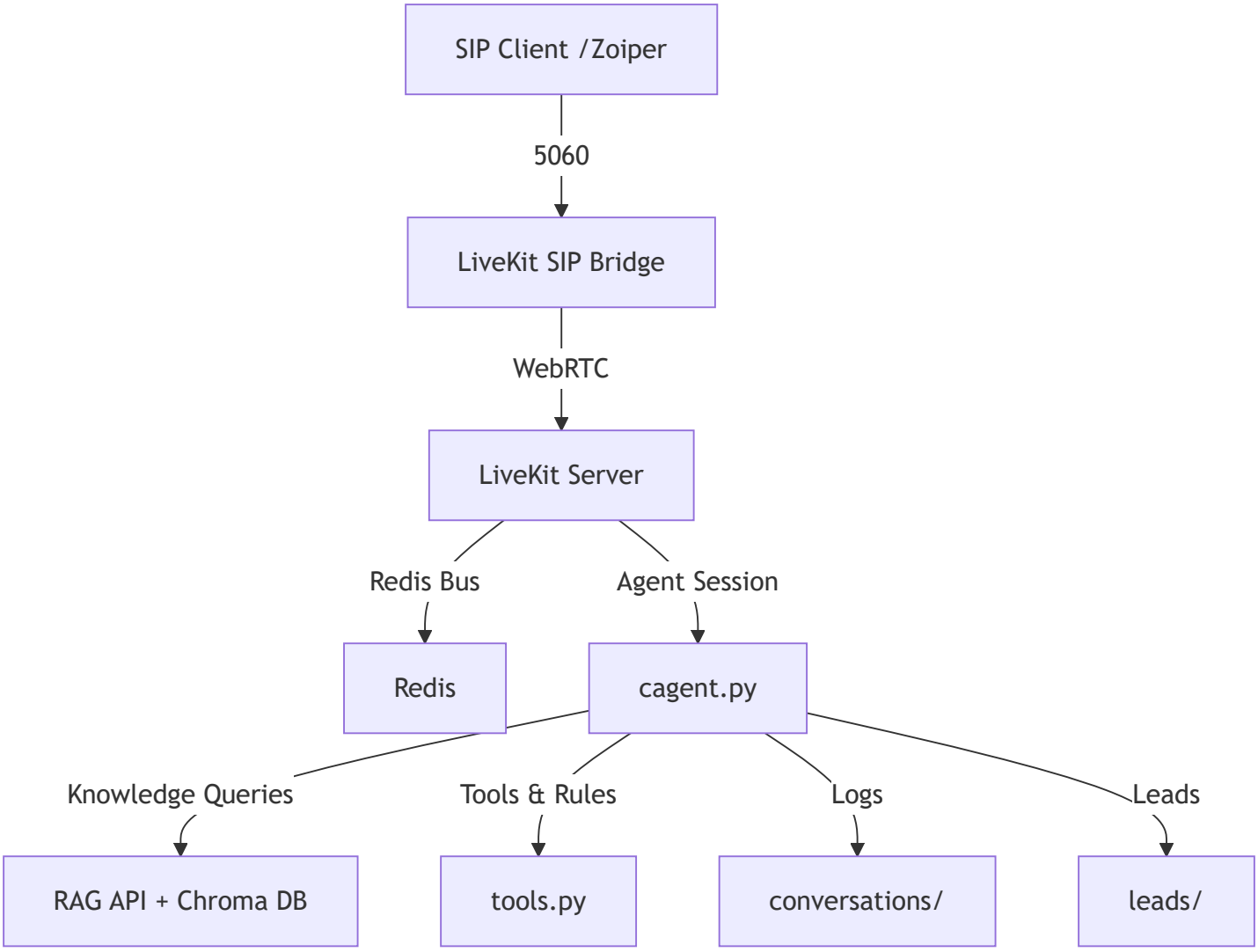
Friday AI — Voice Agent with SIP Telephony + RAG

Friday AI is an intelligent voice assistant built for **Triotech Bizserve Pvt. Ltd.**
It combines SIP telephony, RAG-based knowledge retrieval, and lead capture automation, all powered by LiveKit infrastructure.

🚀 Features

- **SIP Telephony Integration** (Zoiper ↔ LiveKit ↔ AI Agent)
- **Hybrid Knowledge System** (JSON + RAG/ChromaDB)
- **Real-Time Voice Communication** via WebRTC & Redis
- **Lead Management & Detection** in Hinglish
- **Conversation Logging & Analytics**
- **REST API** for RAG Queries
- **Plugin-Ready Architecture** for STT, TTS, and LLM providers

🧩 System Overview



⚙️ Deployment Guide

This guide details the manual steps to deploy the full stack. For one-click deployment, use the provided `setup.sh` script.

