Step5: Test and Refine the Solution

Test Case 1: Pet Eats Normally (Normal Path)

**Scenario:** For example, it is feeding time (8:00 AM), and there is enough food in the food container for the pet to eat. The bowl is empty before feeding.

**Expected Result:** The system dispenses a serving of food at 8:00 AM. The bowl's weight increases by approximately 50 grams. The pet eats within 10 minutes and eats all food in the bowl, so the bowl's weight drops back down to nearly 0 grams. Since everything is normal, no alerts are generated.

**Actual Result (algorithm behaviour):** At 8:00 AM, the “food\_available” check passes (TRUE), so the servo dispenses food. The pseudocode records the increase in “new\_bowl\_weight” (e.g., from 0 grams to 50 grams, indicating success). After 10 minutes, assume that “latest\_bowl\_weight” reads approximately 5 grams (with some crumbs remaining). This is less than “new\_bowl\_weight”, so the "pet did not eat" condition is false. The code continues to the else branch, indicating a successful feeding. So, no alerts are generated. This is as expected.

**Result:** Passed. The normal feeding scenario works correctly. No improvements are required here, but we note that logging successful feeding events (for recordkeeping) would be beneficial to the system.

Test Case 2: Pet Does Not Eat Food

**Scenario:** Feeding time arrives (6:00 PM), food is dispensed, but the pet does not eat (perhaps because the pet is not hungry or is absent). After the food is dispensed, the bowl's weight changes from 0 grams to 50 grams and remains at 50 grams even after 10 minutes (the food remains untouched).

**Expected Result:** After waiting 10 minutes, the system should detect that the pet has not eaten the food and send an alert to notify the staff. No more food will be dispensed until the next scheduled time.

**Actual Result:** At 6:00 PM, the feeder dispenses food (because “food\_available” is true). The weight increases (e.g., 0 grams -> 50 grams). The system waits 10 minutes. The “latest\_bowl\_weight” remains approximately 50 grams (no change). The condition “latest\_bowl\_weight >= new\_bowl\_weight” is true, so the pseudocode sets “alert\_status = ALERT\_NOT\_EATEN” and calls “SEND\_ALERT” ("Pet did not eat..."). This works exactly as expected: the system alerts that the food has not been consumed.

**Result:** Passed. This logic successfully handles the situation where the pet has not eaten and alerts the staff. However, there is still an important consideration: what to do if the pet has eaten but not finished the meal. I've included this possibility in the following improvement considerations for further discussion.

Test Case 3: Food Bin Empty (No Food to Dispense)

**Scenario:** It’s time to feed, but the system operators do not know, the food hopper is empty. The schedule calls for a feeding at 8:00 AM, but there is no food to drop.

**Expected Outcome:** The system should realize there’s no food available before attempting to dispense and immediately alert the staff that the container is empty (so they can refill it). The motor should not run (or if it does, it results in no food, which should also trigger an alert).

**Actual:** At the feeding time, the pseudocode checks “food\_available”. If the container sensor correctly reports empty (FALSE), the code goes into the branch that sets “ALERT\_FOOD\_EMPTY” and calls “SEND\_ALERT ("Food container is EMPTY…") without activating the servo. This alert would occur right at 08:00. If the sensor was not available and the system only detected the issue by weight, then after attempting to dispense, “new\_bowl\_weight” would equal “last\_bowl\_weight”, triggering the “no food dispensed” alert. In either case, an alert is sent at feeding time. This matches the expectation. Staff would be notified immediately that the feeder failed to dispense due to lack of food.

**Result:** Pass. The algorithm handles empty-food conditions. To improve user experience, the system might continuously monitor the level of food which in the container and perhaps alert before a scheduled feeding if food is low, rather than waiting until the moment of feeding.

**Comparison with Expected Results:**

For each scenario, the actual results of the implemented logic matched the expected results. The system correctly distributed and monitored food under normal conditions and correctly identified and flagged error conditions (e.g., empty containers, uneaten food). Progressive testing confirmed that the integrated problem-solving approach provided effective solutions to the defined problems. However, significant areas for improvement remained.

**Suggested Improvements and Refinements:**

**Low Food Alert:**

As mentioned above, the system can monitor the food sensor regularly (not just at feeding time) and issue an alert before the scheduled feeding time, prompting the user to refill the food. Regarding the algorithm, the completely low alert can be changed to an alert when the food is less than 100g (not enough for two feedings), which can prevent missed feedings.

**Adjustable Alarm Time:**

The 10-minute wait time is a simple assumption. We can make the "pet not eating" detection time configurable (some pets may eat immediately, others may take longer, and sometimes they simply don't consume the food in the allotted time). A wait time that is too short may trigger false alarms, while a wait time that is too long may delay necessary intervention. Optimizing this parameter or making it adaptive can improve accuracy.

**Multiple Pets/Portions:**

If a shelter uses a single feeder to feed multiple pets or larger animals, we may need to dispense different amounts or multiple portions of food. The algorithm can be extended to control portion sizes or repeat dispensing cycles (along with sensors to prevent overfeeding) as needed. For now, we assume that only one portion of food is dispensed at each scheduled time.

**Logging and User Interface:**

In a real-world implementation, it would be useful to log every event (dispensing time, quantity, whether the pet consumed the food, alerts sent) to a log file or cloud dashboard. A small improvement would be to include these logs in our pseudocode (for example, by including some PRINT statements for simulation).

**Error recovery:**

If a "no food dispensed" alarm occurs (which could indicate a blockage or malfunction), the system could automatically attempt a second dispense after a short delay, or activate a motor vibration to clear the blockage, for example. This is a potential enhancement, but it's worth considering.