**Real Time AI Sales Call Assistant – Agile Documentation**

**Table of Contents**

1. Introduction
2. Project Overview
3. Objectives
4. Agile Framework
   * Development Approach
   * Sprint Planning
5. System Architecture
6. User Stories & Use Cases
7. API & Integration Documentation
8. Code Documentation
9. Deployment & Maintenance
10. Continuous Improvement & Feedback

**1. Introduction**

This document serves as a living, lightweight guide for the Sales Call Assistant project. It is intended to be updated iteratively throughout development to ensure it remains current and useful for both developers and stakeholders.

**2. Project Overview**

The Real Time AI Sales Call Assistant is designed to empower sales teams by providing real-time insights and recommendations during sales calls. It leverages AI to improve call outcomes through features like transcription, sentiment analysis, and actionable suggestions.

**3. Objectives**

* Automate customer call analysis.
* Provide sentiment trends and future predictions.
* Offer personalized product recommendations.
* Improve objection handling using AI-driven responses.
* Enable data-driven decision-making through actionable insights.
* Implement real-time speech recognition.

**4. Agile Framework**

**Development Approach**

The project follows an Agile methodology with milestone-based sprints, emphasizing incremental progress and continuous improvement.

**Sprint Planning**

* **Sprint 1:** Environment Setup & Basic Transcription

Set up the environment for real-time speech analysis

Implement initial speech-to-text processing using Groq-whisper for real-time transcription.

Train and familiarize with LLMs for sentiment analysis.

Begin collecting initial data from mock sales calls.

Establish initial benchmarks for data quality and processing speed.

Develop a Python-based speech recognition system using Groq-Whisper and Sounddevice to process live audio input.

* **Sprint 2:** Sentiment Analysis of text and voice

Implement python libraries NLTK and Pydub for sentiment analysis.

Develop algorithms that detect sentiment changes based on tone, language, and context.

Optimize speech processing latency to enable real-time analysis.

Conduct multiple iterations to refine sentiment detection accuracy.

Develop visual dashboards to display real-time sentiment shifts during sales calls.

Integrate Groq-whisper transcription output with sentiment analysis modules for real-time interpretation.

* **Sprint 3:** Advanced Recommendation System & CRM Integration

Integrate data to recommend products based on customer profiles.

Implement a dynamic question prompt generator that suggests questions or objection-handling techniques based on conversation flow.

Design and deploy a recommendation engine that adapts based on conversation tone and user history.

Validate recommendations through iterative testing and feedback loops.

Improve objection-handling logic by analyzing real-world sales interactions.

Optimize integration between Groq-whisper recognition and objection-handling modules.

* **Sprint 4:** Deployment, Testing, and Feedback Integration

Deploy the post-call summary generation module.

Provide automated insights for sales training and improvement.

Implement a sentiment-based summary report for sales calls.

Develop an AI-driven insights module that highlights conversation patterns.

Test with real sales call data and iterate based on sales team feedback.

Fine-tune speech-to-text accuracy to enhance post-call analytics and summaries.

**5. System Architecture**

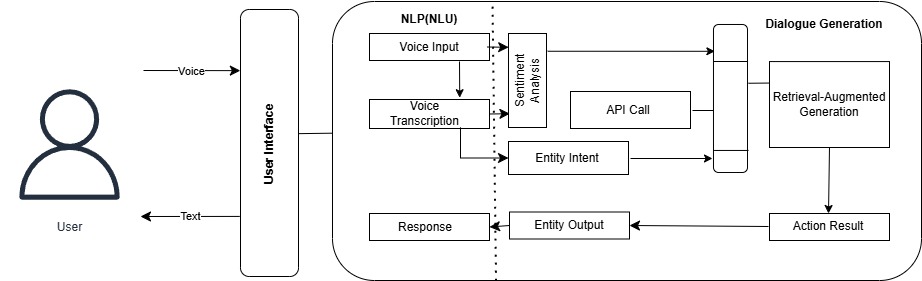
**High-Level Workflow:**

1. **Audio Input:** Captures real-time audio during calls.
2. **Transcription Module:** Converts audio to text using Groq-Whisper.
3. **Analysis Engine:** Processes text and voice for sentiment analysis (NLTK, PyDub).
4. **Recommendation Engine and objection handling:** Provides real-time suggestions.
5. **User Interface:** Displays insights via Streamlit.

**Technologies Used:**

* **Frontend:** Streamlit
* **Backend:** Python (Sounddevice, FAISS, Groq API)
* **Data Processing:** NLTK, spaCy
* **Integrations:** CRM systems via APIs

**Diagram:**



**6. User Stories & Use Cases**

* **Real-Time Insight:** As a sales rep, I want real-time transcription and sentiment feedback to adjust my approach during calls.
* **Post-Call Analysis:** As a manager, I want summaries to coach my team effectively.
* **CRM Sync:** As an admin, I need automated CRM data processing.

**7. API & Integration Documentation**

**API Endpoints:**

* **/transcribe:** POST audio data, return text.
* **/analyze:** POST text, return sentiment & keywords.

**Data Models:**

* **CallRecord:** timestamp, transcription, sentiment.
* **User:** Intented product.

**8. Code Documentation**

**Folder Structure:**

* app.py: Streamlit interface
* audio\_processing.py: Handles audio input and audio-to-text
* sentiment\_analyzing.py: Processes sentiment
* crmd\_system.py: Provides suggestions
* storing\_conversations.py: store user history.

**Inline Documentation:**

def record\_audio(audio\_file):

"""Record audio of the user."""

# Audio recording logic here

def transcribe\_audio(audio\_file):

"""Converts audio file to text using Groq-Whisper."""

# Transcription logic here

def sentiment\_analisis(text, audio\_file):

"""Analyze sentiment of text and audio."""

# Sentiment analysis logic here

def workflow():

"""Recommend products and objection handling."""

# workflow logic here

def summary():

"""Provides a summary based on all recommendations."""

# summary logic here

def store\_response():

"""Stores responses for each query."""

# storing response logic here

**9. Deployment & Maintenance**

**Local Deployment:**

bash ```

git clone https://github.com/Divyanshu671/Sales\_Call\_Assistant.git

cd Sales\_Call\_Assistant

python -m venv venv

venv\Scripts\activate

pip install -r requirements.txt

streamlit run app.py

```

**Streamlit Cloud Deployment:**

* Push code to GitHub
* Connect to Streamlit Cloud
* Deploy & Monitor

**10. Continuous Improvement & Feedback**

* Regular code reviews and Agile retrospectives.
* User feedback integration after each sprint.
* Continuous integration & deployment (CI/CD) pipelines.

This Agile documentation is designed to evolve along with the project. Regular updates will ensure its relevance as the project grows.