Macalalag, Marion Elizabeth P.
U3303144
Masters in Business Informatics
4478 Introduction to Information Technology Assignment 1

Part 3 : AI Integration

Overall Reflection :

The two key areas that I've decided to prompt co-pilot with is the real world implementation of the pet feeder and the creation of a professional readme file for project presentation. First, in terms of the real world implementation, upon prompting the co-pilot, I was able to further improve on the project being able to see not just the internal logic but also how to implement the project in the real world setting. It introduced me to reliable hardware components that can be considered in the implementation of the project which ultimately helped me gain a better understanding of the project externally. The co-pilot was also a huge help in terms of the creation of a readme.md of the system. It provided a very detailed readme taking into consideration all aspects from background, logic, software, hardware and even installation. This has been of huge help for me as it is able to summarize clearly what I want to achieve with my pet feeder project

Overall, the co-pilot definitely influenced my final solution and understanding of the problem, with the copilot I was able to take into consideration factors I initially did not consider. While many of its suggestions were insightful and led to meaningful improvements, there were moments when the output still lacked reliability—reminding me of the importance of verifying AI-generated content with human judgment.
One key takeaway from the activity is that although co-pilot is a powerful assistant, users must remember that its suggestions are tools and not truths and must always be paired with human judgement, good data and clear goals.

[Appendix of Prompt and Output below]**

Macalalag, Marion Elizabeth P.
U3303144
Masters in Business Informatics
4478 Introduction to Information Technology Assignment 1

<u>APPENDIX</u> :

Chosen Questions :

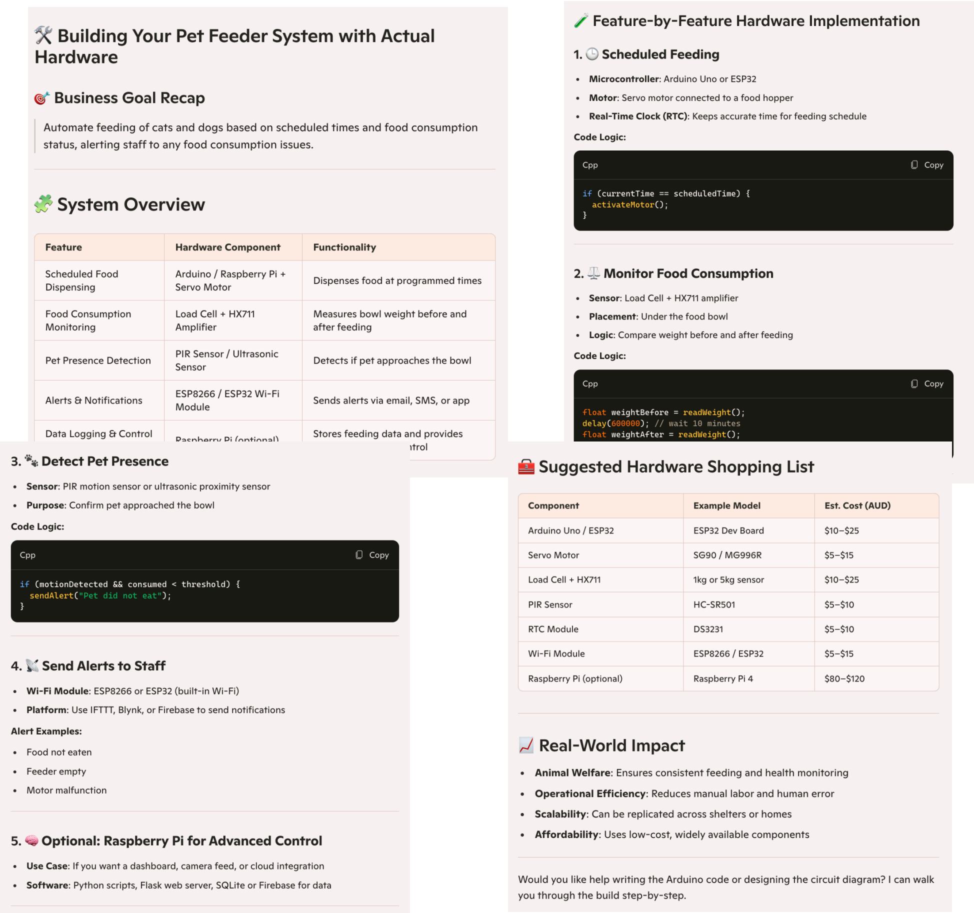
1. Explore real-world implementation: Use Copilot to discuss how your system could be built using actual hardware (e.g., Arduino, Raspberry Pi).

