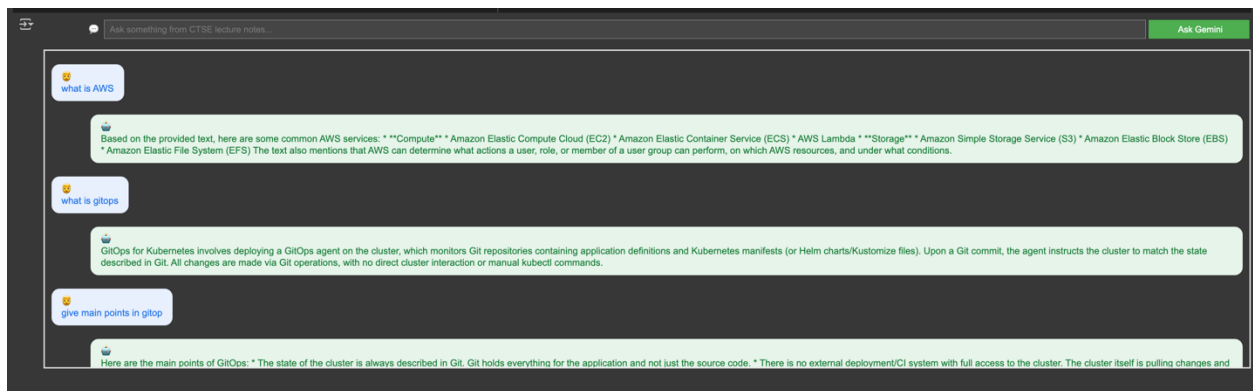


Figure 3: Sample chat with created ChatBot

Challenges and lessons Learned

Challenges

- Handling API key security in Colab
- Manage Chunk overlap to preserve context
- Identify the Architectural flow of the system
- Understand the difference between vector stores
- Create a GUI using python to chat with modal

Lessons Learned

- Importance of proper text chunking for high quality retrieval
- Transparent use of GenAI tools and clear explanations improves reliability
- How to work with RAG

