Week 2 – Scheduler & Rules Engine (AI Calling Platform)

This document is a strict continuation of Week 1. Follow it exactly. No AI conversations yet. This week stabilizes call execution, retries, and rules.

Day 1 – Scheduler Hardening

- Refactor scheduler to be its own module.
- Ensure scheduler runs independently of API traffic.
- Add locking to avoid duplicate execution.
- Verify scheduler only picks valid calls.
- Add logs for every scheduler run.
- Commit code.

Day 2 – Retry Policy Core

- Define retry rules (max_retries, retry_delay_minutes).
- Add retry_count column to calls table.
- Implement retry increment logic.
- Prevent retries after max_retries.
- Test retry exhaustion scenarios.
- Commit code.

Day 3 – Rules Configuration Layer

- Create rule config structure (JSON or DB-based).
- Map bot type → retry policy.
- Validate rule configs at runtime.
- Ensure defaults exist for all bots.
- Commit code.

Day 4 – Provider Event Normalization

- Normalize provider events into internal events.
- Handle ANSWERED, COMPLETED, NO_ANSWER, FAILED.
- Update call state correctly per event.
- Ensure idempotent event handling.
- Commit code.

Day 5 – Rescheduling Logic

• On NO_ANSWER, calculate next_action_at.

• Persist rescheduled time.

• Ensure scheduler respects new times.

• Prevent immediate retry loops.

• Commit code.

Day 6 – Failure & Stop Conditions

• Define stop conditions per bot.

• Stop retries on COMPLETED or terminal failures.

• Mark calls as FAILED when exhausted.

• Verify no calls remain stuck.

• Commit code.

Day 7 – End-to-End Stability Test

• Simulate multiple bots with different rules.

• Run scheduler continuously.

• Validate DB integrity.

• Write Week 2 documentation.

• Final commit for Week 2.

Week 2 Rules (Must Follow)

• No AI conversations.

• No UI.

• No provider-specific hacks.

• Stability over features.

If Week 2 is completed, the platform becomes execution-safe and scalable.