Offline Hindi Voice Assistant Documentation

Project Description

The Offline Hindi Voice Assistant is a privacy-focused, fully offline conversational AI system designed to run efficiently on low-resource devices such as Raspberry Pi.
The system enables natural Hindi voice interaction without relying on any cloud services.It integrates wake word detection, speech recognition, hybrid intent classification, knowledge retrieval, and local language model inference into a unified pipeline

The assistant is optimized for low latency and real-time performance.
A hybrid NLU architecture ensures accurate intent detection using deterministic and machine learning approaches.
A structured knowledge base enables fast and reliable factual responses.
For unsupported queries, a locally hosted LLM generates intelligent responses in Hindi.
The entire system operates without internet connectivity, ensuring complete user privacy.

Modular architecture allows easy scalability and future improvements.
The project demonstrates practical deployment of edge AI for regional language voice assistants.

Key Features

1. Fully Offline Operation

- Runs completely offline
- No cloud APIs required
- Ensures user privacy and low latency

2. Wake Word Detection

- Supports wake phrases: “Hey Nova” and “Hyva”
- Fast partial-result detection using small audio blocks (1024 samples)
- Low-latency trigger activation

3. Noise-Resistant Speech Capture

- RMS-based noise gate to ignore silence and background noise
- Tokenization to ensure meaningful speech input
- Timeout handling to recover from incomplete speech

4. Hybrid Intent Recognition System

Deterministic NLU

- Exact keyword matching
- Fuzzy matching (Levenshtein distance ≤ 1)

Machine Learning Fallback

- Scikit-learn classifier

- Confidence threshold > 0.65

5. Knowledge Base Integration

- Structured domain-based knowledge:
    - History
    - Indian History
    - Politics
    - General Knowledge
- Exact and substring matching search
- Fast retrieval using Pickle-based storage

