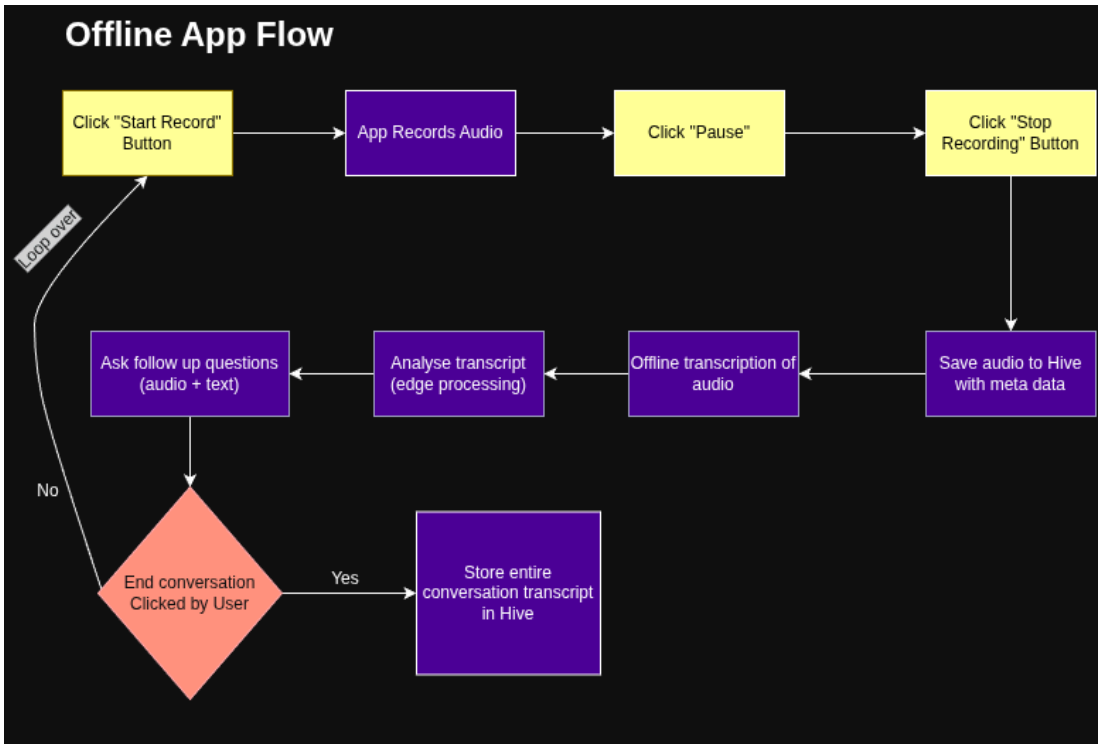
Technical Requirements:

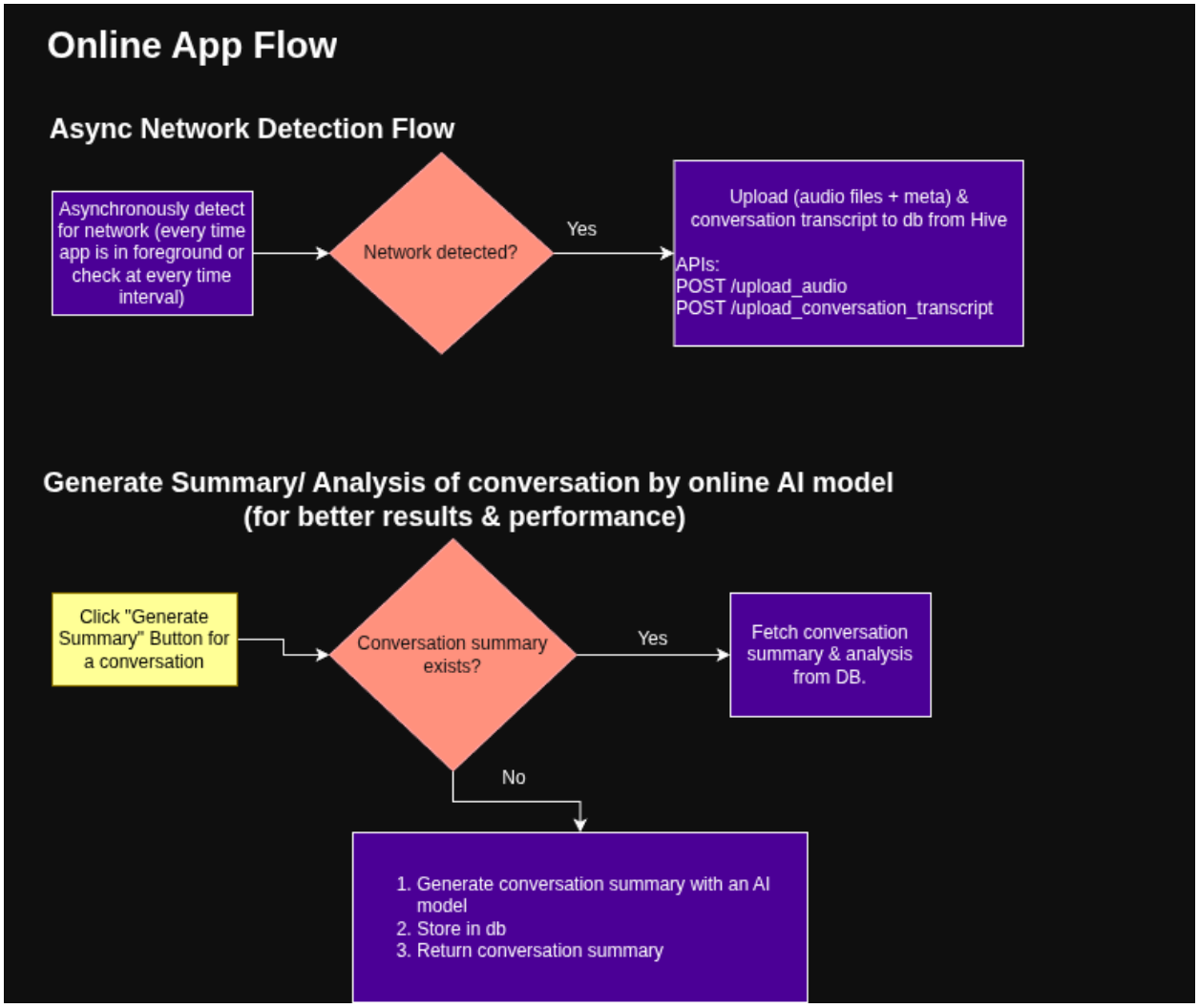
1. Record sound
2. Audio transcription (offline)
3. Analyse transcription & ask follow up questions
4. Store audio (offline) & eventually over cloud
5. Store transcription (offline) & eventually over cloud
6. Generate analysis/ summary of conversation
7. Option to end conversation

