Agile Documentation for Real-Time AI Sales Call Assistant

by Zahra Shaikh (https://github.com/zia9571/AI-Sales-Call-Assistant-)

1. Project Overview

1.1 Project Name:

Real-Time AI Sales Call Assistant for Enhanced Conversation

1.2 Description:

The Real-Time AI Sales Call Assistant enhances sales conversations by providing real-time analysis, sentiment detection, product recommendations, and objection handling. It leverages speech recognition, sentiment analysis, and machine learning to transcribe and analyze live sales calls, providing immediate feedback and actionable insights for sales representatives.

1.3 Goals & Objectives:

- Improve the efficiency and effectiveness of sales representatives.
- Provide real-time feedback and actionable insights during sales calls.
- Enhance customer experience through personalized interactions.
- Enable sales managers to monitor and analyze sales performance.

1.4 Target Users:

- Sales Representatives → Get real-time feedback, product recommendations, and objection handling.
- Sales Managers → Monitor performance and analyze sales interactions via call summaries and sentiment analysis.

1.5 Tech Stack:

Component	Technology Used
Languages	Python
Frameworks	Streamlit

APIs	Google Sheets API, Hugging Face API
Libraries	SpeechRecognition, Vosk, Sentence Transformers, Faiss, Transformers, Pandas, Plotly, PyAudio
Databases	Google Sheets
Other Tools	dotenv (for environment variables)

---

2. Backlog (User Stories & Tasks)

2.1 User Stories

2.1.1 Real-Time Sales Call Analysis

As a Sales Representative, I want to analyze sales calls in real-time, so that I can get immediate feedback and improve my sales techniques.

2.1.2 Dashboard for Call Summaries and Sentiment Analysis

As a Sales Manager, I want to view summaries and sentiment analysis of all sales calls, so that I can monitor the performance and effectiveness of my sales team.

2.2 Completed Tasks

2.2.1 Real-Time Sales Call Analysis

- Task 1.1: Implemented real_time_analysis() in app.py.
- Task 1.2: Integrated speech recognition and sentiment analysis.
- Task 1.3: Provided real-time objection handling and product recommendations.
- Task 1.4: Summarized and stored conversation data in Google Sheets.

2.2.2 Dashboard for Call Summaries and Sentiment Analysis

- Task 2.1: Implemented run_app() in app.py to set up the Streamlit app.
- Task 2.2: Fetched call data from Google Sheets and displayed summaries.
- Task 2.3: Provided visual representations of sentiment analysis.
- Task 2.4: Allowed detailed view of specific call data.