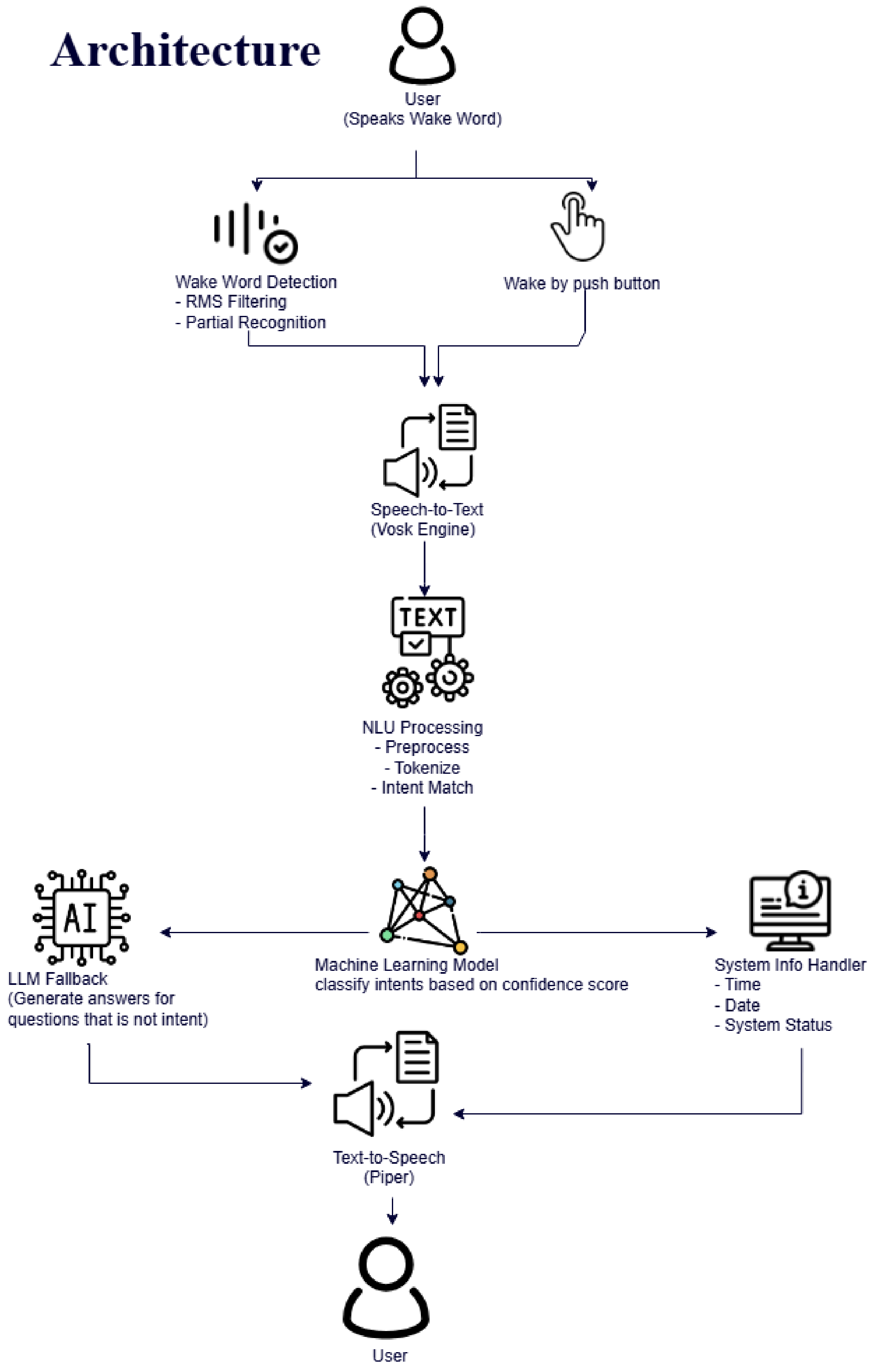
6. Intelligent LLM Fallback

- Local inference using Llama-3.2-1B-Instruct
- Handles general conversation
- Optimized for short, concise Hindi responses
- Runs on CPU (4 threads, 128-token context window)

7. Neural Hindi Text-to-Speech

- Piper TTS (hi_IN-priyamvada-medium)
- Fast WAV generation
- Platform-aware audio playback (Windows/Linux)

Architecture



Core Pipeline Flow

The assistant follows a modular, sequential pipeline optimized for low latency and fully offline execution.

Microphone Input
→ Wake Word Detection
→ Speech-to-Text
→ Intent Recognition
→ Knowledge Base or LLM Fallback
→ Text-to-Speech Output
→ Return to Wake State

Wake Word Detection

Purpose

Continuously listens for predefined activation keywords.

Some Wake Words

- Iva
- Kira
- Ava
- Hyva
- Taiwan

Function

- Captures microphone audio
- Processes audio in small chunks
- Uses Vosk speech recognition engine
- Detects wake words from partial recognition

Importance

Without wake word detection:

- High CPU usage
- Increased latency
- False activations
- Unnecessary noise processing

The wake word acts as a trigger to activate the assistant.

Output

- If detected → Transition to command listening mode
- Otherwise → Continue passive listening

Speech-to-Text (STT)

Engine Used

Vosk Speech Recognition Engine

Advantages

- Offline operation
- Low latency
- Lightweight
- Works efficiently on Raspberry Pi

Purpose

Convert user speech into text format for further processing.

Process

1. Capture microphone audio
2. Send audio to Vosk recognizer
3. Convert speech to text
4. Return recognized text

Example:
Input (Hindi speech): “समय क्या हुआ?”
Output (Text): “समय क्या हुआ?”

Intent Recognition (NLU)

Purpose

Determine the meaning or intent behind user input.

Example:
“समय क्या हुआ?” → Identified as time_query

Two-Layer Intent Detection System

Layer 1: Deterministic Matching

- Keyword matching
- Fuzzy matching
- intent.json mapping

Fast and reliable.

