Streaming AI-supported healthcare assistance platform with LangGraph and Retrieval-Augmented Generation

```
"""
# Title
Agentic Healthcare customer support system using LangGraph with streaming and

# Project overview:
Current healthcare support systems are used to solve a range of customers’ qu
First-generation chatbots which are used rule based tasks, can handle with se
To overcome these limitations, agentic AI system is considered. It

•   Identifies user goals and emotions.
•   Incorporates RAG for accurate knowledge responses.
•   Dispatches user questions automatically with workflow decision graph.
•   Routes to human agents.
•   Streams emergency medical operations
•   Supports live and non-live LLM processing

# Mission of the Project:
The objective of this project (end-to-end Agentic AI pipelines with LLM) incl
•   Tags healthcare customer questions.
•   Interprets user feelings.
•   Allocates queries based on context using a graph associated agent.
•   Runs agents in live streaming approach for questions responses.
•   Creates evidence-based results using RAG.
•   Sends user questions to either human agent (or medical team).

# Core tools:
•   google.colab (Python) |
•   Jupyter Widgets | LangChain | LangGraph | OpenAI (gpt-4o, LLM) GPT-4o.
•   Vector Database (Chroma) | Prompt Engineering | TypedDict.
•   Retrieval-Augmented Generation (RAG).
•   Real time and planned LLM Tasks.

# Notable Features:
•   Multi-Agent graph framework
•   Automated query sorting
•   Emotion-sensitive routing
•   RAG
•   Human involved in the loop pipeline
•   Urgent medical response workflow
•   Online and offline performance
•   Fully configured operational framework
```

```
# Project file:
Healthcare_Support_Agents_LangGraph_RAG_LLM_Pipelines.ipynb

"""
```

