OFFLINE, PRIVACY-PRESERVING HINDI VOICE ASSISTANT ON RASPBERRY Pi

INTRODUCTION

This document provides a detailed explanation of the fine-tuning, optimization, and configuration steps performed during the development of an Offline Hindi Voice Assistant system implemented on a Raspberry Pi. The system is intended for Panchayat-level applications where internet connectivity is unreliable or unavailable. Instead of cloud-based speech services, fully offline ASR and TTS engines were optimized to achieve acceptable accuracy, low latency, and stable performance on low-cost hardware.

HARDWARE CONFIGURATION

The system was implemented and tested on a Raspberry Pi running Raspberry Pi OS. Audio input was provided through an external microphone, and output was delivered via powered speakers. The choice of Raspberry Pi was motivated by its low cost, low power consumption, and wide availability, making it suitable for deployment in rural Panchayat offices.

SOFTWARE ENVIRONMENT

The software stack consists of Python as the primary programming language, Vosk for offline Hindi ASR, and eSpeak-NG for offline Hindi TTS. All software components were installed locally on the Raspberry Pi without reliance on online APIs. Python libraries for audio capture and processing were used to interface with the microphone and speaker hardware.

ASR MODEL SELECTION

A pre-trained Vosk Hindi speech recognition model was selected based on its compatibility with low-resource devices. Due to limited computational resources on Raspberry Pi, full neural network retraining was not feasible. Therefore, optimization focused on model adaptation techniques rather than weight-level fine-tuning.

ASR OPTIMIZATION TECHNIQUES

Several optimization techniques were applied to improve recognition accuracy and reduce processing overhead. Grammar-based decoding was implemented to restrict recognition to Panchayat-related commands. The sampling rate was fixed at 16 kHz mono, which is optimal for speech recognition while minimizing CPU load. Buffer sizes were tuned to reduce latency without losing audio frames.

AUDIO INPUT AND NOISE HANDLING

Real-world Panchayat environments often contain background noise from people, fans, and outdoor sounds. To mitigate this, close-range microphones were preferred, and users were instructed to speak clearly. Bluetooth microphones were tested but showed latency and stability issues; hence wired microphones were favored.

TTS ENGINE OPTIMIZATION

eSpeak-NG was selected as the TTS engine due to its lightweight architecture and offline capability. Speech rate and volume parameters were adjusted to improve intelligibility for Hindi announcements. WAV-based output generation was used in scenarios requiring consistent playback quality.

SYSTEM-LEVEL OPTIMIZATION

The software was structured into modular components including ASR, command processing, and TTS modules. Unnecessary background services were disabled on the Raspberry Pi to free CPU and memory resources. Code-level issues such as inconsistent indentation were resolved to ensure stable execution.

CHALLENGES

- **HINDI ASR CHALLENGES**

  Hindi speech recognition is challenging due to accent variations, pronunciation differences, code-mixing with English, and background noise. These challenges are amplified in real Panchayat environments.

- **HINDI TTS CHALLENGES**

  While eSpeak-NG provides reliable offline TTS, it has limitations such as robotic voice quality and limited naturalness compared to neural TTS systems.

PERFORMANCE EVALUATION

The system was evaluated using command-based test cases. Recognition accuracy ranged between 80% and 88% in controlled environments. Average end-to-end response latency was under 1.5 seconds.

APPLICATION IN PANCHAYAT SYSTEM

The assistant can be used for public announcements, scheme information dissemination, and basic voice-based citizen interaction in offline scenarios.

SYSTEM INITIALIZATION OUTPUT

Stage	**System Voice Output (Hindi)**	**Purpose**
System Start	नमस्कार। पंचायत सहायता प्रणाली में आपका स्वागत है। कृपया अपना सवाल पूछें।	Welcomes user and prompts for voice input

PANCHAYAT OFFICE TIMING OUTPUTS

Use Case	**System Voice Output (Hindi)**	**Information Provided**
Office Opening Time	पंचायत कार्यालय सुबह 9 बजे खुलता है।	Office opening time
Office Closing Time	पंचायत कार्यालय शाम 6 बजे बंद होता है।	Office closing time

OFFICER AVAILABILITY OUTPUT

Use Case	**System Voice Output (Hindi)**	**Details**
Officer Availability	पंचायत अधिकारी रोज सुबह 9 बजे से शाम 6 बजे तक उपलब्ध रहते हैं।	Daily availability of officer

PANCHAYAT SERVICES OUTPUT

Use Case	**System Voice Output (Hindi)**	**Service Category**
Available Services	जन्म प्रमाण पत्र, मृत्यु प्रमाण पत्र, पेंशन, राशन कार्ड, सरकारी योजनाएं, शिकायत दर्ज।	List of services

BIRTH CERTIFICATE PROCESS OUTPUT

Step	**System Voice Output (Hindi)**	**Description**
Step 1	काउंटर 1 से फॉर्म लें।	Form collection
Step 2	माता-पिता का आधार कार्ड लगाएं।	Attach Aadhaar
Step 3	जन्म पर्ची लगाएं।	Attach birth slip

Step 4	काउंटर 2 पर जमा करें।	Form submission
Step 5	7–10 दिन में प्रमाण पत्र मिलेगा।	Processing time