App Flow Diagrams:

Offline App Flow/ User Flow



Online App Flow/ User Flow



Tech Implementation Choices

Requirement	Tech	Reason
1. Record sound	flutter_sound	Supports audio recording + playback, custom UI, supporting streams & different codecs
2. Offline Audio transcription	flutter_whisper_kit	High accuracy, support of 100+ languages, allows choosing model sizes (tiny, base, small)
3. Store audio offline	1. Phone's filesystem for	Hive is NoSQL so allows easy schema

4. Store conversation transcription	offline audio storage 2. Hive to store meta data (file path, duration, conversation id, audio id, timestamp, farmer name etc.)	evolution. Fast performance.
5. Analyse transcription & ask follow up questions (offline)	1. SLM (**Gemma 2B + MediaPipe**) grounded via RAG (Retrieval-Augmented Generation) using a locally synced knowledge base of 'Scientist Research Goals' and 'Common Crop Issues'. 2. **Backend:** Chroma DB (Master Index) 3. **Mobile:** ObjectBox (Edge Replica) 4. **Sync:** Embeddings Sync Strategy	**SLM grounded via RAG** will give smarter analysis rather than a rule based system. Allows scientists to change the "Questions" dynamically from the cloud (Chroma DB) without updating the app code. We just sync a new Knowledge Base file. **Chroma DB (Cloud):** Stores the complete, high-dimensional vector index of all scientific research. Helps with similarity search. **ObjectBox (Mobile):** A lightweight, embedded vector store. We sync relevant embeddings from Chroma to ObjectBox when online. **Offline Inference:** The SLM queries ObjectBox locally to retrieve context ("Grounding") without needing an internet connection.
6. Detect network	1. connectivity_plus 2. internet_connection_checker_plus	Standard & recommended approach by Flutter community. internet_connection_checker_plus - an additional layer to check for actual internet connectivity (eg, a user can be connected to a Wifi with no internet connection). This ensures preventing failures while online syncs.
7. Backend APIs	Python FastAPI	Faster & robust backend APIs. Keeps the stack for backend & MLOps same.
8. Database	1. AWS S3/ Google Cloud Storage - audio files 2. MongoDB - meta data & other information	AWS S3/ Google Cloud Storage - standard practice to store media files. Fast retrieval due to CDN. MongoDB - fast, scaling NoSQL database which gives us schema flexibility for easily adding new features.

9. Generate conversation summary & analysis (online)	Gemini 1.5 Flash via Vertex AI (can be grounded using RAG from our knowledge base - Chroma DB) Chroma DB (RAG Grounding)	**Gemini 1.5 Flash** is optimized for high-volume, low-latency summarization tasks. By grounding it with **Chroma DB** (backend), we ensure the "Online Analysis" has access to the complete, up-to-date scientific database that might be too large to sync fully to the offline device.
10. State management	flutter_bloc	Industry standard for BLoC implementation. It provides robust tools for separating presentation from business logic, making the "Event-to-State" flow testable and predictable.
11. User Authentication	Supabase	Generous free tier, secure

Database Design/ Schema

Hive/ MongoDB

1. box/ collection - audios - stores meta for audio files. Actual audio is stored in the phone's file system
   Structure:

```
{
      audio_id: "",
      filepath: "",
      conversation_id: "",
      timestamp: "",
      duration: "",
      farmer_id: ""
}
```

2. box/ collection - conversations
   Structure:

```
{
      conversation_id: "",
      time: "",
      farmer_id: "",
```

```
      conversation: "farmer: This is maize
      AI: How tall is the crop?...."
}
```

Additional MongoDB Collection on Cloud

1. collection - summary
   Structure:

```
{
      conversation_id: "",
      time: "",
      farmer_id: "",
      summary: ""
}
```

Bloc Architecture