Step 1: System Prerequisites

Update your system and install the required packages.

```
sudo apt update
sudo apt install -y curl wget git redis-server python3 python3-venv python3-pip screen
```

Step 2: Setup Redis

Enable and start the Redis server.

```
sudo systemctl enable redis-server
sudo systemctl start redis-server
redis-cli ping  # Should return PONG
```

Step 3: Install Application & Dependencies

Clone the repository and set up the Python virtual environment.

```
# Clone your repository
git clone <your-repo-url>
cd <your-repo-directory>

# Setup Python environment
python3 -m venv venv
source venv/bin/activate
pip install --upgrade pip
pip install -r requirements.txt
```

Step 4: Install LiveKit Components

Download and install the specific versions of the LiveKit server, SIP bridge, and CLI.

```
# Install LiveKit Server v1.9.1
wget -q https://github.com/livekit/livekit/releases/download/v1.9.1/livekit-server_1.9.1_linux_a
tar -xzf livekit-server_1.9.1_linux_amd64.tar.gz
sudo mv livekit-server /usr/local/bin/
sudo chmod +x /usr/local/bin/livekit-server

# Install LiveKit SIP Bridge v1.5.1
wget -q https://github.com/livekit/livekit-sip/releases/download/v1.5.1/livekit-sip_1.5.1_linux_
tar -xzf livekit-sip_1.5.1_linux_amd64.tar.gz
sudo mv livekit-sip /usr/local/bin/
sudo chmod +x /usr/local/bin/livekit-sip

# Install LiveKit CLI v1.5.2
wget -q https://github.com/livekit/livekit-cli/releases/download/v1.5.2/livekit-cli_1.5.2_linux_
tar -xzf livekit-cli_1.5.2_linux_amd64.tar.gz
sudo mv livekit-cli /usr/local/bin/
sudo chmod +x /usr/local/bin/livekit-cli

# Create convenient alias
echo 'alias lk="livekit-cli"' >> ~/.bashrc
source ~/.bashrc
```

Step 5: Start Services with `screen`

We will use `screen` to run each service in a detached session.

```
# Start LiveKit Server
screen -dmS livekit-server bash -c "livekit-server --config livekit.yaml"

# Start LiveKit SIP Bridge
screen -dmS sip-bridge bash -c "livekit-sip --config sip-setup/config.yaml"

# Start the Python Backend Agent
screen -dmS backend bash -c "cd $(pwd) && source venv/bin/activate && python cagent.py"
```

To see your running services, use `screen -ls`. To attach to a session, use `screen -r <session_name>`.

Step 6: Configure SIP Routing (One-Time Setup)

After the services are running, configure the LiveKit project to route SIP calls to the agent.

```
# 1. Add a new LiveKit project
lk project add --name friday --url ws://127.0.0.1:7880 --api-key APIntavBoHTqApw --api-secret pl

# 2. Create an inbound SIP trunk
lk sip inbound create --project friday sip-setup/inbound_trunk.json

# 3. Create the dispatch rule to route calls
lk sip dispatch create --project friday sip-setup/sip_dispatch.json

# 4. Verify installation
echo "✅ Versions:"
livekit-server --version
livekit-sip --version
livekit-cli --version

# 5. Check running ports
sudo netstat -tunlp | grep -E '7880|5060|6379'
```

📞 SIP Client Setup (Zoiper)

Setting	Value
Host	`<YOUR-SERVER-IP>`
Port	`5060`
Username	`1001`
Password	`1001`
Protocol	`SIP (UDP)`

> **Note:** Once registered, dial any number to connect to the Friday AI agent.

🧠 RAG Web API (Optional)

```
python model/runapi.py
```

Accessible at: `http://localhost:5000`

🏗️ Project Structure

```
friday-ai/
├── cagent.py                # Main agent
├── tools.py                 # Business logic tools
├── prompts.py               # Hinglish prompts & lead rules
├── config.py                # Shared configuration
├── model/
│   ├── build_db.py          # Vector DB builder
│   └── runapi.py            # RAG API server
├── sip-setup/
│   ├── config.yaml          # SIP bridge config
│   ├── inbound_trunk.json   # Inbound trunk definition
│   └── sip_dispatch.json    # Dispatch rules
├── conversations/           # Saved chat logs
├── leads/                   # Detected leads
└── requirements.txt
```

🔧 Configuration Files

File	Purpose
`.env`	API keys & environment variables
`livekit.yaml`	LiveKit server setup
`sip-setup/config.yaml`	SIP bridge configuration
`inbound_trunk.json`	Defines SIP user credentials
`sip_dispatch.json`	Routes incoming calls to the agent's room

Required API Keys

- `LIVEKIT_API_KEY` , `LIVEKIT_API_SECRET`
- `GOOGLE_API_KEY` (Gemini)
- `HUGGINGFACE_API_KEY` (Embeddings)

🧪 Testing

```
python test_triotech_assistant.py
python test_dummy_plugins.py
python test_lead_detection.py
```

💻 Service Management

Check Running Services

```
# List all screen sessions
screen -ls

# Check specific ports
sudo netstat -tunlp | grep -E '7880|5060|6379'

# Verify Redis connection
redis-cli ping
```

Attach to Services

```
# Attach to LiveKit server logs
screen -r livekit-server

# Attach to SIP bridge logs
screen -r sip-bridge

# Attach to agent logs
screen -r backend

# Detach from session: Ctrl+A, then D
```

Stop Services

```
# Kill specific screen sessions
screen -S livekit-server -X quit
screen -S sip-bridge -X quit
screen -S backend -X quit
```

📄 License