Step 3: Design the Algorithm (Planning the Solution)

In this step, I planned the solution as a series of decision points and actions:

**Main Loop and Timing:** The feeder's controller continuously checks the current time against the feeding schedule. This can be implemented as a loop that runs every minute. If it's the scheduled feeding time, the feeding sequence begins. If the time hasn't arrived, the system waits or continues checking (without performing any significant action).

**Pre-Dispense Check (Food Available):** When the feeding time arrives, the algorithm first verifies whether food is in the container (using the food container sensor). If the food container is empty, the system should skip dispensing and immediately alert the worker that the food cannot be dispensed due to insufficient supply. This is a pre-handled error condition. (For example, "Is there food in the container? If not, send an alert: 'Food container empty.'")

**Dispense Action:** If food is present, the system activates the dispenser (activates the servo motor) and places a portion of food into the bowl. This is a process action (for example, "Put food in bowl").

**Post-Dispense Verification:** After dispensing, the algorithm uses the bowl weight sensor to confirm that the food was dispensed. It records the weight of the bowl before and after dispensing for comparison. If the bowl's weight doesn't increase as expected when the motor is running, the system triggers an alarm (e.g., "Food not dispensed successfully"). This is another error condition check implemented in the logic.

**Wait and monitor consumption:** If the food is dispensed successfully (the bowl's weight increases by approximately one serving), the system enters the monitoring state. It waits 10 minutes to give the pet time to eat. After the wait, the system checks the bowl's weight again. If the pet has eaten, the bowl's weight should drop significantly (closer to the original weight, accounting for any small amount of leftover food). If the bowl's weight doesn't drop (meaning the food is still untouched), the system assumes the pet hasn't eaten. In this case, it sends an alarm to the staff (e.g., "Alert: Pet did not eat"). This facilitates follow-up.

**Loop continues:** After processing a feeding event (either a normal feeding or an alarm), the system returns to the monitoring state until the next scheduled feeding time. This logic repeats daily at each feeding time. It continues in this loop if the system is powered on.

These decision points and actions can be visualized in a flowchart. This is a flowchart of the pet feeder algorithm I created in Draw.io.

