### **POC Documentation for Survey Bot**

**Document Prepared by:** Indumati

**Designation:** Data Engineer, Arca AI Technology

**Date:** 09 May 2025

### **Introduction**

**Objective**:

The objective of the Survey Bot PoC is to automate the process of conducting surveys, making it more efficient and scalable. The bot is designed to understand and process responses in multiple languages, including Malayalam, Tamil, Kannada, and English, while being capable of handling complex survey questions and providing structured responses.

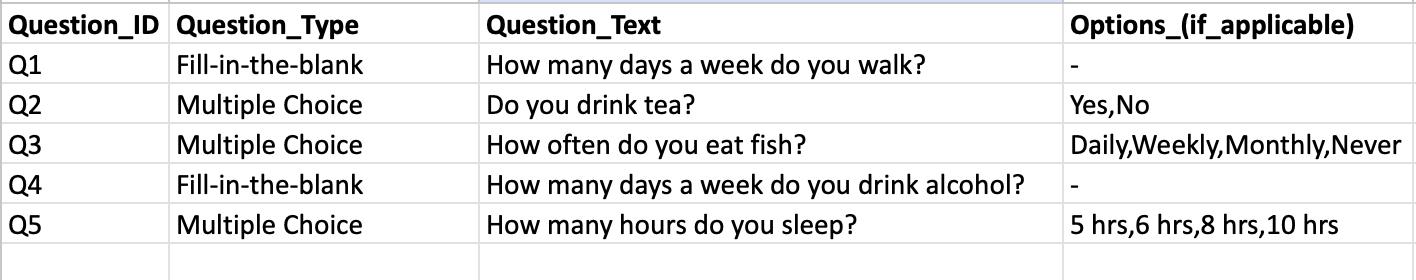
**Scope**:

This PoC focuses on the bot’s ability to:

* Handle **voice-based input**.
* Accurately **transcribe multilingual speech**.
* **Understand the context** of natural language responses.
* **Map responses** to structured survey formats.

The goal is to assess the feasibility of deploying a fully functioning survey system with multilingual capabilities and the ability to handle **real-time speech recognition**.

### **Phase 1:**

**Phase 1 questionnaire -**

**1. Initial Evaluation: Audio Capture & Speech Recognition**

#### **Requirement:**

The system needed to handle **long-form inputs** in **Kannada**, from:

* 🎤 **Live microphone audio**
* 📁 **Uploaded audio files**
* 📝 **Direct Kannada text input**

This was crucial to ensure that survey responses, which may not be short or crisp, are still captured effectively across multiple modes of input.

#### **Models Tried:**

1. **Whisper by OpenAI ❌**
   * **Strengths**:
     + Capable of transcribing long audio files with high accuracy.
     + Strong multilingual support including Kannada.
   * **Limitations**:
     + Slower inference speed for real-time use.
     + Heavier compute requirements.
     + Not optimal for capturing live audio in an interactive survey session.
2. **Python speech\_recognition library (using Google Web Speech API) ✅**
   * **Strengths**:
     + Fast and responsive for **live audio input**.
     + Easy to integrate with Flask-based systems.
     + Reasonably good transcription for Kannada with clear speech.

#### **Model used:**

#### We **chose speech\_recognition** for this PoC due to its **better performance in real-time use cases**. It worked well with **live Kannada audio**.

### **2. Translation Layer: Kannada Text to English Text**

#### **Requirement:**

#### After transcribing Kannada speech to text, the next step was to **translate** the **Kannada text into English** so that it could be processed by standard NLP models.

#### **Model used:**

We selected **IndicTrans2** as the final translation model for this PoC due to its **superior handling of Kannada-English translation**, especially for long responses and colloquial phrasing typical in spoken survey answers.

**IndicTrans2 (by AI4Bharat)** ✅

MODEL\_NAME : "ai4bharat/indictrans2-indic-en-1B"

* Strengths:
  + Improved version with better translation fidelity and support for more sentence structures.
  + Open-source, customizable, and ideal for integration into AI pipelines.
  + Worked well with real-world Kannada inputs and longer sentences.

## **3. Enhancing Readability: Automatic Punctuation Restoration**

### **Requirement:**

#### The **text derived from Kannada speech recognition** was generated as **unstructured, unpunctuated paragraphs**. This is a limitation of most ASR (Automatic Speech Recognition) systems, such as the speech\_recognition library with Google Web Speech API, which does not insert punctuation.