Example:
“समय” → time_query

Layer 2: Machine Learning Prediction

- Used if deterministic matching fails
- ML model loaded using pickle
- TF-IDF vectorization
- Classifier prediction

Improves accuracy for varied phrasing.

Knowledge Base

Overview

Stores predefined factual information in Hindi.
Used to answer general knowledge questions without using the LLM.
All data is stored in Pickle (.pkl) files as keyword–definition pairs.

Domains Covered

- History
- Indian History
- Politics
- World General Knowledge
- India General Knowledge

How It Works

1. System receives user query
2. Searches relevant knowledge files
3. If match found → Return stored answer
4. If no match → Proceed to LLM

Advantages

- Fast response
- Fully offline
- No heavy processing
- Reliable factual answers
- Reduces LLM usage

LLM Fallback

Purpose

Handles unknown or unsupported queries.

If no intent is confidently detected through:

- Deterministic matching
- Machine Learning prediction

The system forwards the query to the LLM.

Engine Used

- llama_cpp
- Model: Llama-3.2-1B-Instruct
- Fully offline

How It Works

1. User speaks in Hindi
2. Input text (Hindi) passed to LLM
3. LLM generates dynamic response in Hindi
4. Response sent to TTS

Advantages

- Flexible response generation
- Handles complex and open-ended questions
- Fully offline execution
- Supports natural Hindi conversation

Text-to-Speech (TTS)

Purpose

Convert response text into audible speech.

Engine Used

Piper TTS

Process

Text → Piper conversion → Audio playback

Main Control Flow

Initialize System

- Load Vosk models (wake + STT)
- Load intents (intent.json)
- Load ML model (intent_model.pkl)
- Load knowledge base (*.pkl)
- Load Llama model
- Initialize Piper TTS

Start Infinite Loop

Run continuous `while True` loop.

Wake Detection

- Call `listen_for_wake()`
- Remain passive until wake phrase detected

On Wake Trigger

- Play acknowledgment (“Han bataiye”)
- Switch to command capture mode

Capture Command

- Call `listen_loop(timeout)`
- Apply RMS filtering
- Validate tokens
- If invalid input → Return to wake stage

Deterministic Intent Check

- Exact match
- Fuzzy match

If found → Execute

ML Intent Fallback

- Convert to TF-IDF
- Predict using classifier
- If confidence > 0.65 → Execute

LLM Fallback

- Send text to `qwen_reply()`
- Generate response

Response Output

- Convert to speech (Piper)
- Play audio

Reset State

- Clear temporary variables
- Return to wake listening mode

Running the Voice Assistant System

1. Clone Repository

Shell
```shell
git clone <repository-url>
cd <project-folder>
```

2. Create Virtual Environment

Shell
```
python -m venv venv
```

Activate:

Windows:

Shell
```
venv\Scripts\activate
```

Linux / Mac:

Shell
```
source venv/bin/activate
```

3. Install Dependencies

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

4. Run the Assistant

Shell
```
python main.py
```

5. Using the System

- Start program
- Activate via wake word(Taiwan,hyva,kira,Nova) or manual trigger
- Speak command
- System processes and responds

6. Stop the Program

Press `CTRL + C`

Project Dependencies

Library	**Purpose**
Vosk	Offline speech recognition
Psutil	*System Info Handler* (CPU, RAM, battery, processes, uptime, etc.).
PyAudio / sounddevice	Microphone input
NumPy	Audio processing
Scikit-learn	Intent classification
Llama-cpp-python	LLM fallback
Piper TTS	Offline text-to-speech
Python ≥ 3.9	Runtime requirement

Linux system should have c++ compiler to build wheel required for the llm Llama to load and run locally

Here is the command to build it ,if it is not present by default

```
sudo apt update && sudo apt install -y build-essential cmake
```

The hardware requirements and specifications are:

- **Core Device:** Low-resource devices such as **Raspberry Pi** 4, 5, **Arduino uno q, Arduino RDK , other SBC WHICH RUNS LINUX.**

- **Processor (CPU):** Optimized for running the LLM (Llama-3.2-1B-Instruct) locally on the CPU using **4 threads**.