Load LLM workflow components for Agentic AI systems (Agentic router with RAG and Sentiment processing)

```
!pip install langchain==0.3.21

Requirement already satisfied: langchain==0.3.21 in /usr/local/lib/python3.12/
Requirement already satisfied: langchain-core<1.0.0,>=0.3.45 in /usr/local/lib
Requirement already satisfied: langchain-text-splitters<1.0.0,>=0.3.7 in /usr/
Requirement already satisfied: langsmith<0.4,>=0.1.17 in /usr/local/lib/python
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/python
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.12
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.12/dis
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in /usr/local/l
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in /usr/local/lib/pyth
Requirement already satisfied: typing-extensions<5.0.0,>=4.7.0 in /usr/local/l
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in /usr/local/lib/pyt
Requirement already satisfied: uuid-utils<1.0,>=0.12.0 in /usr/local/lib/pytho
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/python3
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/local/l
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/pyt
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/pyth
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/pyth
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.12/d

!pip install langchain-community==0.3.20

Collecting langchain-community==0.3.20
```

```
  Downloading langchain_community-0.3.20-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: langchain-core<1.0.0,>=0.3.45 in /usr/local/li
Requirement already satisfied: langchain<1.0.0,>=0.3.21 in /usr/local/lib/pyt
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.1
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.12/di
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in /usr/local/lib/
Collecting dataclasses-json<0.7,>=0.5.7 (from langchain-community==0.3.20)
  Downloading dataclasses_json-0.6.7-py3-none-any.whl.metadata (25 kB)
Requirement already satisfied: pydantic-settings<3.0.0,>=2.4.0 in /usr/local/
Requirement already satisfied: langsmith<0.4,>=0.1.125 in /usr/local/lib/pyth
Requirement already satisfied: httpx-sse<1.0.0,>=0.4.0 in /usr/local/lib/pyth
Requirement already satisfied: numpy<3,>=1.26.2 in /usr/local/lib/python3.12/
Requirement already satisfied: aiohappyeyeballs>=2.5.0 in /usr/local/lib/pyth
Requirement already satisfied: aiosignal>=1.4.0 in /usr/local/lib/python3.12/
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.12/dis
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.12
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.12/
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.12
Collecting marshmallow<4.0.0,>=3.18.0 (from dataclasses-json<0.7,>=0.5.7->lan
  Downloading marshmallow-3.26.2-py3-none-any.whl.metadata (7.3 kB)
Collecting typing-inspect<1,>=0.4.0 (from dataclasses-json<0.7,>=0.5.7->langc
  Downloading typing_inspect-0.9.0-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: langchain-text-splitters<1.0.0,>=0.3.7 in /usr
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/pytho
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in /usr/local/lib/pyt
Requirement already satisfied: typing-extensions<5.0.0,>=4.7.0 in /usr/local/
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in /usr/local/lib/py
Requirement already satisfied: uuid-utils<1.0,>=0.12.0 in /usr/local/lib/pyth
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.12/
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/python
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/local/
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/py
Requirement already satisfied: python-dotenv>=0.21.0 in /usr/local/lib/python
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/pyt
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/pyt
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dis
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.12/
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/pytho
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python
```

```
Collecting mypy-extensions>=0.3.0 (from typing-inspect<1,>=0.4.0->dataclasses
  Downloading mypy_extensions-1.1.0-py3-none-any.whl.metadata (1.1 kB)
Downloading langchain_community-0.3.20-py3-none-any.whl (2.5 MB)
[2K   [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━[0m [32m2.5/2.5 MB[0
[?25hDownloading dataclasses_json-0.6.7-py3-none-any.whl (28 kB)
Downloading marshmallow-3.26.2-py3-none-any.whl (50 kB)
[2K   [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━[0m [32m51.0/51.0 kB
[?25hDownloading typing_inspect-0.9.0-py3-none-any.whl (8.8 kB)
Downloading mypy_extensions-1.1.0-py3-none-any.whl (5.0 kB)
Installing collected packages: mypy-extensions, marshmallow, typing-inspect,
Successfully installed dataclasses-json-0.6.7 langchain-community-0.3.20 mars

!pip install langchain-openai==0.3.9

Collecting langchain-openai==0.3.9
  Downloading langchain_openai-0.3.9-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: langchain-core<1.0.0,>=0.3.45 in /usr/local/li
Collecting openai<2.0.0,>=1.66.3 (from langchain-openai==0.3.9)
  Downloading openai-1.109.1-py3-none-any.whl.metadata (29 kB)
Requirement already satisfied: tiktoken<1,>=0.7 in /usr/local/lib/python3.12/
Requirement already satisfied: langsmith<1.0.0,>=0.3.45 in /usr/local/lib/pyt
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in /usr/local/
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in /usr/local/lib/pyt
Requirement already satisfied: PyYAML<7.0.0,>=5.3.0 in /usr/local/lib/python3
Requirement already satisfied: typing-extensions<5.0.0,>=4.7.0 in /usr/local/
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in /usr/local/lib/py
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/pytho
Requirement already satisfied: uuid-utils<1.0,>=0.12.0 in /usr/local/lib/pyth
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.12/
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.12/
Requirement already satisfied: jiter<1,>=0.4.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: sniffio in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.12/
Requirement already satisfied: requests>=2.26.0 in /usr/local/lib/python3.12/
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.12/
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/python3
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/local/
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/pyt
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/pytho
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/pyt
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/pyt
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12
```

```
Downloading langchain_openai-0.3.9-py3-none-any.whl (60 kB)
[2K   [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━[0m [32m60.9/60.9 kB[
[?25hDownloading openai-1.109.1-py3-none-any.whl (948 kB)
[2K   [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━[0m [32m948.6/948.6 kB
[?25hInstalling collected packages: openai, langchain-openai
  Attempting uninstall: openai
    Found existing installation: openai 2.14.0
    Uninstalling openai-2.14.0:
      Successfully uninstalled openai-2.14.0
Successfully installed langchain-openai-0.3.9 openai-1.109.1

!pip install langgraph==0.3.18

Collecting langgraph==0.3.18
  Downloading langgraph-0.3.18-py3-none-any.whl.metadata (7.5 kB)
Requirement already satisfied: langchain-core<0.4,>=0.1 in /usr/local/lib/pyth
Collecting langgraph-checkpoint<3.0.0,>=2.0.10 (from langgraph==0.3.18)
  Downloading langgraph_checkpoint-2.1.2-py3-none-any.whl.metadata (4.2 kB)
Collecting langgraph-prebuilt<0.2,>=0.1.1 (from langgraph==0.3.18)
  Downloading langgraph_prebuilt-0.1.8-py3-none-any.whl.metadata (5.0 kB)
Collecting langgraph-sdk<0.2.0,>=0.1.42 (from langgraph==0.3.18)
  Downloading langgraph_sdk-0.1.74-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: langsmith<1.0.0,>=0.3.45 in /usr/local/lib/pyth
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in /usr/local/
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in /usr/local/lib/pyth
Requirement already satisfied: PyYAML<7.0.0,>=5.3.0 in /usr/local/lib/python3
Requirement already satisfied: typing-extensions<5.0.0,>=4.7.0 in /usr/local/
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in /usr/local/lib/py
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/python
Requirement already satisfied: uuid-utils<1.0,>=0.12.0 in /usr/local/lib/pytho
Requirement already satisfied: ormsgpack>=1.10.0 in /usr/local/lib/python3.12/
Requirement already satisfied: httpx>=0.25.2 in /usr/local/lib/python3.12/dist
Requirement already satisfied: orjson>=3.10.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.12/d
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.12/dis
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/local/
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/py
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/pyth
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/pyth
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12
Downloading langgraph-0.3.18-py3-none-any.whl (136 kB)
[2K   [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━[0m [32m136.5/136.5 kB
```

```
[?25hDownloading langgraph_checkpoint-2.1.2-py3-none-any.whl (45 kB)
[2K   [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━[0m [32m45.8/45.8 kB[
[?25hDownloading langgraph_prebuilt-0.1.8-py3-none-any.whl (25 kB)
Downloading langgraph_sdk-0.1.74-py3-none-any.whl (50 kB)
[2K   [90m━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━[0m [32m50.3/50.3 kB[
[?25hInstalling collected packages: langgraph-sdk, langgraph-checkpoint, lan
  Attempting uninstall: langgraph-sdk
    Found existing installation: langgraph-sdk 0.3.1
    Uninstalling langgraph-sdk-0.3.1:
      Successfully uninstalled langgraph-sdk-0.3.1
  Attempting uninstall: langgraph-checkpoint
    Found existing installation: langgraph-checkpoint 3.0.1
    Uninstalling langgraph-checkpoint-3.0.1:
      Successfully uninstalled langgraph-checkpoint-3.0.1
  Attempting uninstall: langgraph-prebuilt
    Found existing installation: langgraph-prebuilt 1.0.5
    Uninstalling langgraph-prebuilt-1.0.5:
      Successfully uninstalled langgraph-prebuilt-1.0.5
  Attempting uninstall: langgraph
    Found existing installation: langgraph 1.0.5
    Uninstalling langgraph-1.0.5:
      Successfully uninstalled langgraph-1.0.5
Successfully installed langgraph-0.3.18 langgraph-checkpoint-2.1.2 langgraph-p

import langchain

import langchain_community

import langchain_openai

import langgraph

!pip install langchain-chroma==0.2.2

Requirement already satisfied: langchain-chroma==0.2.2 in /usr/local/lib/pytho
Requirement already satisfied: langchain-core!=0.3.0,!=0.3.1,!=0.3.10,!=0.3.1
Requirement already satisfied: numpy<2.0.0,>=1.26.2 in /usr/local/lib/python3
Requirement already satisfied: chromadb!=0.5.10,!=0.5.11,!=0.5.12,!=0.5.4,!=0
Requirement already satisfied: build>=1.0.3 in /usr/local/lib/python3.12/dist
Requirement already satisfied: pydantic>=1.9 in /usr/local/lib/python3.12/dist
Requirement already satisfied: chroma-hnswlib==0.7.6 in /usr/local/lib/python3
Requirement already satisfied: fastapi>=0.95.2 in /usr/local/lib/python3.12/di
Requirement already satisfied: uvicorn>=0.18.3 in /usr/local/lib/python3.12/di
Requirement already satisfied: posthog>=2.4.0 in /usr/local/lib/python3.12/dis
Requirement already satisfied: typing_extensions>=4.5.0 in /usr/local/lib/pyth
Requirement already satisfied: onnxruntime>=1.14.1 in /usr/local/lib/python3.1
Requirement already satisfied: opentelemetry-api>=1.2.0 in /usr/local/lib/pyth
Requirement already satisfied: opentelemetry-exporter-otlp-proto-grpc>=1.2.0 i
Requirement already satisfied: opentelemetry-instrumentation-fastapi>=0.41b0 i
```

```
Requirement already satisfied: opentelemetry-sdk>=1.2.0 in /usr/local/lib/pyth
Requirement already satisfied: tokenizers>=0.13.2 in /usr/local/lib/python3.12
Requirement already satisfied: pypika>=0.48.9 in /usr/local/lib/python3.12/dis
Requirement already satisfied: tqdm>=4.65.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: overrides>=7.3.1 in /usr/local/lib/python3.12/d
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.1
Requirement already satisfied: grpcio>=1.58.0 in /usr/local/lib/python3.12/dis
Requirement already satisfied: bcrypt>=4.0.1 in /usr/local/lib/python3.12/dist
Requirement already satisfied: typer>=0.9.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: kubernetes>=28.1.0 in /usr/local/lib/python3.12
Requirement already satisfied: tenacity>=8.2.3 in /usr/local/lib/python3.12/di
Requirement already satisfied: PyYAML>=6.0.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: mmh3>=4.0.1 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: orjson>=3.9.12 in /usr/local/lib/python3.12/dis
Requirement already satisfied: httpx>=0.27.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: langsmith<1.0.0,>=0.3.45 in /usr/local/lib/pyth
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in /usr/local/lib/pyth
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in /usr/local/lib/pyt
Requirement already satisfied: uuid-utils<1.0,>=0.12.0 in /usr/local/lib/pytho
Requirement already satisfied: pyproject_hooks in /usr/local/lib/python3.12/di
Requirement already satisfied: starlette<0.51.0,>=0.40.0 in /usr/local/lib/pyt
Requirement already satisfied: annotated-doc>=0.0.2 in /usr/local/lib/python3
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.12/d
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python
Requirement already satisfied: websocket-client!=0.40.0,!=0.41.*,!=0.42.*,>=0
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: requests-oauthlib in /usr/local/lib/python3.12/
Requirement already satisfied: urllib3!=2.6.0,>=1.24.2 in /usr/local/lib/pytho
Requirement already satisfied: durationpy>=0.7 in /usr/local/lib/python3.12/di
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/local/l
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/pyt
Requirement already satisfied: coloredlogs in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: flatbuffers in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: protobuf in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: sympy in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: importlib-metadata<8.8.0,>=6.0 in /usr/local/li
Requirement already satisfied: googleapis-common-protos~=1.57 in /usr/local/li
Requirement already satisfied: opentelemetry-exporter-otlp-proto-common==1.39
Requirement already satisfied: opentelemetry-proto==1.39.1 in /usr/local/lib/p
Requirement already satisfied: opentelemetry-instrumentation-asgi==0.60b1 in /
Requirement already satisfied: opentelemetry-instrumentation==0.60b1 in /usr/l
```

```
Requirement already satisfied: opentelemetry-semantic-conventions==0.60b1 in /
Requirement already satisfied: opentelemetry-util-http==0.60b1 in /usr/local/l
Requirement already satisfied: wrapt<2.0.0,>=1.0.0 in /usr/local/lib/python3.1
Requirement already satisfied: asgiref~=3.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: backoff>=1.10.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: distro>=1.5.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/pyth
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/pytho
Requirement already satisfied: huggingface-hub<2.0,>=0.16.4 in /usr/local/lib/
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.12
Requirement already satisfied: httptools>=0.6.3 in /usr/local/lib/python3.12/d
Requirement already satisfied: python-dotenv>=0.13 in /usr/local/lib/python3.1
Requirement already satisfied: uvloop>=0.15.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: watchfiles>=0.13 in /usr/local/lib/python3.12/d
Requirement already satisfied: websockets>=10.4 in /usr/local/lib/python3.12/d
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/pyth
Requirement already satisfied: humanfriendly>=9.1 in /usr/local/lib/python3.12
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12
```

```
import os

from getpass import getpass

os.environ['OPENAI_API_KEY'] = getpass('put OpenAI API key: ')
```

```
put OpenAI API key: ··········
```