#### This caused **issues** for downstream **Question Answering (QA) models**, which rely on clear sentence boundaries to extract relevant responses.

#### **Model used:**

#### To address this, we used the **DeepMultilingualPunctuation** **library**, which provides automatic punctuation restoration.

#### Library name : deepmultilingualpunctuation

#### Class used: PunctuationModel

#### Functionality: The library restores punctuation in unpunctuated text, enabling better structure and readability for English transcriptions.

### **Impact:**

#### Added sentence boundaries for better comprehension by QA models.

#### Improved the accuracy of the answers extracted from the transcriptions.

#### Enhanced the quality and clarity of longer, conversational responses.

## **4. Accuracy Evaluation: Evaluating Transcription and Translation Quality**

### **Problem:**

#### To ensure the quality of transcriptions and translations, it was necessary to measure **accuracy** and **semantic alignment**. Word-level accuracy alone was insufficient, especially for longer and more conversational responses, where **contextual correctness** was paramount.

### **Models Tried:**

#### **JiWER (Word Error Rate) ❌**

#### **Strengths:**

#### Measured **word-level accuracy** between predicted and reference transcriptions.

#### Simple and effective for checking basic transcription errors.

#### **Limitations:**

#### Did not account for **semantic similarity** or **contextual correctness**.

#### Inadequate for long-form responses, where different phrasing could still convey the same meaning.

#### **SentenceTransformers (Semantic Similarity) ✅**

#### **Library Used:** sentence\_transformers

#### **Class Used:** SentenceTransformer

#### **Functionality:**

#### Utilized **transformer-based models** to measure **semantic similarity** between the generated text (transcription + translation) and the reference text.

#### It calculates a similarity score based on the meaning, regardless of exact word choices.

#### **Why This Was Chosen:**

#### Provided a **contextually accurate** measure of transcription and translation quality.

#### Allowed for better handling of longer, more complex answers by focusing on **meaning** rather than specific word matches.

### **Impact:**

#### Shifted from **word-level evaluation** to **semantic evaluation**, improving accuracy checks.

#### Ensured the generated text captured **intended meaning**, making the overall system more reliable.

#### Enhanced **evaluation** of longer, nuanced responses in the **survey bot**.

## **5. Question Answering: Extracting Answers from Transcribed and Translated Text**

### **Problem:**

After translating the Kannada speech input to English text, we needed a model that could extract **precise answers** to survey questions from long, often conversational paragraphs. This required a **reliable question answering (QA) model** capable of understanding context.

### **Models Tried:**

### **[Hugging Face] facebook/bart-large-cnn❌**

* Designed for summarization rather than extractive QA.
* Failed to consistently return accurate, pinpointed answers.

1. **[Hugging Face] google/flan-t5-base❌**

* General-purpose and good with diverse tasks.
* Performance was unstable for focused QA in a survey setting.

1. **[Hugging Face] t5-small❌**

* Lightweight and fast.
* Lacked depth and accuracy in extracting answers from longer text blocks.

1. **[Ollama] mistral❌**

* Strong open-ended generator.
* Overkill for structured QA and had higher inference time with inconsistent results.

1. **[Hugging Face] distilbert-base-cased-distilled-squad (Final Model Chosen) ✅**

* Fine-tuned on the SQuAD dataset for **extractive question answering**.
* Lightweight, accurate, and efficient in handling paragraph-level answers.
* Easily integrated into our real-time pipeline.

**(Note-** SQuAD (Stanford Question Answering Dataset): A benchmark dataset for training and evaluating machine learning models on question answering tasks. Models trained on SQuAD learn to extract **exact answer phrases** based on context — ideal for structured survey bots.)

### **Why this was chosen:**

* Delivered **precise answer spans** based on context.
* Balanced performance and resource efficiency for production deployment.
* Worked well with survey-style, translated English inputs.

**Impact:**

* Enabled the Survey Bot to **automatically extract answers** from transcribed responses.
* Played a critical role in **converting unstructured voice input into structured survey responses**.

