# **Murdock: Technical Writeup**

## **Overview**

Murdock is an offline-first, AI-powered multimodal assistant designed to empower blind and visually impaired individuals with real-time scene understanding, text reading, and translation capabilities. The app runs entirely on-device, leveraging Google’s Gemma 3n model via TFLite and MediaPipe, ensuring privacy and speed without relying on internet access.

This writeup details the engineering decisions, model integration pipeline, and architecture that underlie the app, as well as key challenges and innovations implemented during development.

## **Project Highlights**

* **Offline Multimodal AI**: Murdock uses the Gemma 3n model in.task format for on-device inference — no internet needed.
* **Voice-First Interaction**: Supports full audio input via Android SpeechRecognizer and audio output via TextToSpeech.
* **Scene and Text Understanding**: Captures real-time camera input and runs OCR + LLM vision prompting for scene description and sign reading.
* **Multilingual Support**: Users can choose from multiple output languages, including Spanish, Chinese, Japanese, and more.
* **LangGraph-style Routing**: Dynamically decides whether visual input is needed before running camera or direct LLM response.
* **RAG-Style Context Memory**: Maintains short-term memory across prompts for conversational continuity — fully local, no server.
* **Modes: Q&A and Translation**: Offers hands-free question answering or in-place text translation based on user commands.
* **Optimized for 4GB+ Phones**: Runs on real Android devices using MediaPipe LLM APIs with quantized Gemma 3n models.

## **Architecture (see Figure 1 below)**

### **Platform**

* **Language**: Kotlin (Android Native)
* **Model Runtime**: [Google AI Edge SDK (MediaPipe Tasks)](https://developers.google.com/mediapipe)
* **Device Compatibility**: ARM64 devices with NNAPI/CPU acceleration (4GB RAM+)

### **Core Modules**

* **Gemma 3n TFLite model** (.task format via MediaPipe LlmInference)
* **CameraX** for real-time image capture
* **MLKit OCR** for reading text from images
* **Android SpeechRecognizer** for real-time voice input
* **Android TextToSpeech** (TTS) for immediate voice feedback

### **Multimodal Input Routing (LangGraph-style logic)**

* Voice/Text Input → Decision Prompt (LLM): \*“Is vision input needed?”
* If yes → Trigger camera capture and OCR
* If no → Text-only streaming LLM response

A diagram of a network

AI-generated content may be incorrect.

Figure 1. Murdok Multimodal Routing Architecture

## **How Gemma 3n Was Used**

Gemma 3n (E2B-it) was used in the following capacities:

### **Text-to-Text Generation**

Used for:

* Responding to natural language queries (Q&A mode)
* Translating OCR-extracted text (Translate mode)
* Providing summaries and instructions in user-selected language

Implementation:

* Streaming token generation via LlmInferenceSession.generateResponseAsync()
* Language prefix injection (e.g. “Please answer the following in Korean”)
* Local RAG-style memory buffer of last N Q&A pairs (for context persistence)
* Mode-based prompt handling: Translation prompts explicitly tagged, while Q&A mode uses user intent classification via LLM

### **4.2.** **Vision + Prompt**

Used for:

* Scene descriptions
* Reading physical signs (e.g. menus, street signs, screens)

Implementation:

* Camera bitmap captured via PreviewView.getBitmap()
* Image fed into LLM session as MPImage
* Joint prompt+image used for multimodal inference

### **4.3.** **ASR-to-LLM and Voice Output**

Used for:

* Hands-free interaction with the app
* Voice-based question answering and scene description

Implementation:

* Audio input captured via Android’s SpeechRecognizer
* Transcribed prompt sent directly to the LLM decision node
* Output streamed and spoken aloud using TextToSpeech, sentence-by-sentence
* Language and voice controlled to support different locales and (optionally) male voice

This ensured fully accessible voice-first usage, including use cases such as:

* “What do you see?” (Q&A mode)
* “Read the sign” (Vision + Prompt)
* “Translate this into Spanish” (Translate mode)

### **4.4.** **Retrieval-Augmented Generation (RAG)**

To provide lightweight conversational memory within limited prompt context size, we implemented a simplified local RAG system. This stored the last 3–5 Q&A pairs in a queue and prepended them to every LLM prompt. This helped maintain context across back-and-forth interactions without needing cloud RAG pipelines.

Implementation:

* In-memory linked list (LinkedList<Pair<String, String>>) as local context cache
* Auto-cleared on vision prompts to avoid token overflow
* Automatically resets if prompt exceeds threshold length

1. **App Features and Instructions**

* **RUN Text** button: Sends text input to the AI
* **Talk** button: Captures voice input using speech recognition
* **Describe** button: Captures image from the camera and sends it to the LLM
* **Switch** button: Toggles front and back cameras
* **Mode Selector (top-right corner)**: Toggle between **Q&A Mode** and **Translation Mode**
* **Language Dropdown**: Choose the response language (English, Chinese, Japanese, Korean, French, German)

### **5.1. Q&A Mode**

* Tap **RUN Text** or **Talk** to input your question.
* The LLM decides whether vision input is needed:
  + If yes, it will automatically capture an image and use it to answer.
  + If no, it responds using text-only.
* You can also manually tap **Describe** to get a scene description.
* Regardless of the language you input, Murdock replies in your selected language via audio output.

