Here’s the **complete, updated, enterprise-level guide** for implementing **Voice-Guided Git Automation using Hybrid Speech Recognition (Offline + Online)** with **full updated source code**, **step-by-step setup**, **installations**, **detailed explanations**, and **outputs**.

**🚀 Voice-Guided Git Automation (Offline + Online ASR with Voice Mode Selection)**

This project allows developers to **control Git using voice commands**.  
It integrates **offline ASR (Vosk)** and **online ASR (Google Speech)** with **automatic mode selection via voice**, **Git integration**, **logging**, **TTS feedback**, and **Streamlit GUI logs**.

**🔥 Key Features**

✅ Hybrid ASR (Offline Vosk + Online Google Speech)  
✅ Voice-based ASR mode selection (**Speak "offline" or "online"**)  
✅ NLP intent parser (convert natural language to Git commands)  
✅ Voice feedback (TTS confirmations)  
✅ GitPython/CLI integration for executing Git safely  
✅ Undo last commit, branch creation, status, push, pull, exit  
✅ Logging all commands in logs/commands.txt  
✅ Streamlit dashboard for logs  
✅ Keyboard interrupt (Ctrl+C) safe exit

**1️⃣ PROJECT STRUCTURE**

voice\_git\_automation/

│── main.py # Main orchestrator (ASR mode selection + execution loop)

│── asr\_engine.py # Hybrid ASR (offline/online)

│── nlp\_parser.py # NLP intent parser (voice → Git commands)

│── git\_executor.py # Git commands executor

│── voice\_feedback.py # TTS + Logging

│── gui\_dashboard.py # Streamlit GUI logs

│── requirements.txt # All dependencies

│── models/

│ └── vosk-model-small-en-us-0.15/ # Offline ASR model

│── logs/

│ └── commands.txt # Voice command logs

**2️⃣ INSTALLATION & SETUP (Step-by-Step)**

**✅ Step 1: Install Python & Git**

* Download Python 3.10+ from [python.org](https://www.python.org/downloads/).
* Install Git from [git-scm.com](https://git-scm.com/downloads).
* Verify:

python --version

git --version

**✅ Step 2: Create Project Folder**

mkdir voice\_git\_automation

cd voice\_git\_automation

python -m venv venv

Activate the virtual environment:

* **Windows (CMD/PowerShell):**

venv\Scripts\activate

* **Linux/Mac:**

source venv/bin/activate

**✅ Step 3: Install Required Packages**

Create requirements.txt:

vosk

openai-whisper

SpeechRecognition

pyaudio

spacy

GitPython

streamlit

pyttsx3

google-cloud-speech

librosa

numpy

scikit-learn

PyGithub

Install dependencies:

pip install -r requirements.txt

python -m spacy download en\_core\_web\_sm

**✅ Step 4: Download Offline ASR Model (Vosk)**

mkdir models

cd models

wget https://alphacephei.com/vosk/models/vosk-model-small-en-us-0.15.zip

unzip vosk-model-small-en-us-0.15.zip

cd ..

##windows##

# Create the models directory

mkdir models

# Change to the models directory

cd models

# Download the Vosk model ZIP file

Invoke-WebRequest -Uri "https://alphacephei.com/vosk/models/vosk-model-small-en-us-0.15.zip" -OutFile "vosk-model-small-en-us-0.15.zip"

# Extract the ZIP file

Expand-Archive -Path "vosk-model-small-en-us-0.15.zip" -DestinationPath "."

# Go back to previous directory

cd ..

**✅ Step 5: Initialize Git Repo**

Navigate to your working folder:

git initgit init

**3️⃣ FULL UPDATED SOURCE CODE**

**🔑 asr\_engine.py (Hybrid Speech Recognition)**

import speech\_recognition as sr

from vosk import Model, KaldiRecognizer

import json, pyaudio

# Offline ASR (Vosk)

def offline\_asr():

model = Model("models/vosk-model-small-en-us-0.15")

recognizer = KaldiRecognizer(model, 16000)

pa = pyaudio.PyAudio()

stream = pa.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True, frames\_per\_buffer=8192)

stream.start\_stream()

print("🎤 [Offline Mode] Speak now...")

while True:

data = stream.read(4096, exception\_on\_overflow=False)

if recognizer.AcceptWaveform(data):

result = json.loads(recognizer.Result())

return result['text']

# Online ASR (Google Speech)

def cloud\_asr():

r = sr.Recognizer()

with sr.Microphone() as source:

print("🎤 [Online Mode] Speak now...")

audio = r.listen(source)

try:

return r.recognize\_google(audio)

except sr.UnknownValueError:

print("❌ Could not understand speech.")

return ""

# Hybrid ASR (Offline or Online)

def hybrid\_asr(mode):

return offline\_asr() if mode == "offline" else cloud\_asr()

**🔑 nlp\_parser.py (Voice → Git Commands)**

import spacy, re

nlp = spacy.load("en\_core\_web\_sm")

def parse\_git\_command(command\_text):

command\_text = command\_text.lower()

# Undo commit first (priority)

if "undo" in command\_text or "revert" in command\_text:

return ["git reset --soft HEAD~1"]