**6. Initial Approach**

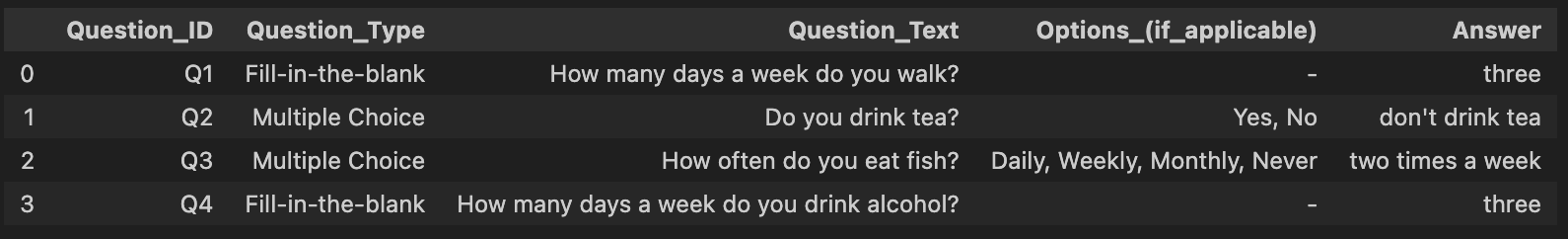
Initially, we ran the code directly to allow the model to pick answers.

**Example -**

**STT (speech to text) model result**-

“I walk **three days a week** i **don't drink** tea i drink alcohol **three times a week** and eat **fish two times a week**”

**QA(question answer) model result-**



The result was satisfactory, but since we needed the answers to be within the given options, we modified the flow of the code. We started by targeting the options first and then the question. The rationale behind this customization is explained below.

**7. Customizations Done:**

1. **Vocab JSON file/dictionary-**

Answer Validation with Vocabulary:

* Initially, the model was providing answers outside of the predefined options.
* To address this, a vocabulary/dictionary was added to limit the model's predictions to the given options.
* This ensured that the model’s answers were always within the acceptable set.

1. **Engineer intervention requirement:**

* We incorporated an **engineer intervention requirement** in cases where the model encountered new, unseen options (i.e., when the model was not trained with the new options).

(Note - If a new question is added but its option is already present in the **vocabulary**, no retraining of the model is required. However, if a new question is added with an option **not present in the vocabulary**, it will require **engineer intervention** to update the vocabulary and retrain the model.)

1. **Context Handling for Irrelevant Questions:**

* For questions like **"How many days a week do you walk?"**, if the transcription or conversation lacked context (e.g., no mention of "walking"), the model was generating irrelevant answers.
* Used **SpaCy** to detect key **keywords from the question**. The model then checked if those keywords (e.g., "walk") were **present in the conversation transcript**. If no relevant context was found, it returned **"No answer found"**.

1. **Text Normalization:**

* After the model predicted the answer but before processing it with the vocabulary, a feature was added to convert **textual numbers** (e.g., "two", "three") into their **numerical equivalents** (e.g., "2", "3").

**Phase 1 results -**

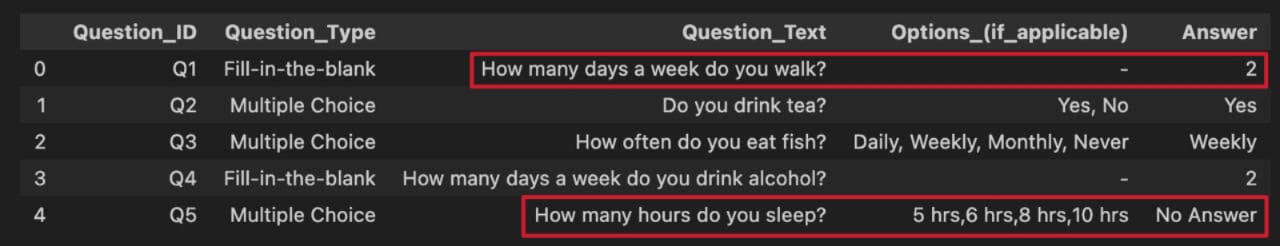
**STT (speech to text) model result**-

“I drink alcohol **two days a week**, I **drink tea** and eat fish **three times a week**, I sleep for five hours.” - This text has no contextual relevance to 'walking'