Load information from outside files from where the AI can answer user questions

```
!pip install --quiet gdown

# Load healthcare knowledge base (KB) documents
```

```
!gdown 1_bQj7VkXDMwwqJmspFgRzH2mgK1CVMUY

Downloading...
From: https://drive.google.com/uc?id=1_bQj7VkXDMwwqJmspFgRzH2mgK1CVMUY
To: /content/healthcare_db.json
  0% 0.00/10.6k [00:00<?, ?B/s]100% 10.6k/10.6k [00:00<00:00, 17.4MB/s]

# Load information from a JSON file. AI can answer user questions using compar

import json

with open('./healthcare_db.json', 'r') as read:
  external_data  = json.load(read)
external_data[:3]

[{'text': "Question: How can I view my healthcare billing statement online? An
  'metadata': {'category': 'billing'}},
 {'text': 'Question: What payment methods are accepted for medical bills? Answ
  'metadata': {'category': 'billing'}},
 {'text': 'Question: Can I get a detailed invoice for my medical services? Ans
  'metadata': {'category': 'billing'}}]

external_data[10:13]

[{'text': 'Question: How do I schedule a medical appointment online? Answer: Y
  'metadata': {'category': 'appointments'}},
 {'text': 'Question: Can I reschedule or cancel my appointment online? Answer
  'metadata': {'category': 'appointments'}},
 {'text': 'Question: What doctors are available for appointments? Answer: The
  'metadata': {'category': 'appointments'}}]

external_data[-1:]

[{'text': 'Question: Is my insurance information kept confidential? Answer: Yo
  'metadata': {'category': 'insurance'}}]

# For vector database processing, we need to change external data into LangCha

from langchain.schema import Document
from tqdm import tqdm

prepared_documents = []

for document_data in tqdm(external_data ):

  document_data_text = document_data['text']
  document_data_metadata = document_data['metadata']
  prepared_documents.append(Document( metadata = document_data_metadata, page_
```

```
prepared_documents[:2]
```

```
100%|██████████| 40/40 [00:00<00:00, 76748.47it/s]
```

```
[Document(metadata={'category': 'billing'}, page_content="Question: How can I
 Document(metadata={'category': 'billing'}, page_content='Question: What payme
```