Co Pilot Prompt : How can my current pet feeder system be built using actual hardware (e.g., Arduino, Raspberry Pi)? This is the overall business goal of my system : Automate feeding of cats and dogs based on scheduled times and food consumption status, alerting staff to any food consumption issues.

Key features of the system include :

- Dispense of food for cats and dogs at scheduled times
- Monitor whether food has been consumed or the amount of food that has been consumed.
- Alert staff on issues regarding food dispenses or if food is not eaten

Co Pilot Response :



🛠️ Building Your Pet Feeder System with Actual Hardware

🎯 Business Goal Recap

> Automate feeding of cats and dogs based on scheduled times and food consumption status, alerting staff to any food consumption issues.

🧩 System Overview

Feature	Hardware Component	Functionality
Scheduled Food Dispensing	Arduino / Raspberry Pi + Servo Motor	Dispenses food at programmed times
Food Consumption Monitoring	Load Cell + HX711 Amplifier	Measures bowl weight before and after feeding
Pet Presence Detection	PIR Sensor / Ultrasonic Sensor	Detects if pet approaches the bowl
Alerts & Notifications	ESP8266 / ESP32 Wi-Fi Module	Sends alerts via email, SMS, or app
Data Logging & Control	Raspberry Pi (optional)	Stores feeding data and provides ntrol

🧪 Feature-by-Feature Hardware Implementation

1. 🕒 Scheduled Feeding

- **Microcontroller**: Arduino Uno or ESP32
- **Motor**: Servo motor connected to a food hopper
- **Real-Time Clock (RTC)**: Keeps accurate time for feeding schedule

Code Logic:

```cpp
if (currentTime == scheduledTime) {
  activateMotor();
}
```

2. ⚖️ Monitor Food Consumption

- **Sensor**: Load Cell + HX711 amplifier
- **Placement**: Under the food bowl
- **Logic**: Compare weight before and after feeding

Code Logic:

```cpp
float weightBefore = readWeight();
delay(600000); // wait 10 minutes
float weightAfter = readWeight();
```

3. 🐾 Detect Pet Presence

- **Sensor**: PIR motion sensor or ultrasonic proximity sensor
- **Purpose**: Confirm pet approached the bowl

Code Logic:

```cpp
if (motionDetected && consumed < threshold) {
  sendAlert("Pet did not eat");
}
```

4. 📡 Send Alerts to Staff

- **Wi-Fi Module**: ESP8266 or ESP32 (built-in Wi-Fi)
- **Platform**: Use IFTTT, Blynk, or Firebase to send notifications

Alert Examples:

- Food not eaten
- Feeder empty
- Motor malfunction

5. 🧠 Optional: Raspberry Pi for Advanced Control

- **Use Case**: If you want a dashboard, camera feed, or cloud integration
- **Software**: Python scripts, Flask web server, SQLite or Firebase for data

🧰 Suggested Hardware Shopping List

Component	Example Model	Est. Cost (AUD)
Arduino Uno / ESP32	ESP32 Dev Board	\$10–\$25
Servo Motor	SG90 / MG996R	\$5–\$15
Load Cell + HX711	1kg or 5kg sensor	\$10–\$25
PIR Sensor	HC-SR501	\$5–\$10
RTC Module	DS3231	\$5–\$10
Wi-Fi Module	ESP8266 / ESP32	\$5–\$15
Raspberry Pi (optional)	Raspberry Pi 4	\$80–\$120

📈 Real-World Impact

- **Animal Welfare**: Ensures consistent feeding and health monitoring
- **Operational Efficiency**: Reduces manual labor and human error
- **Scalability**: Can be replicated across shelters or homes
- **Affordability**: Uses low-cost, widely available components

Would you like help writing the Arduino code or designing the circuit diagram? I can walk you through the build step-by-step.