**QA(question answer) model result-**

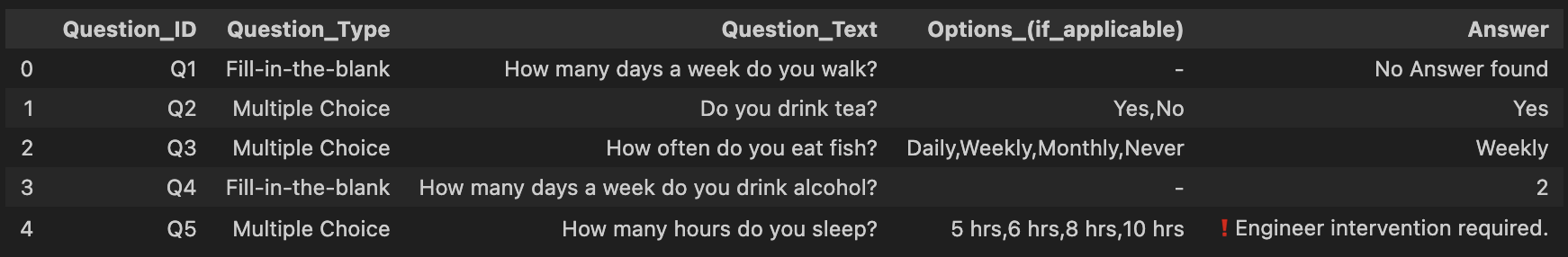
**Before Customization-**

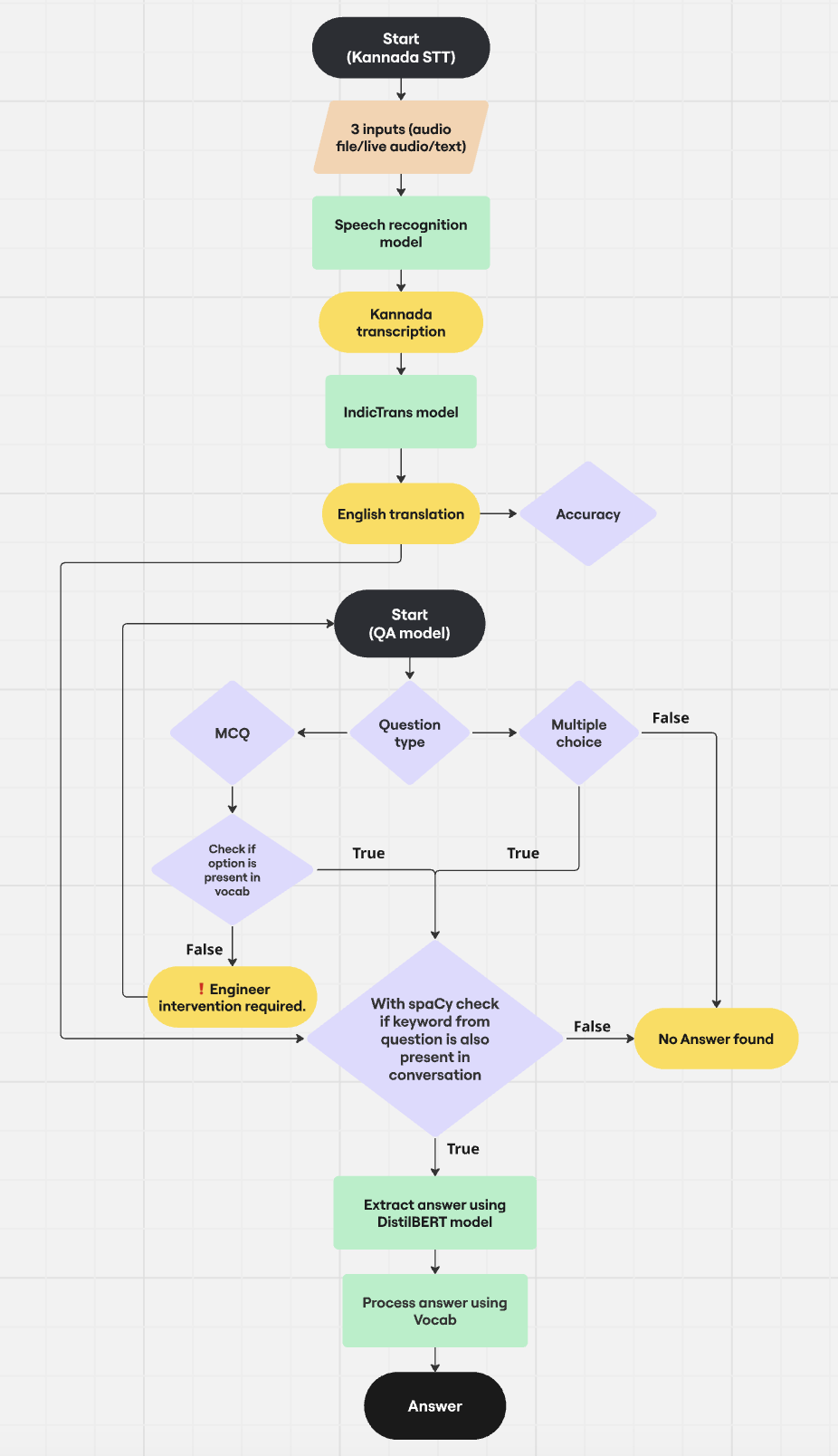
The **code** we used here **didn't include customization** to handle new questions with **new options,** and it also couldn't handle answers when the **context** of the question was **missing from the conversation.**

