You want a **step-by-step example** of building an AI calling agent using **FreeSWITCH**, integrating a LLaMA-based local LLM for AI responses, speech-to-text and text-to-speech, and handling calls.

Quick overview:

- **FreeSWITCH** handles SIP calls.
- On incoming calls, it records caller audio.
- Convert audio to text with **Vosk** (offline STT).
- Pass text to local **LLaMA** model (via GPT4All Python).
- Convert AI text reply to speech using **espeak** or another TTS.
- Play the response back to caller.
- Provide a simple **web UI** to monitor / chat.

Step 1: Install FreeSWITCH on Windows

FreeSWITCH has Windows binaries:

- Download installer: https://files.freeswitch.org/windows/
- Install and run FreeSWITCH service.

Step 2: Configure FreeSWITCH to answer calls and run your AI script

FreeSWITCH uses XML configs, and you can hook your Python script via **mod_event_socket** or by running scripts from the dialplan.

We will use a dialplan to answer and execute an external script to do the AI handling.

Step 3: Enable mod_event_socket (allows external control)

1. Edit `autoload_configs/event_socket.conf.xml`:

```
<configuration name="event_socket.conf" description="Event Socket">
  <settings>
    <param name="listen-ip" value="127.0.0.1"/>
    <param name="listen-port" value="8021"/>
    <param name="password" value="ClueCon"/>
  </settings>
</configuration>
```

2. Restart FreeSWITCH.

Step 4: Create dialplan entry

Edit `dialplan/default.xml` to handle incoming calls:

```
<extension name="AI Calling Agent">
  <condition field="destination_number" expression="^1000$">
    <action application="answer"/>
    <action application="sleep" data="1000"/>
    <action application="lua" data="llama_agent.lua"/>
    <action application="hangup"/>
  </condition>
</extension>
```

Here we call a Lua script `llama_agent.lua` to run the AI call flow.

Step 5: Write Lua script to handle AI call logic

Create `llama_agent.lua` in `scripts/` folder:

```
-- llama_agent.lua
local session = freeswitch.Session("{ignore_early_media=true}1000")
if not session:ready() then return end

-- Play beep to prompt caller
session:execute("playback", "tone_stream://%(200,0,640)")

-- Record caller audio to file
local wav_file = "/tmp/caller.wav"
session:execute("record", wav_file .. " 5000")  -- record 5 seconds max
```

```
-- Call external Python script to do STT, AI, TTS and return response wav
file
local handle = io.popen("python llama_agent.py " .. wav_file)
local response_wav = handle:read("*a")
handle:close()

-- Play AI response audio
session:execute("playback", response_wav)

session:hangup()
```

Step 6: Write the Python script `llama_agent.py` to:

- Convert audio (`caller.wav`) to text (Vosk)
- Pass text to LLaMA/GPT4All model
- Convert AI reply text to speech (eSpeak)
- Output the response wav file path to stdout (for Lua script)

---

llama_agent.py

```
import sys
import wave
import json
import os
import subprocess
from vosk import Model, KaldiRecognizer
from gpt4all import GPT4All

def speech_to_text(wav_path):
    wf = wave.open(wav_path, "rb")
    model = Model("model")  # Vosk model folder path
    rec = KaldiRecognizer(model, wf.getframerate())

    results = []
    while True:
        data = wf.readframes(4000)
        if len(data) == 0:
            break
        if rec.AcceptWaveform(data):
            results.append(json.loads(rec.Result())['text'])
    results.append(json.loads(rec.FinalResult())['text'])
    return " ".join(results)

def text_to_speech(text, out_wav):
    # Using espeak to generate wav file
    subprocess.run(['espeak', '-w', out_wav, text])

def main():
    caller_wav = sys.argv[1]
```

```
    text = speech_to_text(caller_wav)

    # Load GPT4All model
    model = GPT4All("gpt4all-lora-quantized.bin")
    response = model.generate(text, max_tokens=100)

    response_wav = "/tmp/response.wav"
    text_to_speech(response, response_wav)

    # Output wav filename so Lua script can play it
    print(response_wav)

if __name__ == "__main__":
    main()
```

Step 7: Prepare dependencies

- Install Python packages:

```
pip install vosk gpt4all
```

- Download Vosk model: https://alphacephei.com/vosk/models and unzip folder `model` next to `llama_agent.py`
- Download GPT4All model binary `gpt4all-lora-quantized.bin` to your script folder.
- Install **eSpeak** TTS and make sure `espeak` command is available in your PATH.

Step 8: Testing your AI agent

- Place a SIP softphone and register with FreeSWITCH.
- Call extension `1000`.
- Your AI Lua script will record your voice, convert it to text, generate an AI response, synthesize it, and play it back.

Optional: Step 9 Web UI (Flask)

```
from flask import Flask, request, jsonify
from gpt4all import GPT4All
```

```
app = Flask(__name__)
model = GPT4All("gpt4all-lora-quantized.bin")

@app.route('/chat', methods=['POST'])
def chat():
    data = request.json
    user_text = data.get("text", "")
    response = model.generate(user_text, max_tokens=100)
    return jsonify({"response": response})

if __name__ == "__main__":
    app.run(port=5000)
```

Summary

- You run FreeSWITCH on Windows.
- Incoming call triggers Lua script.
- Lua script records audio, calls Python script.
- Python script does STT + LLaMA AI + TTS.
- Response played back to caller.