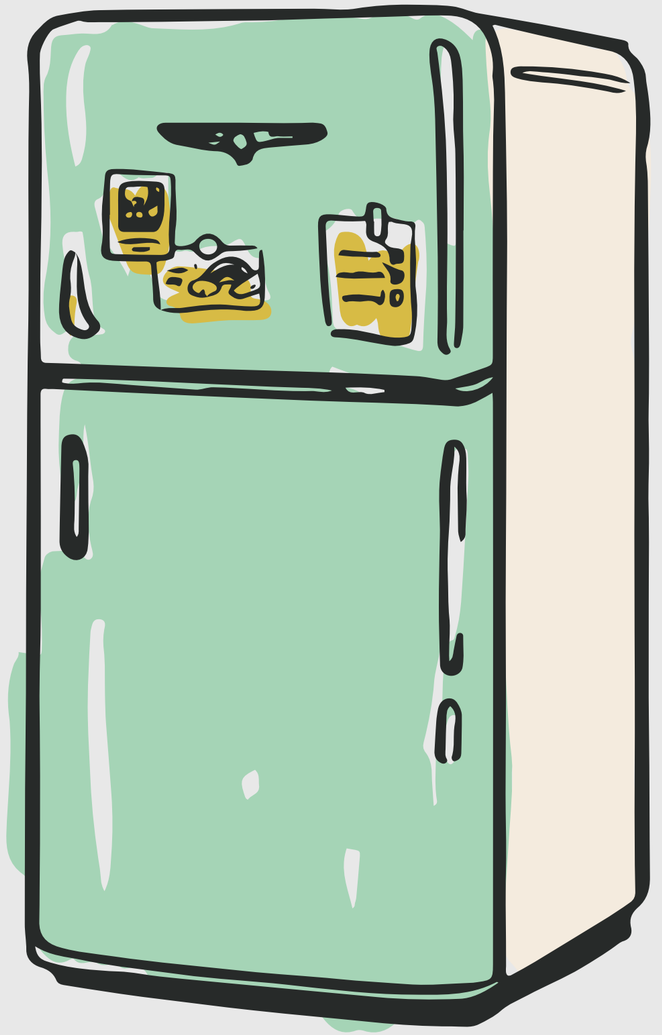
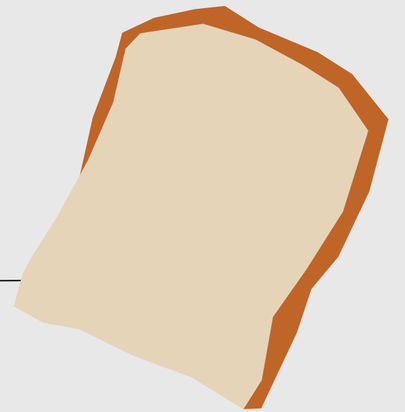
Smart Fridge Guardian

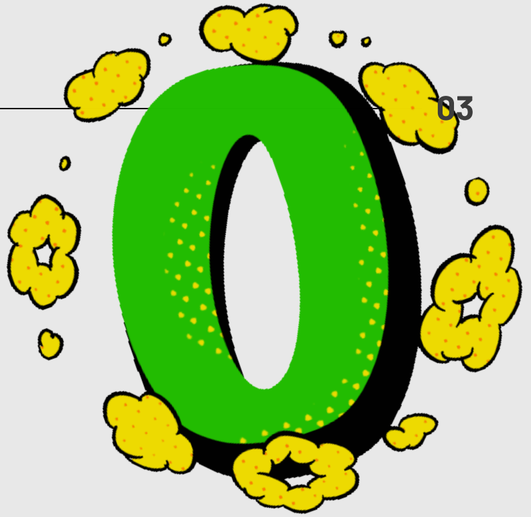
Common "Fridge Storms" in Student Dorms / Offices



Presented by Group 4



Question



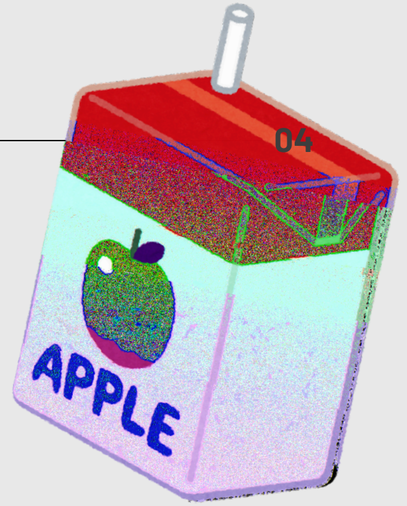
Has your food ever “disappeared” from a shared fridge?

A：Yes, and I was furious!

B：Yes, but I forgot to label it

C：Nope! I always guard my food like treasure.