Macalalag, Marion Elizabeth P.
U3303144
Masters in Business Informatics
4478 Introduction to Information Technology Assignment 1

2. Ask Copilot to help you write a professional README.md file or summarize your project for presentation.

Prompt : Write a professional README.md file for my project pet feeder :

```markdown
# 🐾 Automated Pet Feeder System

## Overview

This project simulates and designs a low-cost, programmable automated pet feeder for use in animal shelters. The system is capable of dispensing food at scheduled times, monitoring consumption, and alerting staff if any issues arise (e.g., food not dispensed, pet not present, food not eaten).

The goal is to create a reliable, scalable solution using affordable components such as servo motors, load cells, and basic sensors, with logic that can be implemented on microcontrollers like Arduino or ESP32.

---

## Features

- ⏰ **Scheduled Feeding**: Dispenses food at configurable times (e.g., 8:00 AM and 6:00 PM).
- ⚙️ **Servo-Controlled Dispensing**: Uses a motor to dispense precise portions.
- ⚖️ **Weight Monitoring**: Load cell measures bowl weight before and after feeding.
- 🐕 **Pet Presence Detection**: Optional sensor to detect if the pet is nearby.
- 📊 **Consumption Tracking**: Monitors whether food has been eaten within a set time window.
- 🚨 **Alert System**: Sends notifications to staff if feeding fails or food is untouched.
- 💾 **Data Logging**: Stores feeding events, sensor readings, and alerts in a database or cloud service.
- 💡 **Status Indicators**: Visual feedback via LEDs (e.g., green for success, red for error).

---

## Hardware Requirements

| Component            | Description                          |
|----------------------|--------------------------------------|
| Microcontroller      | ESP32, Arduino Uno, or Raspberry Pi |
| Servo Motor          | SG90 or equivalent                   |
| Load Cell + HX711    | For bowl weight measurement          |
| Food Level Sensor    | IR or ultrasonic sensor              |
| Pet Presence Sensor  | PIR, RFID, or BLE beacon             |
| LED Indicator        | RGB LED or Neopixel strip            |
```

~~~
| Wi-Fi Module          | Built-in (ESP32) or external         |
| Power Supply          | 5V/12V adapter or battery pack       |

---

## Software Requirements

- Arduino IDE or PlatformIO (for microcontroller code)
- Python (for simulation or Raspberry Pi implementation)
- Firebase / Google Sheets API (for cloud logging)
- Twilio / SMTP (for alerts via SMS or email)

---

## Installation & Setup

1. **Hardware Assembly**
   - Connect servo motor to dispensing mechanism.
   - Mount load cell under feeding bowl.
   - Install sensors for food level and pet presence.
   - Connect components to microcontroller.

2. **Software Deployment**
   - Flash the microcontroller with the feeder logic.
   - Configure feeding times, portion sizes, and thresholds.
   - Set up cloud logging and alert services.

3. **Testing**
   - Calibrate load cell.
   - Simulate feeding cycles.
   - Verify alerts and data logging.

---

## Logic Flow Summary

```text
1. At scheduled time (e.g., 8AM or 6PM), check if feeding has occurred.
2. If not:
   - Rotate servo to dispense food.
   - Measure bowl weight before and after.
   - Verify food was dispensed successfully.
   - Check pet presence (optional).
   - Wait 10–45 minutes and monitor bowl weight.
   - If food was eaten, log success and show green light.
   - If not, trigger alert and show red light.
~~~