Use of GenAI Tools

1. Create a sample UI for chat bot

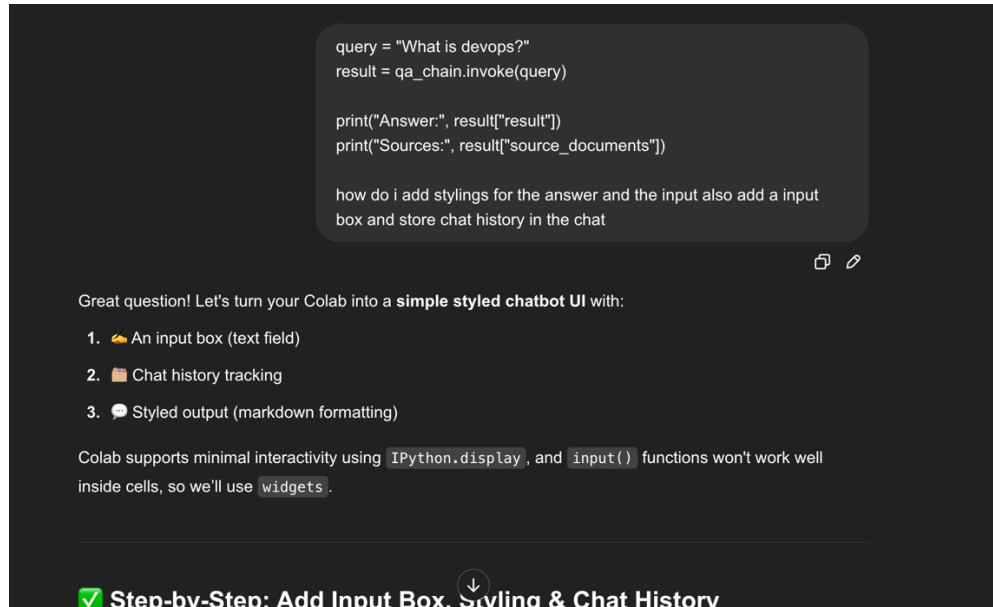


Figure 4: Create a sample UI for chat bot Question

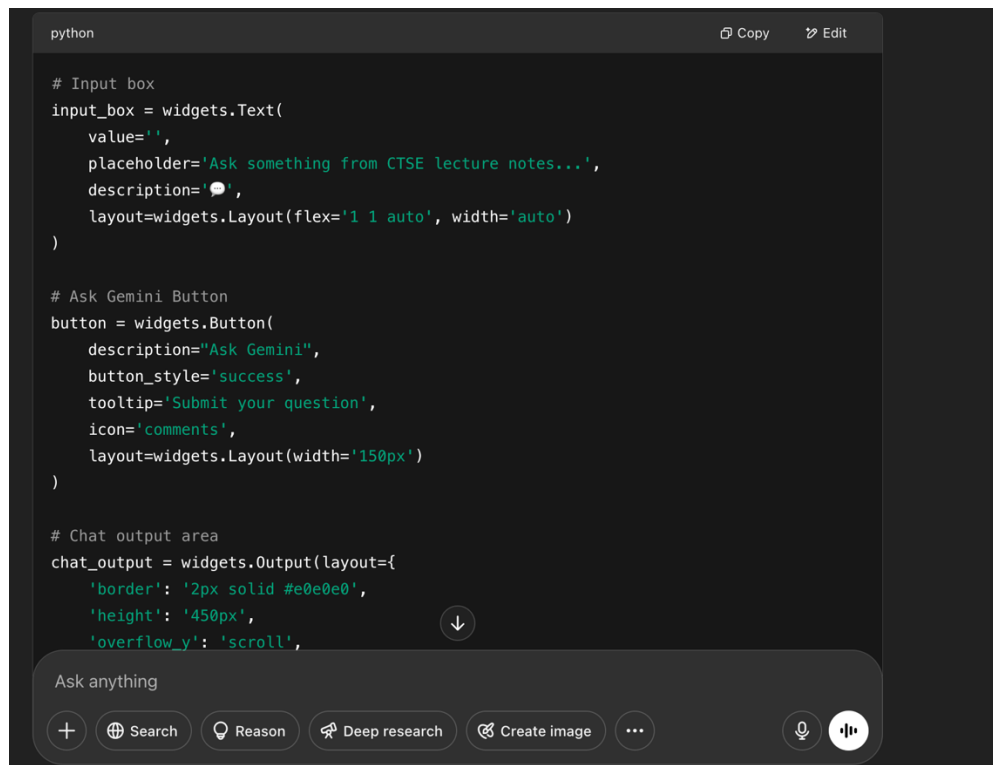
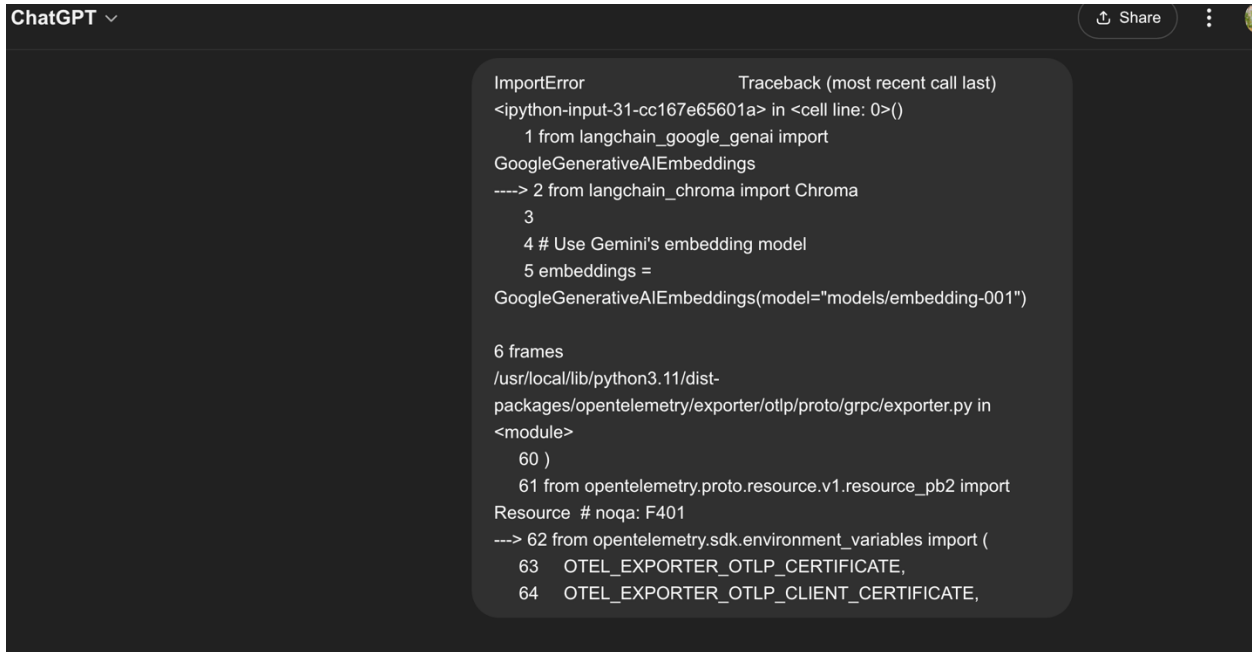