# Commit changes

if "commit" in command\_text:

msg = re.search(r"message ['\"](.\*?)['\"]", command\_text)

commit\_msg = msg.group(1) if msg else "voice commit"

return [f"git add .", f'git commit -m "{commit\_msg}"']

# Push changes

elif "push" in command\_text:

return ["git push origin main"]

# Pull changes

elif "pull" in command\_text:

return ["git pull origin main"]

# Create branch

elif "branch" in command\_text and "create" in command\_text:

match = re.search(r"branch ([\w\-]+)", command\_text)

if match:

branch\_name = match.group(1)

return [f"git branch {branch\_name}", f"git checkout {branch\_name}"]

# Switch branch

elif "switch" in command\_text or "checkout" in command\_text:

match = re.search(r"branch ([\w\-]+)", command\_text)

if match:

branch\_name = match.group(1)

return [f"git checkout {branch\_name}"]

# Show status

elif "status" in command\_text:

return ["git status"]

# Exit

elif any(word in command\_text for word in ["exit", "quit", "stop", "close"]):

return ["exit"]

return ["Unknown command"]

**🔑 git\_executor.py (Execute Git Commands)**

import subprocess

def execute\_git\_commands(commands):

for cmd in commands:

if cmd == "exit": # Handle exit

return "exit"

print(f"⚡ Running: {cmd}")

process = subprocess.run(cmd, shell=True, capture\_output=True, text=True)

if process.returncode == 0:

print(process.stdout)

else:

print(f"❌ Error: {process.stderr}")

**🔑 voice\_feedback.py (TTS + Logs)**

import pyttsx3, datetime, os

engine = pyttsx3.init()

def speak(text):

engine.say(text)

engine.runAndWait()

def log\_command(command):

os.makedirs("logs", exist\_ok=True)

with open("logs/commands.txt", "a") as f:

f.write(f"{datetime.datetime.now()} - {command}\n")

**🔑 gui\_dashboard.py (Streamlit Logs)**

import streamlit as st

def display\_logs():

st.title("📜 Voice-Git Automation Logs")

with open("logs/commands.txt", "r") as f:

st.text(f.read())

if \_\_name\_\_ == "\_\_main\_\_":

display\_logs()

**🔑 main.py (Main Orchestrator with Voice Mode Selection)**

from asr\_engine import hybrid\_asr

from nlp\_parser import parse\_git\_command

from git\_executor import execute\_git\_commands

from voice\_feedback import speak, log\_command

print("🚀 Voice-Git Automation Started")

speak("Please say offline or online to select A S R mode")

# Voice-based ASR mode selection

while True:

print("🎤 Please say 'offline' or 'online' to select ASR mode...")

mode\_choice = hybrid\_asr("online").lower().strip()

print(f"🎤 Recognized (Mode Selection): {mode\_choice}")

if "offline" in mode\_choice:

mode = "offline"

print("✅ Selected ASR Mode: Offline")

speak("Selected ASR Mode Offline")

break

elif "online" in mode\_choice:

mode = "online"

print("✅ Selected ASR Mode: Online")

speak("Selected ASR Mode Online")

break

else:

speak("I could not detect the mode. Please say offline or online.")

# Main loop

try:

while True:

text = hybrid\_asr(mode)

if not text:

continue

print(f"🎤 [{mode.capitalize()} Mode] Recognized: {text}")

commands = parse\_git\_command(text)

if commands[0] == "exit":

print("👋 Exiting Voice-Git Automation.")

speak("Exiting Voice Git Automation.")

break

if commands[0] == "Unknown command":

print("❌ Unknown command.")

speak("Unknown Git command.")

continue

log\_command(text)

speak(f"Executing {', '.join(commands)}")

result = execute\_git\_commands(commands)

if result == "exit": break

speak("Command executed successfully.")

print("✅ Command executed successfully.")

except KeyboardInterrupt:

print("\n👋 Exiting (Keyboard Interrupt).")

speak("Exiting Voice Git Automation.")

**4️⃣ HOW TO RUN**

**Start Voice Git:**

python main.py

✅ Speak **"offline"** or **"online"** to select ASR mode.  
✅ Then speak Git commands like:

* *"Commit all changes with message 'initial setup'"*
* *"Push changes"*
* *"Undo last commit"*
* *"Create branch feature-login"*
* *"Exit"*

**View Logs in Browser:**

streamlit run gui\_dashboard.py

Opens a browser showing command history.

**5️⃣ OUTPUT EXAMPLE**

🚀 Voice-Git Automation Started

🎤 Please say 'offline' or 'online' to select ASR mode...

🎤 Recognized (Mode Selection): online

✅ Selected ASR Mode: Online

🎤 [Online Mode] Speak now...

🎤 Recognized: commit all changes with message "initial setup"

⚡ Running: git add .

⚡ Running: git commit -m "initial setup"

✅ Command executed successfully.

**✅ Now You Have:**

✔ Full Hybrid ASR (voice-based mode selection)  
✔ Git voice commands with undo, branches, status, push, pull  
✔ Logging + GUI  
✔ TTS confirmations

Would you like me to **add GitHub PR creation (voice-triggered)** as the **next feature**?