**Step 5: Test and Refine the Solution**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Scenario** | **Expected → Actual** | **Result** |
| TC1 | Pet eats normally (100g dispensed; ~20 g remaining) | No alerts → No alerts; log status=OK (consumed ~180 g) | PASS |
| TC2 | Pet does not eat (~95% remaining after 10 min) | Alert 'Not eaten' → Alert fired; log NOT EATEN | PASS |
| TC3 | Hopper empty at feed time | Alert 'Hopper EMPTY'; no dispense → As expected | PASS |
| TC4 | Jam on first attempt; retry succeeds | Retry once → Success; log OK | PASS |
| TC5 | Jam persists beyond retries | Alert 'Jam detected'; no feed → As expected | PASS |
| TC6 | Small portion rounding (cat 80 g) | About 80 g target → Measured ~78 g; kibble variance | OBSERVE |

**Discussion of Logic**

• TC1 (Normal Feeding): The system provided the right volume, the consumption was recorded properly, and no alerts were raised. This helps us verify that the inner feeding pattern and weight measurement is working as required.

• TC2 (Pet Did Not Eat): The system indicated that the animal did not eat and generated an alarm as it should. This indicates efficient monitoring and error-detection following the feeding process.

• TC3 (Empty Hopper): The controller recognized that there is no food and prevented a failed dispense that is non-safety and transparent.

• TC4 and TC5 (Jam Handling): TC4 had the retry logic behaving as designed and TC5 had a persistent failure causing a suitable alert. This justifies the strength of handling mechanical errors.

• TC6 (Rounding Portion): In fact, the system filled in about 78 g as opposed to the intended 80 g. Although this is acceptable limit of variance when feeding on kibbles, it shows a precision limitation because of granularity and mechanical tolerance. The case is marked OBSERVE indicating small pets may need more calibration or adaptive logic.

**System Refinements**

* **Precision Dispensing**: Add dynamic calibration and kibble profiling to improve accuracy for small portions.
* **Jam Handling**: Use adaptive retry logic and sensor-based detection to better manage feed jams.
* **Consumption Monitoring**: Enable time-based tracking and customizable alert thresholds to detect feeding anomalies.
* **Alert System**: Introduce multi-level alerts with suggested actions and mobile notifications for better user experience.
* **Modular Design**: Support plug-in sensors, remote diagnostics, and firmware updates for future scalability.
* **Testing Framework**: Develop simulation tools and automated tests to validate system reliability under edge conditions.