Load healthcare Knowledge Base (KB) into vector database

AI is just looking for a well-organized dataset where I can load my vector database.

```
# To create a vector database, we need to make embeddings for text documents.
```

```
from langchain_openai import OpenAIEmbeddings
```

```
openai_embedding_model = OpenAIEmbeddings(model = 'text-embedding-3-small')
```

```
from langchain_chroma import Chroma
```

```
# Create Chroma vector database with OpenAI embeddings
company_vector_db = Chroma.from_documents(documents = prepared_documents, embe
                                          persist_directory = './knowledge_bas
```

```
ERROR:chromadb.telemetry.product.posthog:Failed to send telemetry event Client
ERROR:chromadb.telemetry.product.posthog:Failed to send telemetry event Client
```

Vector store query engine

```
company_data_lookup = company_vector_db.as_retriever(search_type = 'similarity
```

```
# Find a list of available categories from chroma vector store.
```

```
metadata_results = company_vector_db._collection.get(include= ['metadatas'])
```

```
list_categories  = set()
for data in metadata_results['metadatas']:
    if 'category' in data:
        list_categories.add(data['category'])

print(list_categories)
```

```
ERROR:chromadb.telemetry.product.posthog:Failed to send telemetry event Colle
```

```
{'billing', 'appointments', 'medical_records', 'insurance'}
```

```
# Is each category working well?

question = 'How can I update my car insurance policy? '
general_data_filter = {'category': 'insurance'}

company_data_lookup.search_kwargs['filter'] = general_data_filter
company_data_lookup.invoke(question)
```

```
ERROR:chromadb.telemetry.product.posthog:Failed to send telemetry event Colle
```

```
[Document(id='a7a6bb35-51e1-4935-ad4f-bfbfdd26afdc', metadata={'category': 'in
 Document(id='fcabed08-2d81-4628-b673-df4b3ee2c77e', metadata={'category': 'in
 Document(id='bf8ee54e-e3e2-4c1f-9f03-37fddd9a0691', metadata={'category': 'in
 Document(id='133056b9-bac1-4f77-8370-0fb20c7a618c', metadata={'category': 'in
```

```
question = 'Can I get access for my medical records?'
general_data_filter = {'category': 'medical_records'}

company_data_lookup.search_kwargs['filter'] = general_data_filter
company_data_lookup.invoke(question)
```

```
[Document(id='4be7da9c-7e28-4074-a35f-6509634e3089', metadata={'category': 'me
 Document(id='5b766faf-fce5-4a8d-a9dd-b79fb7a5cb46', metadata={'category': 'me
 Document(id='cb782ccb-089c-4bc7-abdf-975445b35ae0', metadata={'category': 'me
 Document(id='6e602c30-cc62-4e15-ba69-d1a2add543d7', metadata={'category': 'me
```

```
question = 'Can I pay the due amount by credit card?'
general_data_filter = {'category':  'billing'}

company_data_lookup.search_kwargs['filter'] = general_data_filter
company_data_lookup.invoke(question)
```

```
[Document(id='23b2ae07-e2a5-4f09-871f-5896a554fb8c', metadata={'category': 'bi
```

```
 Document(id='ba2894af-a4e3-4044-923e-a3c01e1699ab', metadata={'category': 'b
 Document(id='78113614-4cff-4d10-9ac4-dda5f758f174', metadata={'category': 'b
 Document(id='cc8d54e9-e73a-44e6-bb7f-d7d4c1fe58f7', metadata={'category': 'b

question = 'Can you help me schedule an appointment with my regular doctor?'
general_data_filter = {'category': 'appointments'}

company_data_lookup.search_kwargs['filter'] = general_data_filter
company_data_lookup.invoke(question)

[Document(id='5f0c5e5b-30d3-4fb8-be77-a4a7ec3c4dd8', metadata={'category': 'a
 Document(id='ff1cc517-b254-4d45-9371-4eea634cb58c', metadata={'category': 'a
 Document(id='bd883d93-1dab-4b81-881a-61f68ebef566', metadata={'category': 'a
 Document(id='6d3f444f-2cb4-4700-8523-a9849bc12d57', metadata={'category': 'a
```

