# Week 2 – Consolidated Technical Learnings & Revision Notes

This document consolidates the key technical knowledge acquired during Week 2, focusing on call orchestration, scheduler design, provider integration, and failure handling.

The emphasis is on concepts worth revisiting, not a chronological chat log.

Key Architectural Principles Learned:

APIs create intent, not execution.
Schedulers execute intent.
Providers define reality.
Webhooks (or fallbacks) finalize state.

Violating these boundaries leads to brittle systems.

Scheduler Design Insights:

A scheduler must never assume success.
Any external provider call can fail.
CLAIMED state must always be recoverable.
Failure recovery is mandatory, not optional.

Schedulers without recovery become dead-letter systems.

Provider Integration Realities:

Different providers expose lifecycle events differently.
Assuming uniform webhook behavior across providers is incorrect.
Provider-agnostic fallback logic (timeouts, retries) is essential.

A correct architecture survives provider limitations.

Inbound vs Outbound Calls:

Inbound calls are provider-initiated but must still be tracked via provider_call_id.
Phone-number-based correlation is unsafe.
Provider identity is the only reliable anchor.

Inbound is not simpler; it is differently triggered.

Retry and Terminal State Handling:

Retries must be bounded.
Terminal states must be immutable.
Duplicate events must be idempotent.

State machines must be resilient to noise and repetition.

Operational Lessons:

Local development requires public reachability for true webhook testing.
ngrok is a development bridge, not a production solution.
Dirty databases destroy debugging clarity.

Reset state before reasoning.

Final Reflection:

Most integration failures are not code bugs but mismatched mental models.
Clear separation of responsibility between systems prevents endless debugging.

This knowledge is reusable across any orchestration, scheduling, or distributed system.