- **Input/Output:**
  - **Microphone** (for capturing user speech, using libraries like PyAudio/sounddevice). **USB mic or Bluetooth mic. for input.**
  - **Speaker/Audio Output** (for Text-to-Speech playback).

Limitations and Problems Faced

1. **CPU UTILIZATION IN IDLE STATE:**

   Cpu was getting utilized more in the idle state, so we fixed it with the wake word, if wake word detected then only the utilization starts.

2. **Speech Recognition Errors (Hindi)**
   Initially we faced errors in stt, we fixed it with proper fine tuning of the stt model, now it is working properly.

3. **Limited Intent Understanding**
   Intents help in fast running .So we used it ,but as of now there are limited intents, in future we are planning to add more intent libraries for fast response.

4. **Bluetooth Audio Configuration Challenges**
   Difficulty in managing separate Bluetooth input (mic) and output (speaker) devices. We faced this issue in the older version v3 , now we fixed the errors in the v4.

5. **Timeout & Partial Command Problems**
   Fixed listening windows sometimes cut off user commands.This was happening due to multiple python files accessing the same mic, so we fixed it with finetuning the pipeline flow.

Future Improvements / Vision

1. **Improved Wake Word Engine**
   Integrate a more robust and noise-resistant wake word system for higher accuracy.

2. **Advanced Speech Recognition**
   Upgrade to larger or optimized Hindi models to improve recognition quality.

3. **Smarter Intent Understanding**
   Replace rule-based matching with a lightweight ML/NLP

model for better contextual understanding.

4. **Optimized Edge Performance**
 Use model quantization and hardware acceleration to improve speed on Raspberry Pi.

5. **Better Audio Handling**
 Improve Bluetooth stability and implement dynamic audio device management.

6. **Scalable Knowledge System**
 Add a structured knowledge base with RAG (Retrieval-Augmented Generation) for dynamic information retrieval.

Offline Hindi Voice Assistant: Business Model and Market Analysis

Based on the project documentation, here is a proposed business model, followed by a Total Addressable Market (TAM), Serviceable Available Market (SAM), and Serviceable Obtainable Market (SOM) analysis.Business Model Canvas

Component	Description (Based on Project Features)
Value Proposition	**Privacy-Focused, Offline Voice AI:** Provides a fully functional, low-latency, and highly private Hindi voice assistant that runs entirely on low-resource, inexpensive hardware (like Raspberry Pi) without any cloud dependency. It offers reliable factual answers (Knowledge Base) and intelligent fallback (Local LLM: Llama-3.2-1B-Instruct).

Customer Segments	- **Individual Users in India/Hindi-speaking regions:** Seeking a personal, private voice assistant. - **OEMs/Hardware Manufacturers:** Looking to integrate an offline, regional language AI into smart home, educational, or IoT devices. - **Developers/Hobbyists:** Interested in open-source edge AI and regional language projects.
Channels	- **Direct-to-Consumer:** Sales via e-commerce (e.g., selling a pre-loaded Raspberry Pi kit). - **Business-to-Business (B2B):** Licensing the software/technology to hardware companies (OEMs). - **Open Source Platform:** GitHub/Community for the core software/development kit.
Customer Relationships	- **Automated Service:** Online documentation, community forums, and FAQs for developers/hobbyists. - **Dedicated B2B Support:** Consulting and customization for OEM partners.
Revenue Streams	- **Software Licensing (B2B):** Charging a per-unit license fee to hardware manufacturers for embedding the assistant. - **Premium Features/Add-ons:** Offering additional, specialized knowledge bases or enhanced LLM models for a fee. - **Hardware Kits (D2C):** Selling pre-configured, ready-to-run hardware kits (e.g., Raspberry Pi with pre-installed software and mic/speaker).