### **5.2. Translation Mode**

* Tap **RUN Text** or **Talk** to input a sentence.
  + Murdock will translate it into the selected language.
* Tap **Describe** to capture an image and extract text using OCR.
  + The extracted text will be translated and spoken aloud in your selected language.

## **Challenge**

### **6.1. Slow Vision Inference**

* Initial image + prompt inference took 15–30 seconds or more

Solution:

* Disabled full context memory when generating from image+prompt (no RAG)
* Downscaled image prior to creating MPImage
* Avoided streaming response delays by speaking sentence-by-sentence

### **6.2. Coordinating Multimodal Routing**

* Managing when to trigger vision, when to use text-only, and how to translate responses required careful state logic.

Solution:

* Implemented a LangGraph-style input router using LLM decision prompts, train LLM to make decisions when to trigger vision
* Automatically selected camera capture, OCR, or direct LLM prompt
* Synced image, OCR, and TTS pipelines to avoid collisions or state conflicts

### **6**.**3. Multilingual Support**

* Needed to support visually impaired users in multiple languages (e.g., Chinese, Spanish)

Solution:

* Injected language-specific prefix into every LLM prompt
* Modified system prompt to enforce format.
* Used Android TTS engine with selected locale and fallback support

### **6.4. Hallucinations and Abstract Language**

* Gemma 3n occasionally produced poetic or vague descriptions like: *“The scene appears to be an abstract arrangement…”*

Solution:

* Hardcoded a system prompt: > You are Murdock, a multimodal AI assistant for blind users. Describe scenes clearly and simply. Avoid abstract, poetic, or artistic language. List visible objects, people, and text. Do not ask questions.

This drastically improved output consistency.

## **Why These Technical Choices Were Right**

* **Gemma 3n TFLite**: Enabled offline, privacy-first inference on real Android devices without cloud calls
* **MediaPipe**: Allowed flexible integration of LLM, vision, and audio pipelines via task-based APIs
* **Custom LangGraph-style Routing**: Ensured multimodal coordination (camera, voice, text) felt smooth and responsive
* **Streaming + Speech Output**: Designed for immediate feedback, especially for blind users
* **Local RAG memory**: Gave the app lightweight conversational continuity without any server or external retrieval

## **Conclusion**

SightSpeak (Murdock) is a deeply engineered multimodal app that brings the power of Gemma 3n to a real-world assistive use case. By combining offline LLM inference, dynamic camera capture, OCR, speech recognition, and accessibility-first feedback design, the app delivers real value in daily life.

Every technical decision — from session resets to memory-safe model loading — was made to support reliability, speed, and usability for the visually impaired.

Gemma 3n proves its impact here not as a chatbot, but as a practical tool for independence and inclusion.

1. **Future Work**

While Murdock already provides a compelling on-device multimodal assistant, several areas remain open for future enhancement:

* **Model Fine-Tuning**: Custom fine-tuning of Gemma 3n on assistive-specific datasets could reduce hallucinations further and improve clarity of scene description.
* **Expanded Language Support**: Support for additional languages and regional dialects will enhance global accessibility.
* **Braille Integration**: Adding tactile output through Braille displays could increase accessibility for users who are deafblind.
* **Wearable Device Support**: Porting the app to AR glasses or wearable devices would allow continuous, hands-free assistance.
* **Memory Optimization**: Further memory profiling could allow the app to run on devices with less than 4GB RAM, broadening user reach.

1. **References**
2. **Gemma 3n - Google DeepMind**  
   https://ai.google.dev/gemma  
   *Gemma 3n is a lightweight multimodal LLM designed by Google, supporting on-device inference with privacy-first design.*
3. **MediaPipe LLM Inference API - Google AI Edge SDK**  
   <https://developers.google.com/mediapipe>  
   *Used for loading and executing .task models such as Gemma 3n on Android with multimodal support (vision + text).*
4. **ML Kit for OCR (Text Recognition)**  
   https://developers.google.com/ml-kit/vision/text-recognition  
   *Used for extracting scene text from camera input to feed into translation or prompt generation.*
5. **Android TextToSpeech (TTS)**  
   https://developer.android.com/reference/android/speech/tts/TextToSpeech  
   *Used for offline voice output of LLM responses with localized language and gender control.*
6. **Android SpeechRecognizer (ASR)**  
   https://developer.android.com/reference/android/speech/SpeechRecognizer  
   *Used to capture voice prompts and convert them to text for Gemma 3n input.*
7. **LangGraph (Inspiration)**  
   <https://github.com/langchain-ai/langgraph>  
   *Inspired the routing logic for multimodal state transitions between text, vision, and translation tasks.*
8. **Kotlin Android SDK**  
   https://developer.android.com/kotlin  
   *Used to implement the entire Android app natively with coroutine support and UI/UX optimizations.*