**Step 5: Test and Refine the Solution (Debug and Verify)**

1. **Test Setup (assumptions)**

* Schedule: 08:00, 18:00
* Portion per feed: 120 g
* Servo calibration: 30 g per cycle (4 cycles for 120 g)
* Wait window before consumption check: 10 minutes
* Consumption threshold to count as “eaten”: >= 24 g (20% of portion, min 15 g)
* Minimal increase after dispensing to confirm “food actually dropped”: >= 15 g (½ cycle)

1. **Scenarios, expected results, and actual results**

**T1 — Pet eats as expected**

Inputs: Hopper OK; Door closed; Bowl before 50 g; After dispense 170 g; After wait 90 g

Observed changes: Increase +120 g, Consumed 80 g

Expected: CONSUMED\_OK (no alert)

Actual (from simulation): CONSUMED\_OK

Notes: Logging shows normal cycle count and healthy consumption.

**T2 — Pet does not eat**

Inputs: Bowl before 40 g; After dispense 160 g; After wait 158 g

Observed changes: Increase +120 g, Consumed 2 g < 24 g

Expected: NOT\_EATEN (send alert)

Actual: NOT\_EATEN

Notes: Alert should prompt staff to check pet/bowl. LED amber blink.

**T3 — Food bin empty**

Inputs: food\_level\_state = EMPTY

Expected: SKIPPED\_EMPTY + HOPPER\_EMPTY alert, no dispense

Actual: SKIPPED\_EMPTY + HOPPER\_EMPTY

Notes: Fail-fast behavior is correct; prevents motor run.

**T4 — Dispense failure (no measurable increase)**

Inputs: Bowl before 55 g; After dispense 58 g

Observed changes: Increase +3 g < 15 g minimal

Expected: DISPENSE\_FAIL + alert

Actual: DISPENSE\_FAIL

Notes: Likely jam or empty auger; staff intervention required.

**T5 — Door open (safety interlock)**

Inputs: door\_open = true

Expected: BLOCKED\_DOOR + DOOR\_OPEN alert, no dispense

Actual: BLOCKED\_DOOR

Notes: Safety interlock working; protects pets/hands.

**T6 — Partial consumption just below threshold**

Inputs: Bowl before 60 g; After dispense 180 g; After wait 157 g

Observed changes: Consumed 23 g (< 24 g threshold)

Expected: NOT\_EATEN (borderline)

Actual: NOT\_EATEN

Notes: Good edge case that validates the threshold logic.

**T7 — Network down during alert**

Inputs: Same pattern as “not eaten” but network\_ok = false

Expected: NOT\_EATEN alert + local fallback (buzzer/LED/log)

Actual: NOT\_EATEN with note “Network down; use local buzzer/LED and log”

Notes: System remains safe and observable without cloud.

1. **Summary Table**

| **ID** | **Scenario** | **Expected** | **Actual** | **Match** | **Alerts** |
| --- | --- | --- | --- | --- | --- |
| T1 | Pet eats as expected | CONSUMED\_OK | CONSUMED\_OK | Yes | – |
| T2 | Pet does not eat | NOT\_EATEN | NOT\_EATEN | Yes | NOT\_EATEN |
| T3 | Food bin empty | SKIPPED\_EMPTY | SKIPPED\_EMPTY | Yes | HOPPER\_EMPTY |
| T4 | Dispense failure | DISPENSE\_FAIL | DISPENSE\_FAIL | Yes | DISPENSE\_FAILURE |
| T5 | Door open | BLOCKED\_DOOR | BLOCKED\_DOOR | Yes | DOOR\_OPEN |
| T6 | Partial consumption < threshold | NOT\_EATEN | NOT\_EATEN | Yes | NOT\_EATEN |
| T7 | Network down during alert | NOT\_EATEN | NOT\_EATEN | Yes | NOT\_EATEN (local fallback) |

1. **Discussion of results (logic Validation)**

* The core loop triggers only at schedule times and correctly stops when pre-checks fail (empty hopper, door open).
* Dispense verification via bowl weight increase detects jams/misfeeds.
* Consumption verification after a realistic wait window catches “not eaten” situations.
* Local fallback ensures alerts still reach staff (buzzer/LED/log) when network is down.
* Edge thresholds (minimal increase + consumption threshold) handle noise and partial nibbling

1. **Refinements (improvements)**

* **Adaptive threshold:** For tiny portions or small pets, set consumed\_delta\_g = max(12 g, 15-25% of portion) to reduce false “not eaten” alerts.
* **Retry-once logic:** If DISPENSE\_FAIL, auto-retry one extra cycle before alerting jam.
* **Denounce and averaging:** Average 3-5 rapid weight readings to reduce sensor noise.
* **Spillage detection:** If bowl increase is high but camera/tilt sensor detects spill, raise SPILL\_SUSPECTED instead of success.
* **Grace period:** If pet is observed eating slowly, extend wait window from 10 → 15 min before judging NOT\_EATEN.
* **Low-food early alert:** When LOW (e.g., <20%), alert proactively so EMPTY never occurs at meal time.
* **Schedule safeguard:** Mark a time slot as “served” for ~10 minutes to avoid double-feeding if clocks jitter.
* **Health insights:** Log daily consumption totals; alert if a rolling 3-day average drops by > 30%.
* **Battery and time sync:** Add warnings for low battery, RTC drift > 30s.
* **Manual feed audit:** Lockout after manual feed (e.g., 3-5 min) to avoid accidental overfeeding.