Question

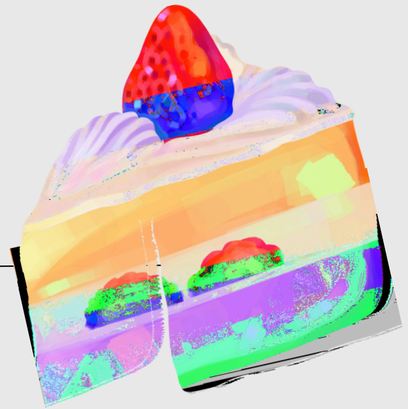


What’s the scariest thing you’ve ever seen in a shared fridge?

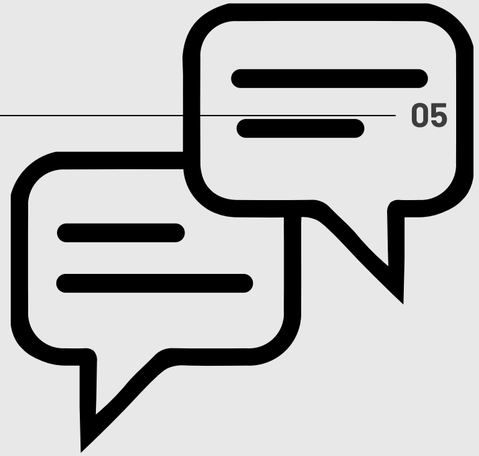
A：A rotten juice bottle

B：A mystery cake that’s been there for weeks

C：I don’t even dare to open the fridge



Question



Which fridge management method do you think is the most useless?

A：Verbal rules no one remembers

B：Group chat reminders nobody reads

C：The “silent agreement” that never works

SPECIAL

Project Objectives We Aim to Achieve

- **Smart Item Management**

Allow users to register food items (name, owner, expiration date) via a Web or mobile App, and keep track of their status.

- **Real-Time Status Monitoring**

Use an ESP32-CAM to detect door opening events and take a photo at the moment of access.

- **Reliable Data Handling**

Transmit door opening records and images to AWS through a Raspberry Pi; store user info, item logs, and event data in the cloud.

- **Smart Alerts & Visualization**

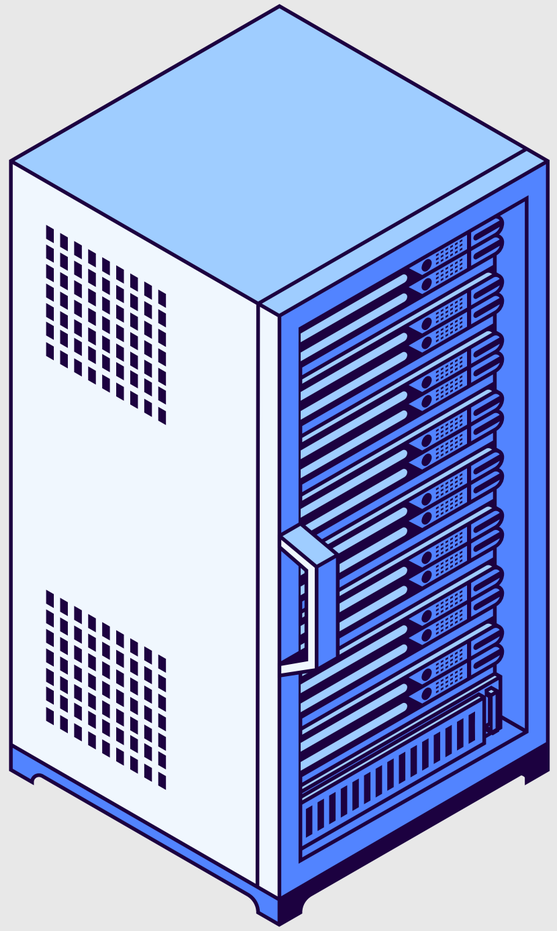
Provide expiration reminders via Email/SNS and display all fridge contents and activity logs on a simple dashboard.

