AI Chat Agent

1 Build the environment

```
from flask import Flask, request
import os

os.environ["OPENAI_API_KEY"] = " API KEY"    #use to connect the os with the openAI environment

app = Flask(__name__)

# use flask to construct backend server for the whole program
```

2 llama_index docs query builder

```
# Load and index documents
subsidy_docs = SimpleDirectoryReader(input_files=["main_subsidy_data.csv"]).load_data()
subsidy_index = VectorStoreIndex.from_documents(subsidy_docs)

# Create query engine
subsidy_engine = subsidy_index.as_query_engine(similarity_top_k=6)
```

this code use to load and build queries for the documents

SDReader : load docs
VSIndex : convert the docs into vector queries

as_query_engine(similarity_top_k=6) : construct engine that can use as browser in queries, return the 6 most similar docs

3 Tools

```
subsidy_tool = QueryEngineTool(
    query_engine=subsidy_engine,
    metadata=ToolMetadata(
        name="get_subsidy_info",
        description="Provides information about subsidies for farmers in India.",
    ),
)
```

use query engine before and package into tool function

ToolMetadata describe tool info

```
sms_tool = FunctionTool.from_defaults(
    fn=send_sms_with_subsidy_info,
    name="send_sms_with_subsidy_info",
    description="Sends an SMS with 'how_to_apply' information for relevant subsidies.",
)
```

4 Make Agent

```
# Custom prompt for the agent
CUSTOM_PROMPT = """
You are a helpful conversational assistant. Below are the global details about the usecase which you need to abide by strictly:
<global_details>
Task: You are an intelligent agricultural assistant. Your goal is to provide helpful information to farmers about subsidies, weather, and farming advice. Always ask follow-up questions to understand the farmer's specific needs better before giving detailed information.
Use the provided tools to gather information, but always ask follow-up questions before providing detailed answers. This will help you give more targeted and useful information to the farmer. For eg: "Agent: Are you looking for subsidies related to seeds, crops, machines or insurance?"
Response style: Your responses must be very short, concise and helpful.
</global_details>


"""

# Initialize the agent
llm = OpenAI(temperature=0, model="gpt-4")
memory = ChatMemoryBuffer.from_defaults(token_limit=2048)
agent_worker = FunctionCallingAgentWorker.from_tools(
  [subsidy_tool, sms_tool],
  system_prompt=CUSTOM_PROMPT,
  memory=memory,
  llm=llm,
  verbose=True,
  allow_parallel_tool_calls=False,
)
agent = agent_worker.as_agent().
```

```
llm = OpenAI(temperature=0, model="gpt-4")
#choose OpenAI gpt-4 as llm
#temperature = 0 reduce randomize answer

memory = ChatMemoryBuffer.from_defaults(token_limit=2048)
#store the conversation at token limit 2048

FunctionCallingAgentWorker.from_tools
#build an agent to use the tools
#verbose = True print out more adjustment info
#allow_parallel = False indicates only one agent in use at the same time
#CUSTOM_PROMPT gives a lead in model behaviour
```

5 Voice Page

```
@app.route("/voice", methods=['POST'])
def voice():
    global to_say
    print(f"to_say: {to_say}")
    resp = VoiceResponse()

    gather = Gather(
        input="speech",
        action="/handle-speech",
        method="POST",
        speechTimeout="1",
        speechModel="experimental_conversations",
        enhanced=True
    )
    gather.say(to_say)
    resp.append(gather)

    resp.redirect("/voice")

    return str(resp)
```

```
#use global variable to_say as to be the text said by the Chat Agent
# resp=VoiceResponse() connect with the Twilio

#use gather to collect the speech
#action='/handle-speech' redirect to the /handle_speech page after input
#speechTimeout=1  wait for 1 sec after input

#use gather to say to_say and resp to write in gather
# resp.redirect("/voice") make a loop in the page
```

6 handle speech page

```
@app.route("/handle-speech", methods=['POST'])
async def handle_speech():
    global to_say
    resp = VoiceResponse()

    speech_result = request.form.get('SpeechResult')

    if speech_result:
        agent_response = agent.chat(speech_result)
        print(f"User: {speech_result}")
        print(f"Assistant: {agent_response}")
        to_say = str(agent_response)
        resp.redirect("/voice")
    else:
        resp.say("I'm sorry, I didn't catch that. Could you please repeat?")
        resp.redirect("/voice")
    return str(resp)
```

```
#get the params SpeechResult by form

#if-else condition indicates the speech result has been gotten
# True: send and get the result to/from chat agent thus redirect to /voice
# False: say error message
```