DOCUMENT 1: MODEL FINE-TUNING AND OPTIMIZATION DOCUMENTATION

1. 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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.