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

System Word Code:

|  |
| --- |
| 1. **START SYSTEM** 2. Check Tray\_Weight.    1. If Tray\_Weight is greater than LEFTOVERS\_BUFFER, check Current\_Time.       * If Current\_Time is less than 20 minutes after a FEEDING\_TIMES, then proceed to step 3).       * Otherwise, send Alert\_Food\_Unconsumed then **STOP SYSTEM**.    2. Otherwise, proceed to step 3). 3. Check if Current\_Time and compare it to all FEEDING\_TIMES.    * + If Current\_Time does not match any entries in FEEDING\_TIMES, wait for 1 minute and return to step 2).      + Otherwise, proceed to step 4). 4. Check Stored\_Food and compare it to MEAL\_AMOUNT and Tray\_Weight.    1. If Stored\_Food is less than MEAL\_AMOUNT - Tray\_Weight, then begin Alert\_Insufficient\_Food and **STOP SYSTEM.**    2. Otherwise, proceed to step 5). 5. Begin Dispense\_Food until Tray\_Weight equals MEAL\_AMOUNT, then stop Dispense\_Food. 6. Check if Current\_Time is after WORK\_END\_TIME.    1. If it is, **STOP SYSTEM.**    2. Otherwise, wait 1 minute and return to step 2). |

#### Sample Scenarios:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenario** | **Tray\_Weight** | **Stored\_Food** | **Current\_Time** | **Case Process & Result** |
| Pet eats food | 0g | 100g | 08:00 | Tray is detected as empty → Current time is detected as a feeding time → Amount of stored food is sufficient to dispense a meal → 35g of food is dispensed → system loops → Tray is detected as full → current\_Time is less than 20 minutes after a feeding time so system continues → after 20 minutes checks weight again, Tray\_Weight is 0g as pet ate all the food → system waits for next feeding time  **Result Output:** 35g of Food is dispensed successfully and the Pet has eaten it.  Result is expected and meets requirements. |
| Leftovers are present / pet doesn’t eat the meal | 24g | 100g | 18:57 | Tray is detected as full → Current time is detected as more than 20 minutes after a feeding time → System sends an Alert\_Food\_Unconsumed and Stops.  **Result Output:** Alert\_Food\_Unconsumed occurs and the system automatically stops.  Result is expected and meets requirements. |
| It’s not feeding time | 0g | 100g | 12:42 | Tray is detected as empty → Current time is not a feeding time → system pauses for 1 minute and loops.  **Result Output:** No output.  Result is expected and meets requirements. |
| End of the workday | 0g | 30g | 22:00 | Tray is detected as empty → Current time is not a feeding time → system pauses for 1 minute and loops.  **Result Output:** No output.  Result is unexpected due to a logical error in the system’s end of workday detection. See Evaluation section for advised fix. |
| Not Enough Food at feeding time | 0g | 23g | 18:30 | Tray is detected as empty → Current time is detected as a feeding time → Amount of stored food is insufficient to dispense a meal → System sends an Alert\_Insufficient\_Food and Stops.  **Result Output:** Alert\_Insufficient\_Food occurs and the system automatically stops.  Result is expected and meets requirements. |

#### Evaluation of System Logic:

Overall the logic for the system mostly functions as expected and required (with an exception discussed below) and should work on low-cost components primarily only requiring scales, a motor and a buzzer or speaker.

The main issue with the first draft of the system is that the check to verify if the day is over will not properly occur as the decision is at the end of the feeding process. To fix this, the following new system code is advised:

|  |
| --- |
| 1. **START SYSTEM** 2. Check Tray\_Weight.    1. If Tray\_Weight is greater than LEFTOVERS\_BUFFER, check Current\_Time.       * If Current\_Time is less than 20 minutes after a FEEDING\_TIMES, then proceed to step 3).       * Otherwise, send Alert\_Food\_Unconsumed then **STOP SYSTEM**.    2. Otherwise, proceed to step 3). 3. Check if Current\_Time and compare it to all FEEDING\_TIMES.    * + If Current\_Time does not match any entries in FEEDING\_TIMES, proceed to step 6).      + Otherwise, proceed to step 4). 4. Check Stored\_Food and compare it to MEAL\_AMOUNT and Tray\_Weight.    1. If Stored\_Food is less than MEAL\_AMOUNT - Tray\_Weight, then begin Alert\_Insufficient\_Food and **STOP SYSTEM.**    2. Otherwise, proceed to step 5). 5. Begin Dispense\_Food until Tray\_Weight equals MEAL\_AMOUNT, then stop Dispense\_Food. Then proceed to Step 6). 6. Check if Current\_Time is after WORK\_END\_TIME.    1. If it is, **STOP SYSTEM.**    2. Otherwise, wait 1 minute and return to step 2). |

(Altered sections of the word code from the previous version are highlighted)

By moving the ‘end of day’ check over to just before the system loops, the system will verify whether it’s end of day properly. In the event that the tray is full and the workday is over, an alert will be sent, however this is preferable as it would indicate issues that staff would need to address regardless.

An accompanying altered diagram for this system has also been included. (Higher Res image is provided in accompanying documentation)

A screenshot of a computer

AI-generated content may be incorrect.

#### Proposed Future Improvement

* **Improvement Possibility #1:** One of the largest issues with the current system I’ve designed is that it assumes that nothing but food will weigh on the feeding tray. Though the design of the feeder itself could significantly reduce the odds of an animal stepping on the tray, edge case scenarios could appear that would result in false alerts being sent to staff.

**Proposal:** A sensor could be attached to each feeder which detects animals (perhaps via movement or image recognition). This sensor would be used when the system goes to verify the contents of the tray as the final step of the alert process if an animal is detected inside the feeding tray and messing up the feeding process, to prevent false positives.

* **Improvement Possibility #2:** Initially I attempted to design the system in such a way that shelter employees could manually adjust the feeding times and the amount to be dispensed. However, this ran into conflict with some of the goals of the Background (primarily the ‘low-cost’ request and the task being to design the *logic and behaviour* of the system rather than it’s operation). Whilst I still think a system like this should have the capacity for this, I decided to proceed with an assumption that the shelter would manufacture the system’s times and feeding amounts to requirement, though this may be somewhat outside the scope of this task.

**Proposal:** Additional components added to the feeders which allow for the manual setting of additional (or fewer) feeding intervals and manual adjustment of each interval’s feeding amount. This would allow multiple feeders to function properly for different animals in the shelter and be swapped without issue, such as adjusting feed times for animals who require more frequent feeding or for enclosures with stricter dietary needs.