Voice-Operated OTC Trading Bot

Contents

Overview

This readme file provides a comprehensive guide to building a high-performance, voice-operated trading bot that leverages the Bland.ai platform to facilitate Over-the-Counter (OTC) digital asset trades. The bot simulates interactions with an OTC crypto trading desk, enabling users to place orders through natural, voice-based conversations. Designed for a web-based interface, this system allows users to interact seamlessly using their computer's microphone and speakers, ensuring an intuitive and accessible experience.

The bot is engineered to handle fluid, natural conversations, supporting advanced features such as user corrections (e.g., "I meant to say Bitcoin, not Ethereum") and clarifying questions to enhance the interaction flow. Additionally, it implements contextual re-prompting logic to manage incomplete user inputs—for instance, if a user provides only a quantity without a price, the bot intelligently re-prompts for the missing details. Follow each section in order to set up, configure, and run the bot, ensuring alignment with your placement assessment requirements.

1 Project File Structure

Below is the project directory structure as provided:

```
- GOQUANT
  - .venv
  - voice-trading-bot
    - templates
      - index.html
    - venv
      - bin
      - include
      - lib
    - pyvenv.cfg
  - .env
  - app.py
  - ngrok-v3-stable-d...
  - voice_trading.log
  - ngrok.zip
```

2 Prerequisites

Before beginning, ensure you have:

- **macOS** with Homebrew installed (or Linux; adapt commands accordingly).
- **Internet** access (may require VPN for some exchange APIs).
- **VS Code** (or any code editor) for editing files and integrated terminal.
- **Bland.ai** account (free "Start" plan provides testing calls).
- **ngrok** for exposing local webhooks publicly.

3 Project Setup & Installation

Follow these steps sequentially.

3.1 Install Python 3.10 (macOS)

Open Terminal in VS Code or macOS Terminal.

Install Python 3.10 via Homebrew and verify the installation:

```
brew install python@3.10
python3 --version
```

Expect: Python 3.10.x

Linux Users: Use your package manager, e.g., `sudo apt install python3.10 python3.10-venv python3.10-dev`.

3.2 Create Project Directory

```
mkdir voice-trading-bot
cd voice-trading-bot
```

3.3 Create & Activate Virtual Environment

```
python3 -m venv venv
source venv/bin/activate
```

For macOS/Linux. On Windows PowerShell, use: `.\venv\Scripts\Activate.ps1`

Your prompt should show `(venv)`.

3.4 Create `requirements.txt`

In project root, create a file named `requirements.txt` containing:

```
flask
requests
python-dotenv
ccxt
```

You may optionally add `ngrok-client` if you plan to control ngrok from Python, but manual ngrok installation is typical.

3.5 Install Dependencies

```
pip install -r requirements.txt
```

Verify installation completes without errors.

3.6 Install ngrok

macOS (Homebrew):

```
brew install --cask ngrok
```

Linux/Windows: Download from https://ngrok.com/, unzip the binary, and place it in your PATH or project folder.

Verify the installation:

```
ngrok version
```

3.7 Initialize Git & .gitignore (Optional)

If using version control:

```
git init
cat <<EOF > .gitignore
venv/
.env
voice_trading.log
EOF
```

4 Environment Configuration

4.1 Create `.env` File

In project root, create a `.env` file with:

```
BLAND_API_KEY=your_bland_api_key_here
NGROK_URL=
```

- Replace `your_bland_api_key_here` with the key from Bland.ai dashboard.
- Leave `NGROK_URL=` blank for now; it will be set after starting ngrok.

Important: Add `.env` to `.gitignore` to avoid leaking secrets.

4.2 Obtain Bland.ai API Key

1. Sign up / log in at https://bland.ai.
2. Navigate to Developer/API section.
3. Generate or copy your API key.
4. Paste into `.env` as `BLAND_API_KEY=...`.

4.3 Verify `load_dotenv()` in Code

Ensure that your backend code calls `load_dotenv()` before reading environment variables.

5 Backend Implementation (Flask)

This section details how to create `app.py` to handle: serving frontend, initiating Bland.ai voice calls, receiving webhooks, managing conversation state, and fetching market data via CCXT.

5.1 `app.py` Snippet

Below is a small portion of the backend code for reference:

```
import os
import logging
import uuid
from flask import Flask, request, jsonify, render_template
from dotenv import load_dotenv
import ccxt
import requests

```

```
def load_environment():
    from pathlib import Path
    if Path('.env').exists():
        load_dotenv()
load_environment()
BLAND_API_KEY = os.getenv('BLAND_API_KEY')
NGROK_URL = os.getenv('NGROK_URL')

app = Flask(__name__)
```

The complete backend code for ***app.py*** *is in the provided code.*

Note: Adjust Bland.ai start-call endpoint and payload according to latest Bland.ai documentation.

6 Frontend Implementation (HTML/JS)

Create `templates/index.html` and optional `static/script.js` to manage the UI.