Create the agentic RAG routing system

Customer support interaction structure

```
from typing import TypedDict

class Support_Interaction_Record(TypedDict):
  user_question: str
  question_type: str
  question_sentiment: str
  human_support_contact: dict
  emergency_contact_information: dict
  agent_reply: str
  user_id: str

from pydantic import BaseModel
from typing import Literal

class Question_Type(BaseModel):
  inquiry_type: Literal['insurance', 'medical_records', 'billing', 'appointme

class Question_Feeling(BaseModel):
  feeling: Literal['Good', 'Average', 'Bad', 'Emergency']
```

Node functions for making a customer support pipeline

```
from langchain_openai import ChatOpenAI

chatopenai_llm = ChatOpenAI(model = 'gpt-4o', temperature=0)

def classify_question(support_state: Support_Interaction_Record) -> Support_In
  '''
  Analyze the customer question. Find a category from insurance, medical_reco
  to the customer's question.
  '''

  classify_prompt = f'''
  You are an experienced healthcare support agent. You can easily classify the
  questions. Finally, you need to categorize questions into one of the follow

  Question:
  {support_state['user_question']}

  Make sure you must return one classify name only.
  '''
  inquiry_category = chatopenai_llm.with_structured_output(Question_Type).invo
  results = chatopenai_llm.invoke(classify_prompt).content.strip().lower()
  support_state['question_type'] = results

  return support_state

# Check: is this function working?
classify_question({"user_question": 'Tell your user which payment methods you

{'user_question': 'Tell your user which payment methods you can consider.',
 'question_type': 'billing'}


check_state = {'user_question': 'Tell your user which payment methods you can
               'human_support_contact': {}, 'emergency_contact_information': {

print(classify_question(check_state))

{'user_question': 'Tell your user which payment methods you can consider.', 'q

classify_question({"user_question": 'how  do I change my policy?'})

{'user_question': 'how  do I change my policy?', 'question_type': 'insurance'}

def sentiment_analysis_node(support_state: Support_Interaction_Record) -> Supp
```

```
    '''
    Research on sentimental options such as Good, Average, Bad, Emergency on
    '''

    sentiment_analysis_node_prompt = f'''
    You are an experienced healthcare customer support agent. You can easily
    review the customer questions as given below. Finally, you need to catego
    'Good', 'Average', 'Bad', 'Emergency'

    You need to consider the following rules:
    - Good means positive results.
    - Average means neutral.
    - Bad means negative results.
    - Emergency means serious health problems.


    Message:
    {support_state['user_question']}

    Make sure you return one of the sentiment labels.
    '''
    sentiment = chatopenai_llm.invoke(sentiment_analysis_node_prompt).content
    support_state['question_sentiment'] = sentiment
    return support_state

# Test the above function
sentiment_analysis_node({"user_question": 'Thank you. I appreciate your suppo

{'user_question': 'Thank you. I appreciate your support.',
 'question_sentiment': 'Good'}

sentiment_analysis_node({"user_question": 'She has a serious health issue. Sh

{'user_question': 'She has a serious health issue. She needs help.',
 'question_sentiment': 'Emergency'}

from langchain_core.prompts import ChatPromptTemplate

# Note:  To get better output, we need to improve the  department_assistant_r

def department_assistant_reply(support_state: Support_Interaction_Record) -> S
    '''
    Generate a department-specific response using RAG (Chroma + LLM).
    '''
    query = support_state['user_question']

    question_type = support_state['question_type']  #  'insurance', 'medical_
```

```
    # Example:  question_type ('insurance'),  Metadata Filter ( 'insurance'),
    question_type_map = {'insurance': ('insurance', 'Insurance'), 'medical_re
                         'billing': ('billing', 'Billing'), 'appointments': (

    # Example: filter_key, label wre  ('insurance', 'Insurance')
    filter_key, label = question_type_map[question_type]

    company_data_lookup.search_kwargs['filter'] = {'category': filter_key}

    #  Want to see matching documents.
    search_documents = company_data_lookup.invoke(query)
    source_text = '\n\n'.join(document.page_content for document in search_do


    chat_template = ChatPromptTemplate.from_template(
    '''
    You are an experienced healthcare customer support agent. Please provide a
    Use ONLY the knowledge base information provided below.

    If the information is insufficient, say:
    "Sorry. I could not provide right information. Please contact customer se

    Customer Question:
    {question}

    Knowledge Base Information:
    {context}
    ''')


    llm_pipeline =  chat_template | chatopenai_llm

    llm_output = llm_pipeline.invoke({'department': label, 'question': query,

    support_state['agent_reply'] = llm_output
    return support_state

function_test_quety = 'How can I request a copy of my medical history?'

sample_input = {'user_question': function_test_quety ,'question_type': 'medica

assistant_reply = department_assistant_reply(sample_input)

print(assistant_reply['agent_reply'])

You can request a copy of your medical history through the patient portal or
```