****

**After Customization-**

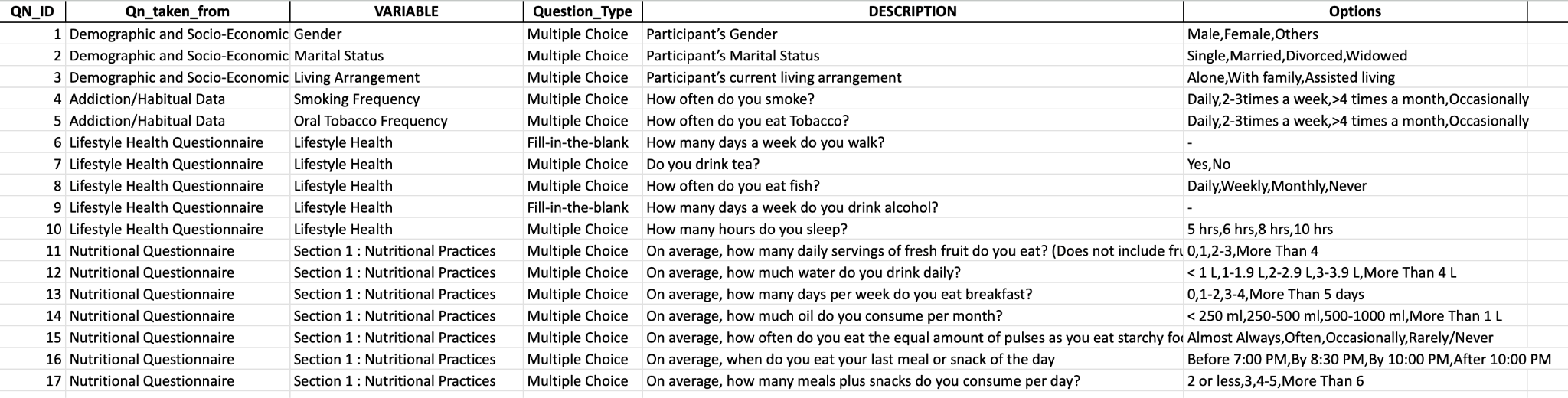
**After customization, the code was modified to handle new questions with new options by displaying 'Engineer intervention required'. Conditions were also added to handle cases where the context is missing.**



**Flowchart -**

**Phase 2 :**

**Phase 2 questionnaire -**

****

**Issues and Improvements Made:**

**– Modified the question format** to align with the standard format used in Confluence, **updated the code** accordingly, and **added more questions to the questionnaire** for testing.

**–** Encountered several challenges when introducing **new questions with previously unseen options,** as the model was not pre-trained to handle them. To ensure the QA model produces 100% accurate answers, **custom handling** was implemented for each option.

**Examples***:*

* **Oil Consumption:**For the question *"How much oil do you consume daily?"*, the model predicted the answer as **"200 ml"**. However, our predefined options were:  
  ['< 250 ml', '250-500 ml', '500-1000 ml', 'More Than 1 L'].  
  By implementing vocabulary mapping, the system correctly matched **"200 ml"** to **"< 250 ml"**.
* **Meal Timing:**For the question *"When do you eat your last meal or snack of the day?"*, the predicted answer was **"around seven in the evening"**.  
  The available options were:  
  ['Before 7:00 PM', 'By 8:30 PM', 'By 10:00 PM', 'After 10:00 PM'].  
  The model successfully matched the response to **"Before 7:00 PM"**.

Achieving this level of precision required careful vocabulary design, manual mapping, and attention to fine-grained contextual cues.

