Technical Challenges I Forsee:

1. **Device storage limit.** We need to handle cases such as that. Perhaps deleting audio files from phone storage after a certain time period & replacing the file path with AWS S3 url in Hive
2. **Network Volatility:** Bridging the gap between zero-connectivity field environments and high-latency rural networks during synchronization.

Handling Latency and Offline Mode with Bloc

The **Bloc (Business Logic Component)** pattern was central to maintaining a responsive UI under varying network conditions:

- **Separation of Concerns:** By isolating UI from logic, the app remains fully functional offline. The `SyncBloc` handles the transition between `SyncInProgress` and `SyncSuccess/Failure` states, ensuring the UI never hangs during a network timeout.
- **Latency Masking:** Using asynchronous event handlers, the app provides immediate visual feedback (e.g., local saving) while deferring expensive network operations until the user triggers a sync.

Learning Path & Implementation

As my first professional implementation of Flutter and Bloc, this project served as a deep dive into structured state management. I focused on mastering the `BlocListener` and `BlocBuilder` relationship to ensure side effects (like toasts) and UI updates (like loading spinners) remained synchronized. This experience demonstrated how Bloc provides a predictable state machine, which is essential for building reliable, mission-critical tools for the agricultural sector.