Records customer contact details when negative sentiment is noticed.

```
import ipywidgets as widgets
from IPython.display import display

def gather_user_contact_information(support_state: Support_Interaction_Record
  '''
  Collect customer information and connect with a support agent
  '''
  form_title = widgets.HTML('<h3>Help Desk Escalation Form</h3>')

  # Input fields

  user_id_field = widgets.Text(description = 'Issue ID:')
  user_name_field = widgets.Text(description = 'Name:')
  user_email_field = widgets.Text(description = 'Email:')
  user_phone_field = widgets.Text(description = 'Phone:')
  user_contact_time_field = widgets.Text(description = 'Best Time to Call:')

  submit_request_button = widgets.Button(description = 'Submit Request')
  result_display = widgets.Output()

  def process_user_input(button):
    support_state['customer_support_details'] = {'issue_id': user_id_field.va
                                                 'phone': user_phone_field.va
    with result_display:
      print('Request submitted')
    submit_request_button.disabled = True

  submit_request_button.on_click(process_user_input)

    #  Display Structure

  display(widgets.VBox([form_title, user_id_field, user_name_field, user_emai

  return support_state

request_state = {}
gather_user_contact_information(request_state)
```

```
VBox(children=(HTML(value='<h3>Help Desk Escalation Form</h3>'), Text(value='
```

{}

Send the issue to a human support agent and notify the user.

```
def notify_human_support(support_state: Support_Interaction_Record) -> Suppor
  '''
  Route the request to human support and inform the user.
  '''
  if 'customer_support_details' not in support_state:
    support_state['agent_reply'] = ('Apologies, your contact information was n
    'Please try again or contact customer support directly.')
    return support_state
  customer_details = support_state['customer_support_details']

  agent_reply_text = (f'Thank you very much {customer_details.get('name', 'Cus
  f'Issue ID: {customer_details.get('issue_id', 'N/A')}\n'
  f'Email: {customer_details.get('email', 'N/A')}\n'
  f'Phone: {customer_details.get('phone', 'N/A')}\n'
  f'Requested Contact Time: {customer_details.get('best_time_to_call', 'N/A')}
  'Someone from our team will reach out to you shortly.\n'
  'Thank you for your cooperation.'
  )
  support_state["agent_reply"] = agent_reply_text
  support_state["human_support_contact"] = customer_details
  return support_state

# test function

support_state = {'question_sentiment': 'Unhappy', 'customer_support_details':


support_state = notify_human_support(support_state)
print(support_state['agent_reply'])

Thank you very much Tim.

Issue ID: GB-097
Email: tim8@email.com
Phone: 447-000-8888
Requested Contact Time:  Today evening
Someone from our team will reach out to you shortly.
Thank you for your cooperation.
```

Collect patient details for emergency medical support.

```
def get_emergency_user_information(support_state: Support_Interaction_Record)
  '''
  Collects emergency details and connects users with emergency medical staff.
  '''

  form_title = widgets.HTML('<h3>Urgent Medical Contact Form</h3>')

  user_name_field = widgets.Text(description = 'Name:', placeholder = 'User fu
  user_phone_field = widgets.Text(description = 'Phone:', placeholder = 'Emerg
  user_medical_issue_field = widgets.Textarea(description = 'Medical Issue:',

  submit_request_button = widgets.Button(description = 'Send emergency form',
  result_display = widgets.Output()

  def manage_user_input(button):
    support_state['urgent_contact_information'] = {'name': user_name_field.val
    with result_display:
      result_display.clear_output()
      print('Emergency details registered. A doctor will contact you immediate
    submit_request_button.disabled = True

  submit_request_button.on_click(manage_user_input)

  #  Display Structure

  display(widgets.VBox([form_title, user_name_field, user_phone_field, user_me

  return support_state


support_state = {'user_question': 'I have pain in both legs and trouble walkin

support_state = get_emergency_user_information(support_state)

VBox(children=(HTML(value='<h3>Urgent Medical Contact Form</h3>'), Text(value=
```

Send the user’s emergency details to the urgent care medical team and confirms them a doctor will contact them right away.

```
def dispatch_to_emergency_team(support_state: Support_Interaction_Record) -> S
  '''
  Routes emergency contact information to the medical staff and Informs the us
  '''
  # Retrieve emergency contact details collected earlier
  emergency_contact_details = support_state.get('urgent_contact_information',

  patient_name = emergency_contact_details.get('name', 'Patient')
  patient_phone = emergency_contact_details.get('phone', 'patient phone numbe
  health_concern = emergency_contact_details.get('symptoms', 'health concern s

  output_message = (f'Emergency request received.\n'
        f'Thank you {patient_name}. Your emergency message has been received.\
        f'A doctor will be calling you soon at {patient_phone}.\n'
        f'Mentioned problem: {health_concern}\n'
        f'Stay in touch and ask for local emergency help if needed.')

  support_state['final_response'] = output_message
  support_state['case_status'] = 'Emergency doctors informed'

  return support_state

support_state = dispatch_to_emergency_team(support_state)
print(support_state['final_response'])

Emergency request received.
Thank you Patient. Your emergency message has been received.
A doctor will be calling you soon at patient phone number.
Mentioned problem: health concern submitted
Stay in touch and ask for local emergency help if needed.

# Check if the function works

support_state = {'urgent_contact_information': {'name': 'Ram', 'phone': '345-6
support_state = dispatch_to_emergency_team(support_state)
print(support_state['final_response'])

Emergency request received.
Thank you Ram. Your emergency message has been received.
```

```
A doctor will be calling you soon at 345-0088.
Mentioned problem: Extreme leg pain affecting walking
Stay in touch and ask for local emergency help if needed.
```