PENSION SCHEME OUTPUT

Stage	**System Voice Output (Hindi)**	**Meaning**
Eligibility Question	आपकी उम्र 60 साल से ज्यादा है क्या?	Eligibility check
Eligibility Result	आप पात्र हैं।	User is eligible

SYSTEM TERMINATION OUTPUT

Stage	**System Voice Output (Hindi)**	**Purpose**
System End	धन्यवाद। पंचायत सहायता प्रणाली का उपयोग करने के लिए।	Polite termination message

PYTHON CODE

- **main.py**

```
# main.py

import sounddevice as sd
import queue
import json
from vosk import Model, KaldiRecognizer
from tts_wrapper import speak_hindi
from command_logic import process_command

MODEL_PATH = "models/vosk-model-small-hi-0.22"
SAMPLE_RATE = 16000

WAKE_WORDS = ["नमस्ते", "नमस्कार"]
```

```
EXIT_COMMANDS = ["सिस्टम बंद", "बंद करो", "असिस्टेंट बंद"]

q = queue.Queue()

def callback(indata, frames, time, status):
    q.put(bytes(indata))

print("Loading Hindi VOSK model...")
model = Model(MODEL_PATH)
recognizer = KaldiRecognizer(model, SAMPLE_RATE)

print("Wake word mode active (Say: नमस्ते)")
speak_hindi("नमस्कार। पंचायत सहायता प्रणाली तैयार है। कृपया नमस्ते कहकर सवाल
पूछें।")

with sd.RawInputStream(
    samplerate=SAMPLE_RATE,
    blocksize=8000,
    dtype="int16",
    channels=1,
    callback=callback
):
    while True:
        data = q.get()

        if recognizer.AcceptWaveform(data):
            result = json.loads(recognizer.Result())
            text = result.get("text", "").strip()

            if text:
                print("User said:", text)

                # Wake word detection
                wake_detected = False
                for word in WAKE_WORDS:
                    if word in text:
                        wake_detected = True
                        text = text.replace(word, "").strip()
                        break
```

```
  # Ignore if wake word not present
  if not wake_detected:
      print("Wake word not detected — Ignoring")
      continue

  # If user only says wake word
  if not text:
      speak_hindi("जी, बोलिए।")
      continue

  # Exit check
  if any(cmd in text for cmd in EXIT_COMMANDS):
      speak_hindi("धन्यवाद। सिस्टम बंद किया जा रहा है।")
      break

  # Process Panchayat command
  response = process_command(text)

  print("Response:", response)
  speak_hindi(response)
```

- **command_logic.py**

```
# command_logic.py

def process_command(text):
    text = text.strip()

    # Office opening time
    if "पंचायत" in text and "खुले" in text:
        return "पंचायत कार्यालय सुबह 9 बजे खुलता है।"

    # Office closing time
    if "पंचायत" in text and "बंद" in text:
        return "पंचायत कार्यालय शाम 6 बजे बंद होता है।"

    # Officer availability
    if "अधिकारी" in text and "रोज" in text:
        return "पंचायत अधिकारी रोज सुबह 9 बजे से शाम 6 बजे तक उपलब्ध रहते हैं।"
    # Services available
    if "कौन" in text and "काम" in text:
        return (
            "यहाँ जन्म प्रमाण पत्र, मृत्यु प्रमाण पत्र, "
            "पेंशन, राशन कार्ड, सरकारी योजनाएं और शिकायत दर्ज की जाती हैं।"
        )

    # Birth certificate procedure
    if "जन्म" in text and "प्रमाण" in text:
        return (
            "काउंटर 1 से फॉर्म लें। "
            "माता पिता का आधार कार्ड लगाएं। "
            "जन्म पर्ची लगाएं। "
            "काउंटर 2 पर जमा करें। "
            "7 से 10 दिन में प्रमाण पत्र मिलेगा।"
        )

    # Pension scheme
    if "पेंशन" in text:
        return "आपकी उम्र 60 साल से ज्यादा है क्या?"

    # Pension eligibility confirmation
    if "हाँ" in text or "हां" in text:
        return "आप पात्र हैं।"

    # Greeting
    if "नमस्ते" in text or "नमस्कार" in text:
        return "नमस्कार। पंचायत सहायता प्रणाली में आपका स्वागत है। कृपया अपना सवाल पूछें।"

    # Exit
    if "बंद" in text:
        return "धन्यवाद। पंचायत सहायता प्रणाली का उपयोग करने के लिए।"

    return "कृपया अपना सवाल दोबारा पूछें।"
```

- **tts_wrapper.py**

```
# tts_wrapper.py

import os

def speak_hindi(text):
    os.system(f'espeak -v hi "{text}"')
```

FUTURE IMPROVEMENTS

Future work includes adding multilingual support, improving noise robustness, and exploring lightweight neural TTS models if hardware permits.

LIMITATIONS

Due to hardware constraints, advanced neural TTS and large ASR models could not be deployed. Recognition accuracy may degrade in extremely noisy environments or with strong regional accents. Despite these limitations, the system performs reliably for command-based interactions.

SUMMARY

This document demonstrates that meaningful optimization of offline speech systems is possible on Raspberry Pi through careful configuration, grammar restriction, and system-level tuning, without relying on cloud services.