6.1 `templates/index.html` Snippet

Below is a small portion of the frontend code for reference:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"
      />
  <title>Voice Trading Bot</title>
  <style>
    body { font-family: Arial, sans-serif; margin: 2rem; }
    #controls { margin-bottom: 1rem; }
  </style>
</head>
<body>
  <h1>Voice Trading Bot</h1>
```

The complete frontend code for ***index.html*** *is in the provided code.*

Integrate the Bland.ai JavaScript or WebSocket SDK according to their documentation. Ensure microphone permission is requested.

7 Running & Testing Step-by-Step

Follow exactly to avoid missing steps.

7.1 Activate Virtual Environment

```
cd voice-trading-bot
source venv/bin/activate
```

7.2 Start Flask Backend

```
python3 app.py
```

Verify terminal shows Flask running on `http://0.0.0.0:5000`.

7.3 Test Backend Endpoint (Optional)

In another terminal:

```
curl -X POST http://localhost:5000/start-call
```

Expect: {"call_id":"<UUID>"}.

7.4 Start ngrok Tunnel

In a separate terminal (keep Flask running):

```
ngrok http 5000
```

Copy the HTTPS forwarding URL (e.g., `https://abc123.ngrok.io`).

7.5 Update `.env` with Ngrok URL

Open `.env` and set:

```
NGROK_URL=https://abc123.ngrok.io
```

7.6 Restart Flask

Stop Flask (`Ctrl+C`) and restart:

```
python3 app.py
```

This reloads environment variables.

7.7 Configure Bland.ai Webhook

- In Bland.ai dashboard or as part of start-call payload, ensure the webhook URL is `https://abc123.ngrok.io/webhook`.
- Verify that Bland.ai can reach your `/webhook` endpoint.

7.8 Open Frontend

In browser: `http://localhost:5000`.

7.9 Perform Voice Interaction

1. Click **Start Call**; allow microphone access.
2. Speak clearly at each prompt:
   - Bot: "Which exchange would you like?" → Say: "Binance"
   - Bot: "Please state the trading symbol." → Say: "BTCUSDT"
   - Bot: "Current price is X. Please state quantity and desired price." → Say: "1 at 50000"
   - Bot: "Confirm Buy 1 BTCUSDT at 50000?" → Say: "Yes"
3. Observe bot confirmations and transcript in the UI.

Check the Flask logs (`voice_trading.log` or console) to trace session state and debug if needed.

8 Maintaining ngrok & Flask Session

- **Ngrok Free Plan**: Tunnel URL changes every ∼2 hours. Each time:
  1. Restart `ngrok http 5000`.
  2. Copy new HTTPS URL.
  3. Update `NGROK_URL` in `.env`.
  4. Restart Flask (`Ctrl+C` → `python3 app.py`).
- **Reminder**: If you fail to update `NGROK_URL`, Bland.ai webhooks will fail.

8.1 Optional: Automate ngrok

- **Authtoken Setup** (run once):

```
ngrok config add-authtoken YOUR_AUTHTOKEN_HERE
```

- **ngrok.yml**: Use for fixed configuration, custom subdomains (paid plan).
- **Automation Script**: Write a shell script to kill and restart Flask and prompt for new ngrok URL.

9 Error Handling & Logging

- **Invalid Exchange**: Bot replies: "Please choose a valid exchange: Binance, OKX, Bybit, or Deribit." Remain in exchange selection step.
- **Symbol Not Found**: Bot replies: "Symbol not found. Please provide a valid trading pair, e.g., BTCUSDT." Stay in symbol step.
- **Price Fetch Failure**: Bot replies: "Unable to fetch current price. Please try another symbol or try later." Optionally go back to symbol step.
- **Incomplete Order Details**: Bot prompts: "Please specify both quantity and price, e.g., '2 at 50000'."
- **Confirmation**: Only accept clear "Yes" or "No". On unclear response, re-ask.
- **Session Timeouts**: (Optional) Implement inactivity timer; after timeout, clear session and notify user.
- **Logging**: All user inputs, bot prompts, errors logged via Python `logging` to `voice_trading.log` and console.

Example logging setup included in `app.py`.

10 Advanced / Bonus Features

- **Natural Language Corrections**: Detect phrases like "I meant Ethereum" to adjust previous selection. Requires NLP parsing or simple keyword detection.
- **Contextual Re-Prompting**: Track which info is missing (exchange/symbol/quantity/price) and ask only for missing parts.

- **Persistent Sessions**: Use Redis or a database to persist session state across Flask restarts.
- **UI Enhancements**: Display quick-reply buttons for exchanges or frequent symbols to reduce speech recognition errors.
- **Automated Tests**: Write pytest tests for helper functions (parsing, state transitions).
- **Dockerization**: Create a `Dockerfile` to containerize the Flask app. Ngrok can be run separately or within container.
- **CI/CD**: Integrate linting and tests in a CI pipeline.

11 Troubleshooting

- **Microphone Access Denied**: Check browser settings to allow microphone on `localhost`.
- **Blank or No Bot Response**: Ensure Flask is running and reachable; confirm Bland.ai webhook configured with correct ngrok URL; check logs.
- **Ngrok Not Forwarding**: Verify ngrok is running on correct port (5000) and firewall allows traffic.
- **Environment Variables Not Loaded**: Confirm `.env` exists and `load_dotenv()` is called; restart Flask after changes.
- **Exchange API Blocked**: Use a VPN if your IP is blocked by certain exchanges; handle CCXT exceptions.
- **Port Conflicts**: If port 5000 is in use, change Flask port in `app.run(port=NEW_PORT)` and update ngrok accordingly.

12 Security & Best Practices

- **Protect API Keys**: Do not commit `.env`; add to `.gitignore`.
- **HTTPS in Production**: Use a proper domain with TLS; verify Bland.ai webhook signatures if provided.
- **Rate Limits**: Cache fetched market data if needed to avoid hitting API rate limits during rapid testing.
- **Resource Cleanup**: Clear session state after confirmation or timeout to avoid memory buildup.
- **Input Sanitization**: Though this is a simulation, validate user inputs carefully.

13 Contact & Support

For questions or issues, reference this README and logs. Provide clear reproduction steps and console outputs when seeking help.