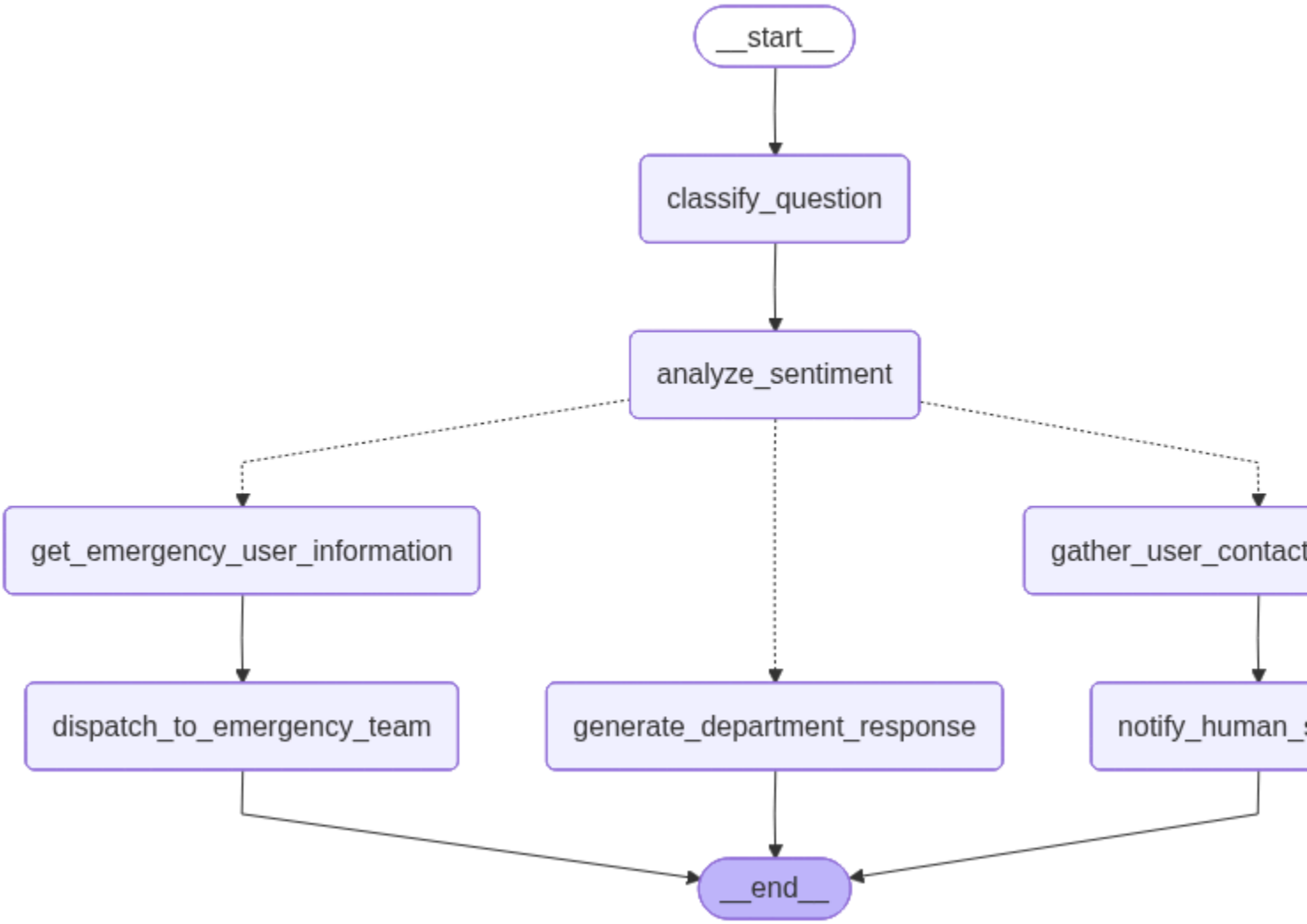
LangGraph agent workflow for query classification, sentiment analysis, and support routing

```
from langgraph.checkpoint.memory import MemorySaver

# Define the workflow state format
from typing import TypedDict

class Support_Interaction_Record(TypedDict, total=False):
    user_question: str
    question_type: str
    question_sentiment: str
    agent_reply: str
    customer_support_details: dict
    urgent_contact_information: dict
    human_support_contact: dict
    final_response: str
    case_status: str

# Assemble the graph
from langgraph.graph import StateGraph
customer_support_graph = StateGraph(Support_Interaction_Record)



customer_support_graph .add_node('classify_question', classify_question)

customer_support_graph .add_node('analyze_sentiment', sentiment_analysis_node

customer_support_graph .add_node('gather_user_contact_information', gather_us

customer_support_graph .add_node('notify_human_support', notify_human_support

customer_support_graph.add_node('get_emergency_user_information', get_emergen

customer_support_graph.add_node('dispatch_to_emergency_team', dispatch_to_eme

customer_support_graph .add_node('generate_department_response', department_a
```

```
# Sentiment evaluation router

def Sentiment_evaluation_router(support_state: Support_Interaction_Record) ->
    sentiment_question = support_state.get('question_sentiment', '').lower()

    if sentiment_question in ['bad', 'negative']:
        return 'gather_user_contact_information'

    if sentiment_question == 'emergency':
        return 'get_emergency_user_information'

    # Default (positive and neutral) processing branch
    return 'generate_department_response'


customer_support_graph.set_entry_point('classify_question')
customer_support_graph.add_edge('classify_question', 'analyze_sentiment')


customer_support_graph.add_conditional_edges('analyze_sentiment', Sentiment_ev


from langgraph.graph import END

customer_support_graph .add_edge('gather_user_contact_information', 'notify_hu
customer_support_graph.add_edge('notify_human_support', END)

customer_support_graph.add_edge('get_emergency_user_information', 'dispatch_to
customer_support_graph.add_edge('dispatch_to_emergency_team', END)

customer_support_graph.add_edge("generate_department_response", END)

# Activate the workflow

from langgraph.checkpoint.memory import MemorySaver
activate_workflow_memorysaver = MemorySaver()
workflow_agent_instance_compiled = customer_support_graph.compile(checkpointe



customer_support_pipeline_image  = workflow_agent_instance_compiled.get_graph

from IPython.display import Markdown
from IPython.display import display

display(Markdown('## Customer support flow graph'))

from IPython.display import Image
```

```
display(Image(customer_support_pipeline_image))
```