---

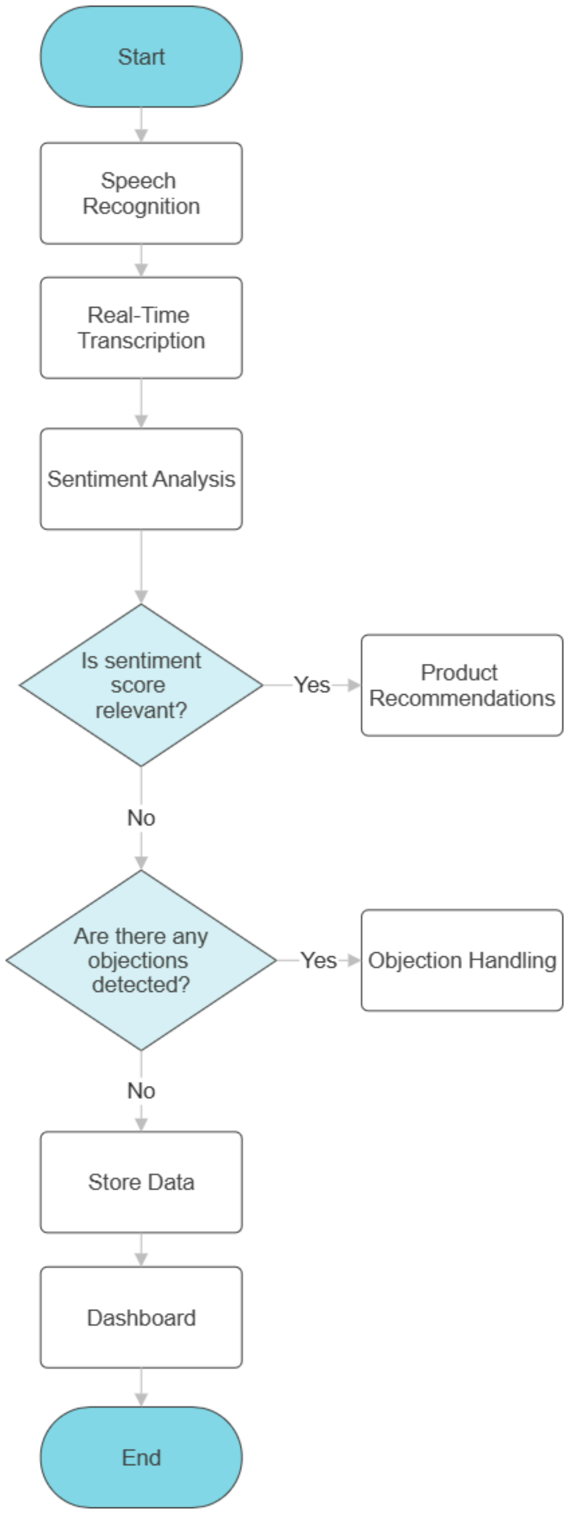
3. Architecture & Design Notes

3.1 High-Level Architecture

- Speech Recognition → Uses Vosk to transcribe live audio into text.

- Sentiment Analysis → Utilizes Hugging Face API with a multilingual sentiment model.
- Product Recommendations → Uses Sentence Transformers and Faiss for real-time suggestions.
- Objection Handling → Provides automated responses using trained embeddings.
- Data Storage → Stores and retrieves conversation data from Google Sheets.
- Dashboard → Built with Streamlit to display summaries and sentiment insights.

Here is a flowchart depicting the architecture (*Figure 1*)-



3.2 Database Schema (Google Sheets)

Column Name	Description
Call ID	Unique identifier for each sales call
Chunk	A segment of the conversation
Sentiment	Sentiment score for each chunk (0-1)
Summary	Generated summary of the call
Overall Sentiment	Overall sentiment classification (Positive, Neutral, Negative)

---

4. Development Process (Iterative Updates)

4.1 Sprint Logs

4.1.1 Sprint 1:

- Goal: Set up real-time transcription and sentiment analysis.
- Completed: Implemented real_time_analysis(), integrated Vosk, and sentiment analysis.
- Issue: Initial latency problems.
- Resolution: Optimized buffering and reduced processing time.

4.1.2 Sprint 2:

- Add real-time objection handling and product recommendations.
- Integrated objection handling and product recommender.
- Issue: The RAG was generating irrelevant suggestions.
- Resolution: Fine-tuned model thresholds for better accuracy.