	- **Custom Development/Consulting:** Charging for professional services to integrate or customize the assistant for a specific client.
Key Activities	- Continuous model optimization (quantization, latency reduction). - Expanding the Hindi Knowledge Base and Intent Libraries. - Improving hardware compatibility (e.g., better Bluetooth audio handling). - Software maintenance and updates.
Key Resources	- Highly optimized Hindi NLP/STT/TTS models. - The structured Hindi Knowledge Base (Pickle-based storage). - The development team's expertise in edge AI and regional language processing.
Key Partnerships	- **Hardware Retailers/Distributors:** For selling D2C kits. - **Raspberry Pi/SBC Providers:** To ensure compatibility and optimal performance. - **Regional Language Data Providers:** To expand and refine the knowledge base.
Cost Structure	- Personnel costs (Developers, Researchers). - Cloud computing for model training and optimization. - Licensing fees for underlying technologies (if any). - Hardware production costs (for D2C kits).

-----Market Analysis: TAM, SAM, and SOM

This analysis is based on the project's focus on India and the Hindi-speaking market, emphasizing the offline, low-resource nature of the solution.

Metric	Definition	Application to Offline Hindi Voice Assistant
TAM (Total Addressable Market)	The entire market demand for the product or service.	**All households and individuals in India and other Hindi-speaking regions who use (or could use) a smart assistant.** This includes all potential users of smart home devices, IoT, and regional language technology.
SAM (Serviceable Available Market)	The segment of the TAM targeted by the product's specific value proposition, which the business can realistically reach.	**The market for *regional language, privacy-focused, and low-cost* smart assistants.** This targets: - Households seeking a voice assistant but concerned about data privacy (offline operation). - Consumers in price-sensitive markets (low-resource/low-cost hardware focus). - OEMs/device makers specifically requiring an edge AI solution for Hindi.

SOM (Serviceable Obtainable Market)	The portion of the SAM that the business can realistically capture in the short to medium term, given current resources and competition.	**The specific share of the SAM that can be captured in the first 1-2 years.** This focuses on: - Early adopters in the developer/hobbyist community (GitHub/DIY kits). - A limited number of initial OEM partnership deals with smaller, regional hardware manufacturers. - Initial sales of the pre-configured **Raspberry Pi** kits to privacy-conscious consumers.

Conclusion

The Offline Hindi Voice Assistant project successfully demonstrates a highly practical and innovative approach to regional language voice technology. By prioritizing **fully offline operation** and **privacy**, the system breaks the dependency on cloud services, making conversational AI accessible on low-resource, edge devices like the Raspberry Pi.

Key Achievements:

- **Robust Offline Pipeline:** The successful integration of wake word detection, Vosk-based Speech-to-Text (STT), a hybrid Intent Recognition System, a Knowledge Base, and a local **Llama-3.2-1B-Instruct** LLM for fallback, all operating in real-time.
- **Edge AI Deployment:** The project proves the feasibility of deploying a sophisticated Hindi voice assistant on affordable, low-power hardware, addressing common issues like high CPU utilization with a wake-word trigger mechanism.
- **Regional Focus:** The use of neural Hindi Text-to-Speech (Piper TTS) and a structured knowledge base in Hindi provides a reliable, high-quality user experience for the target language group.

Future Outlook:

The foundation is established for significant enhancements, including an **Improved Wake Word Engine**, **Advanced Speech Recognition** models, and a **Scalable Knowledge System** utilizing Retrieval-Augmented Generation (RAG) for dynamic information retrieval. This project serves as a compelling proof-of-concept for the future of localized, private, and efficient conversational AI.

References

- https://bhavya-tripathi.github.io/hindi-asr/ Porting Hindi ASR onto Raspberry Pi 3

- https://www.ijasi.org/index.php/ijasi/article/view/132 AI Assistant using LLM on Raspberry Pi

- https://arxiv.org/html/2504.16213v1 TinyML for Speech Recognition

- https://youtu.be/9GAmdWQPYAs Youtube demo video link.

- https://github.com/keerthiherer/Bharat-AI-in-SoC Git Repository link for the source code.