Customer support flow graph



png

Execute customer support flow

```
def run_agent_workflow(agent, user_question: str, user_session_id: str, verbos
  '''
  Use the LangGraph to solve the query and reflect the final result supportin
  '''

  # Stream execution
  graph_items = agent.stream({'user_question': user_question}, {'configurable
  last_item = None
  for  graph_item in graph_items:
```

```
        last_item =  graph_item
        if verbose:
            print(graph_item)

  question_type = last_item.get('question_type', 'N/A')
  user_question_sentiment = last_item.get('question_sentiment', 'N/A')

  response_text = (last_item.get('final_response') or last_item.get('agent_re

  # Present output clearly if needed
  display(Markdown(f'''### Workflow result summary
  ### Question category: {question_type}`
  ###  Question sentiment: {user_question_sentiment}`
  ### Support agent message

  {response_text}
  '''))
  return last_item
```

Check the customer service pipeline with sample data

```
run_agent_workflow(agent = workflow_agent_instance_compiled, user_question =
                   user_session_id = 'user_session_1gb2309')
```

Workflow result summary

Question category: medical_records### Question sentiment: Average
Support agent message

You can request a copy of your medical history through the patient portal or by contacting our records department.

```
{'user_question': 'How can I request a copy of my medical history?',
 'question_type': 'medical_records',
 'question_sentiment': 'Average',
 'agent_reply': 'You can request a copy of your medical history through the pa

query = 'Where can I see my account bill?'

run_agent_workflow(agent = workflow_agent_instance_compiled, user_question = q
```

Workflow result summary

Question category: billing### Question sentiment: Average ### Support agent message

You can view your account bill by logging into your patient portal and selecting ‘Billing Statements’.

```
{'user_question': 'Where can I see my account bill?',
 'question_type': 'billing',
 'question_sentiment': 'Average',
 'agent_reply': "You can view your account bill by logging into your patient p

query = 'Which physicians are currently available?'

run_agent_workflow(agent = workflow_agent_instance_compiled, user_question = q
```

Workflow result summary

Question category: appointments### Question sentiment: Average ### Support agent message

The following doctors are available five days of the week: Dr. Smith (Cardiology), Dr. Johnson (Dermatology), Dr. Lee (Pediatrics), Dr. Patel (Orthopedics), and Dr. Garcia (Neurology). For more details, please check the appointment section in your portal.

```
{'user_question': 'Which physicians are currently available?',
 'question_type': 'appointments',
 'question_sentiment': 'Average',
 'agent_reply': 'The following doctors are available five days of the week: D

query = 'This billing problem has been very stressful, and no one is helping t

run_agent_workflow(agent = workflow_agent_instance_compiled, user_question =

VBox(children=(HTML(value='<h3>Help Desk Escalation Form</h3>'), Text(value='
```

Workflow result summary

Question category: billing### Question sentiment: Bad ### Support agent message

Apologies, your contact information was not received. Please try again or contact customer support directly.

```
{'user_question': 'This billing problem has been very stressful, and no one is
 'question_type': 'billing',
 'question_sentiment': 'Bad',
 'agent_reply': 'Apologies, your contact information was not received.\nPleas

query = 'My legs hurt badly and I can’t easily walk.'
```

```
run_agent_workflow(agent = workflow_agent_instance_compiled, user_question =

VBox(children=(HTML(value='<h3>Urgent Medical Contact Form</h3>'), Text(value=
```

Workflow result summary

Question category: appointments### Question sentiment: Emergency
Support agent message

Emergency request received. Thank you Patient. Your emergency message has been received. A doctor will be calling you soon at patient phone number. Mentioned problem: health concern submitted Stay in touch and ask for local emergency help if needed.

```
{'user_question': 'My legs hurt badly and I can’t easily walk.',
 'question_type': 'appointments',
 'question_sentiment': 'Emergency',
 'final_response': 'Emergency request received.\nThank you Patient. Your emerg
 'case_status': 'Emergency doctors informed'}

query = 'Thank you, I am going to check my doctor appointment schedule right n

run_agent_workflow(agent = workflow_agent_instance_compiled, user_question =
```

Workflow result summary

Question category: appointments### Question sentiment: Good
Support agent message

Sorry. I could not provide right information. Please contact customer service.

```
{'user_question': 'Thank you, I am going to check my doctor appointment schedu
 'question_type': 'appointments',
 'question_sentiment': 'Good',
 'agent_reply': 'Sorry. I could not provide right information. Please contact
```