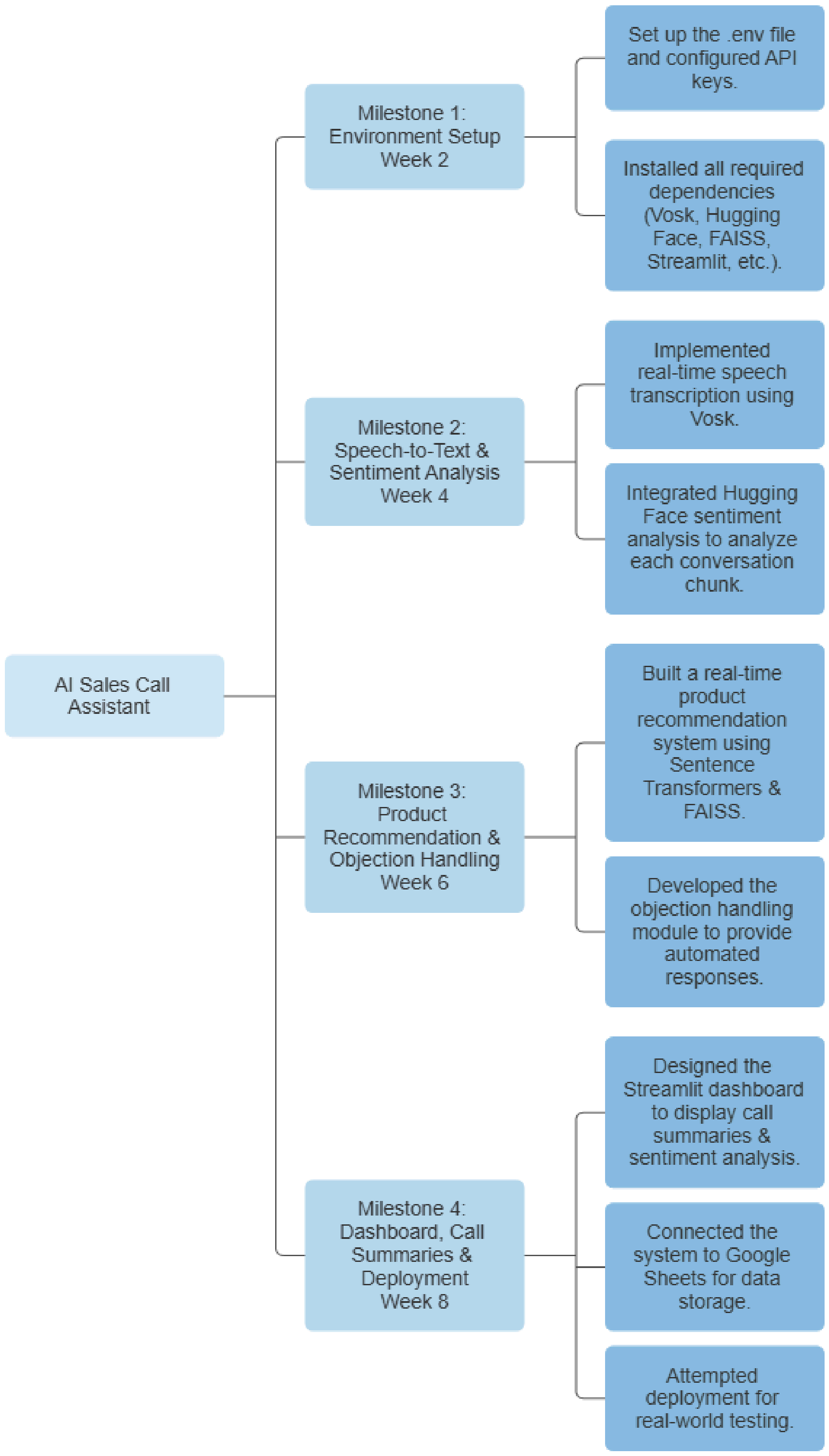

AI Sales Call Assistant
Milestone 1: Environment Setup Week 2
Set up the .env file and configured API keys.
Installed all required dependencies (Vosk, Hugging Face, FAISS, Streamlit, etc.).
Milestone 2: Speech-to-Text & Sentiment Analysis Week 4
Implemented real-time speech transcription using Vosk.
Integrated Hugging Face sentiment analysis to analyze each conversation chunk.
Milestone 3: Product Recommendation & Objection Handling Week 6
Built a real-time product recommendation system using Sentence Transformers & FAISS.
Developed the objection handling module to provide automated responses.
Milestone 4: Dashboard, Call Summaries & Deployment Week 8
Designed the Streamlit dashboard to display call summaries & sentiment analysis.
Connected the system to Google Sheets for data storage.
Attempted deployment for real-world testing.

Figure 2. Sprint Map

4.2 Change Log

Version	Updates
v0.1	Initial setup, basic transcription.
v0.2	Added sentiment analysis.
v0.3	Implemented objection handling.
v0.4	Added product recommendations.
v1.0	Finalized real-time analysis and dashboard.

---

5. Code Documentation

5.1 Setup Instructions

```
# Clone the repository
git clone https://github.com/zia9571/AI-Sales-Call-Assistant.git
cd AI-Sales-Call-Assistant

# Create a virtual environment
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt
```

5.2 Environment Variables (.env File)

```
vosk_model_path=path/to/vosk/model
huggingface_api_key=your_huggingface_api_key
google_creds=path/to/your/google/credentials.json
google_sheet_id=your_google_sheet_id
```

5.3 Running the Project

```
streamlit run app.py
```

---

6. Testing & Deployment Notes

6.1 Test Cases

6.1.1 Manual Testing:

- Test accuracy of speech recognition in noisy environments.
- Verify real-time product recommendations are relevant.

6.1.2 Automated Tests (Future Plan):

- Write unit tests for real_time_analysis().

6.2 Deployment Instructions

6.2.1 Local Deployment:

python main.py

6.2.2 Streamlit Cloud Deployment:

streamlit run app.py

---

7. Final Summary

7.1 What Went Well:

- Successful integration of multiple AI models & APIs.
- Positive feedback from initial users.
- Efficient data storage and retrieval using Google Sheets.

7.2 Challenges Faced:

- Handling noisy environments for speech recognition.
- Fine-tuning AI models for better accuracy.
- Managing dependencies and environment setup.
- Deploying on Hugging Face while supplying a real-time microphone input.

7.3 Next Steps (Future Enhancements)

- Improve noise handling in speech recognition.

- Fine-tune sentiment analysis model.