**Phase 2 results -**

**Kannada audio input content-**

“ನಾನು ವಿಚ್ಛೇದಿತ ಮಹಿಳೆ, ಪ್ರಸ್ತುತ ನನ್ನ ಕುಟುಂಬದೊಂದಿಗೆ ವಾಸಿಸುತ್ತಿದ್ದೇನೆ. ನಾನು ದಿನಕ್ಕೆ ಮೂರು ಊಟಗಳನ್ನು ತಿನ್ನುತ್ತೇನೆ. ನನ್ನ ಕೊನೆಯ ಊಟ ಸಾಮಾನ್ಯವಾಗಿ ಸಂಜೆ 7 ಗಂಟೆಯ ಸುಮಾರಿಗೆ. ಕೆಲವೊಮ್ಮೆ, ನಾನು ಒಂದೇ ಊಟದಲ್ಲಿ ಸಮಾನ ಪ್ರಮಾಣದ ದ್ವಿದಳ ಧಾನ್ಯಗಳು ಮತ್ತು ಪಿಷ್ಟಯುಕ್ತ ಆಹಾರವನ್ನು ಸೇವಿಸುತ್ತೇನೆ. ನಾನು ತಿಂಗಳಿಗೆ ಸುಮಾರು 200 ಮಿಲಿ ಎಣ್ಣೆಯನ್ನು ಬಳಸುತ್ತೇನೆ. ನಾನು ಹಣ್ಣುಗಳನ್ನು ತಿನ್ನುವುದಿಲ್ಲ. ನಾನು ವಾರದಲ್ಲಿ ಆರು ದಿನ ಉಪಾಹಾರ ಸೇವಿಸುತ್ತೇನೆ. ನಾನು ವಾರಕ್ಕೆ ಎರಡು ಬಾರಿ ಮದ್ಯಪಾನ ಮಾಡುತ್ತೇನೆ. ನಾನು ಪ್ರತಿದಿನ ಎರಡು ಲೀಟರ್ ನೀರು ಕುಡಿಯುತ್ತೇನೆ. ನಾನು ವಾರಕ್ಕೆ ಐದು ಬಾರಿ ಧೂಮಪಾನ ಮಾಡುತ್ತೇನೆ. ನಾನು ಪ್ರತಿದಿನ ತಂಬಾಕು ಸೇವಿಸುತ್ತೇನೆ. ನಾನು ಚಹಾ ಕುಡಿಯುತ್ತೇನೆ ಮತ್ತು ವಾರಕ್ಕೆ ಮೂರು ಬಾರಿ ಮೀನು ತಿನ್ನುತ್ತೇನೆ. ನಾನು ಐದು ಗಂಟೆಗಳ ಕಾಲ ನಿದ್ರೆ ಮಾಡುತ್ತೇನೆ.”

**STT (speech to text) model result** -

“I am a **divorced woman** currently **living with my family**. I eat **three meals** a day. My last meal is usually around **seven in the evening**. **Sometimes** I eat the same amount of pulses and fats in one meal. I use about **200ml** of oil a month. I **don't eat fruits**. I eat breakfast **six days a week.** I drink **two litres** of water every day. I smoke **five times** a week. I smoke tobacco **every day**. I drink tea and eat **fish three times a week.** I sleep for five hours.”

**QA(question answer) model result -**

****

**Technologies & Libraries Used:**

### **Libraries Used in Survey Bot PoC -**

#### **Machine Learning / NLP Libraries:**

* transformers – For loading and using Hugging Face models (translation, QA, etc.)
* sentence\_transformers – For semantic similarity and accuracy evaluation
* deepmultilingualpunctuation – For restoring punctuation in transcribed text
* spacy – For keyword detection in questions and conversation
* IndicTransToolkit – For preprocessing and handling Indian languages
* torch – Backend framework for model execution and tensor ops

#### **Speech Processing:**

* speech\_recognition – For live microphone-based transcription (Google Web Speech API)
* pydub – For audio format conversion and audio manipulation
* scipy.io.wavfile – For writing WAV audio files

#### 

#### **Utilities / System:**

* tempfile, os, sys, threading, time, logging, json, re, datetime – For file handling, system operations, threading, and logging
* pandas – For data handling and option mapping in structured survey responses
* numpy – Used indirectly via scipy, for numerical operations

**Project Dependency Versions :**

Python== 3.10.13

pandas==2.2.3

numpy==2.2.4

transformers==4.49.0

sentence-transformers==3.4.1

spacy==3.8.4

torch==2.6.0

speechrecognition==3.14.1

scipy==1.15.2

### **Conclusion**

The Survey Bot PoC successfully validates a multilingual, voice-based survey system. It combines speech recognition, translation, punctuation restoration, and question answering to convert unstructured speech into structured survey responses. **Customizations like vocabulary mapping and context checks ensured high accuracy.** This lays a strong foundation for a scalable, real-time AI survey assistant.