Figure 5: Create a sample UI for chat bot Answer

2. Resolve error chroma import



```

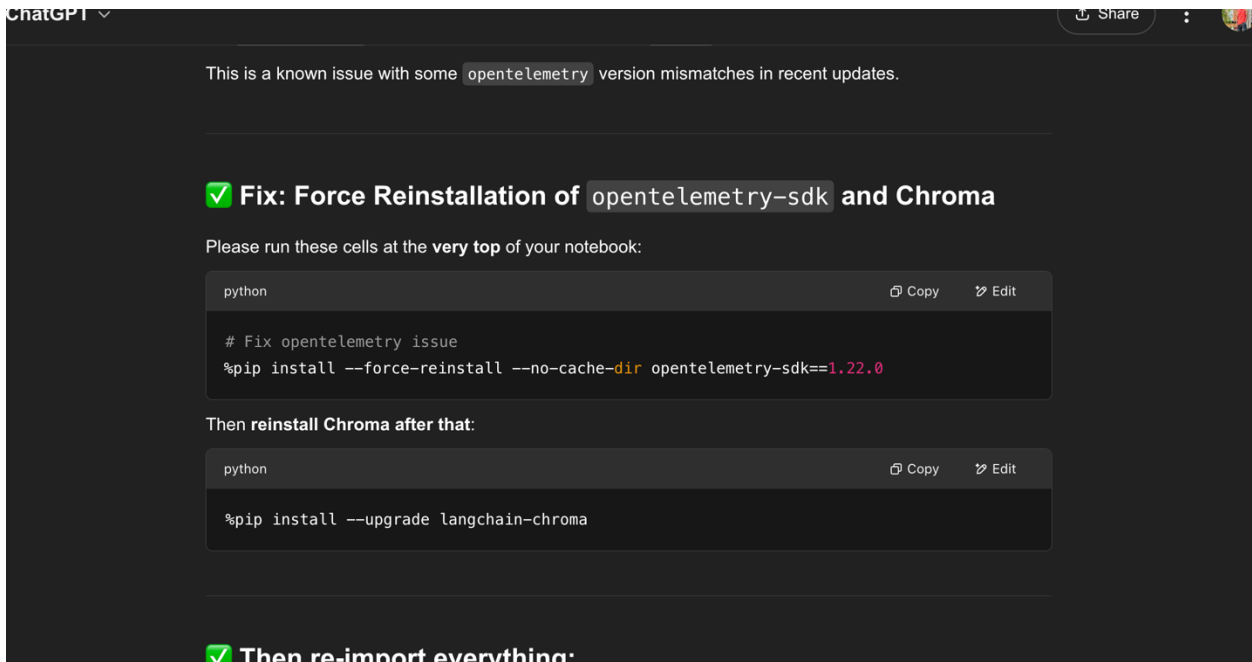
ChatGPT
Share

ImportError                                Traceback (most recent call last)
<jupyter-input-31-cc167e65601a> in <cell line: 0>()
      1 from langchain_google_genai import
GoogleGenerativeAIEmbeddings
----> 2 from langchain_chroma import Chroma
      3
      4 # Use Gemini's embedding model
      5 embeddings =
GoogleGenerativeAIEmbeddings(model="models/embedding-001")

6 frames
/usr/local/lib/python3.11/dist-
packages/opentelemetry/exporter/otlp/proto/grpc/exporter.py in
<module>
    60 )
    61 from opentelemetry.proto.resource.v1.resource_pb2 import
Resource # noqa: F401
--> 62 from opentelemetry.sdk.environment_variables import (
    63     OTEL_EXPORTER_OTLP_CERTIFICATE,
    64     OTEL_EXPORTER_OTLP_CLIENT_CERTIFICATE,

```

Figure 6: Resolve error chroma import Question



This is a known issue with some `opentelemetry` version mismatches in recent updates.

✅ Fix: Force Reinstallation of `opentelemetry-sdk` and Chroma

Please run these cells at the **very top** of your notebook:

```
python
# Fix opentelemetry issue
%pip install --force-reinstall --no-cache-dir opentelemetry-sdk==1.22.0
```

Then **reinstall Chroma** after that:

```
python
%pip install --upgrade langchain-chroma
```

✅ Then re-import everything:

Figure 7: Resolve error chroma import Answer