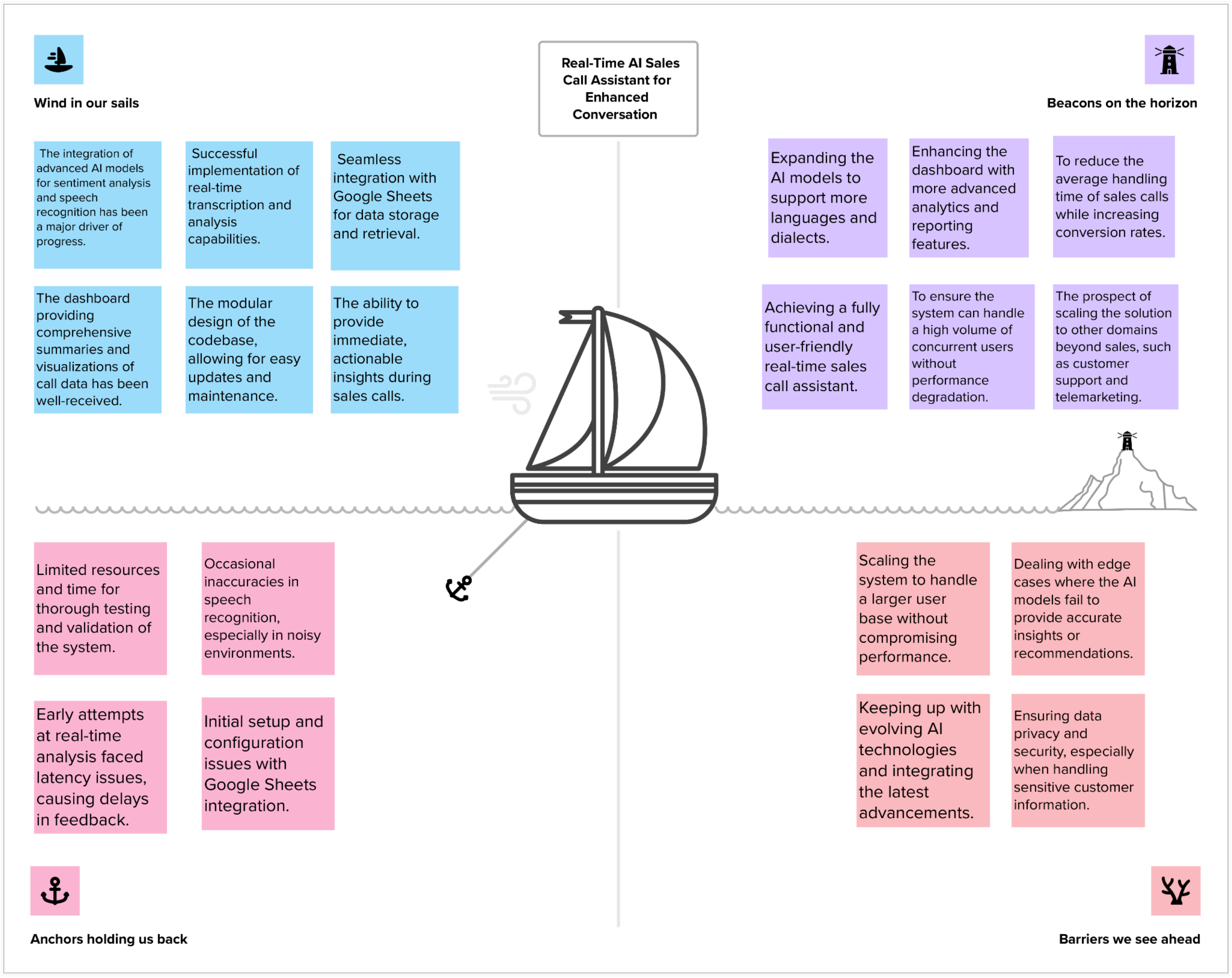

Real-Time AI Sales Call Assistant for Enhanced Conversation
Wind in our sails
The integration of advanced AI models for sentiment analysis and speech recognition has been a major driver of progress.
Successful implementation of real-time transcription and analysis capabilities.
Seamless integration with Google Sheets for data storage and retrieval.
The dashboard providing comprehensive summaries and visualizations of call data has been well-received.
The modular design of the codebase, allowing for easy updates and maintenance.
The ability to provide immediate, actionable insights during sales calls.
Beacons on the horizon
Expanding the AI models to support more languages and dialects.
Enhancing the dashboard with more advanced analytics and reporting features.
To reduce the average handling time of sales calls while increasing conversion rates.
Achieving a fully functional and user-friendly real-time sales call assistant.
To ensure the system can handle a high volume of concurrent users without performance degradation.
The prospect of scaling the solution to other domains beyond sales, such as customer support and telemarketing.
Limited resources and time for thorough testing and validation of the system.
Occasional inaccuracies in speech recognition, especially in noisy environments.
Early attempts at real-time analysis faced latency issues, causing delays in feedback.
Initial setup and configuration issues with Google Sheets integration.
Anchors holding us back
Scaling the system to handle a larger user base without compromising performance.
Dealing with edge cases where the AI models fail to provide accurate insights or recommendations.
Keeping up with evolving AI technologies and integrating the latest advancements.
Ensuring data privacy and security, especially when handling sensitive customer information.
Barriers we see ahead

Figure 3. Sailboat Retrospective