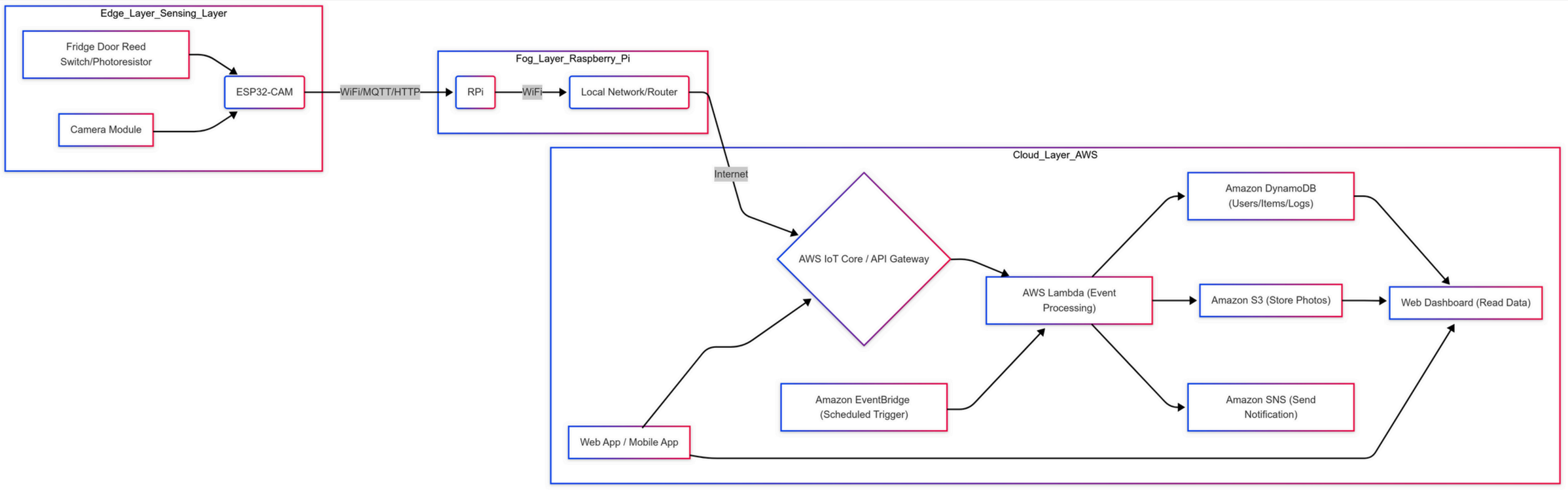
What problems are we solving

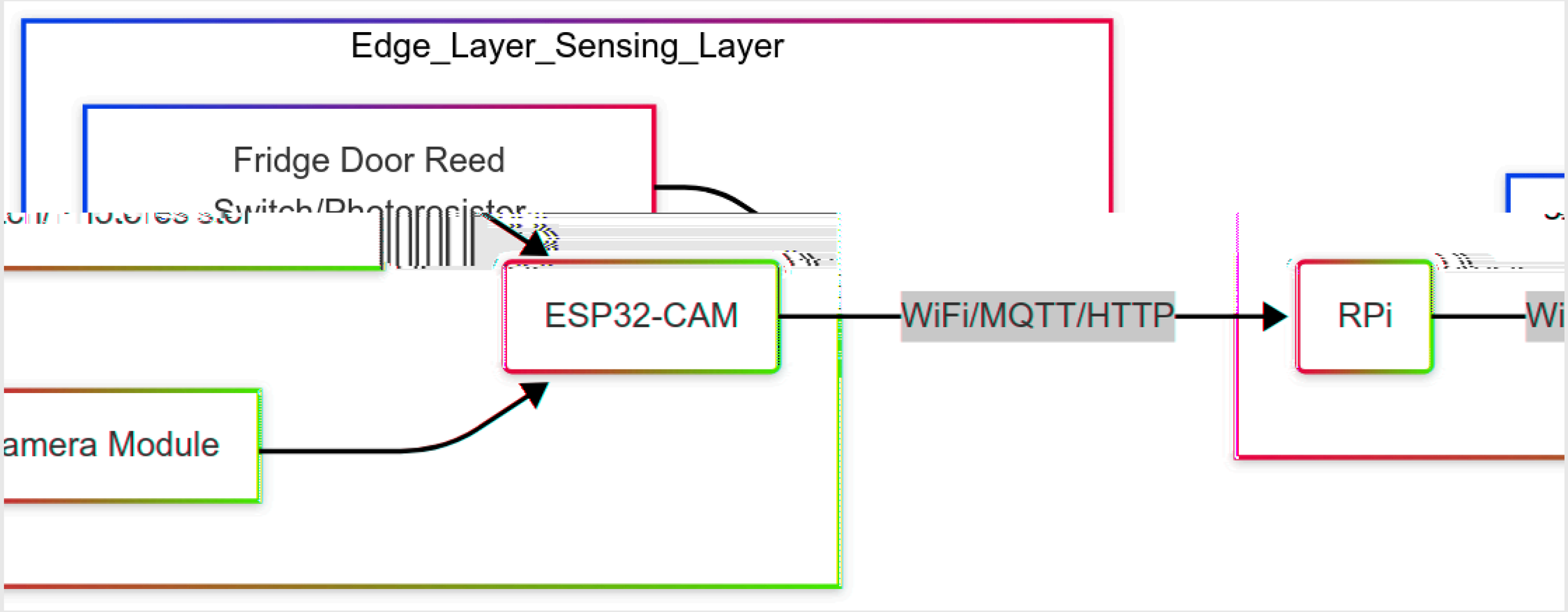
- No more wondering who drank your milk
- No more scary, moldy leftovers hiding in the back
- No more fridge-related arguments
- And most important — making shared living more pleasant and efficient

Full Version Arch

Complete Architectural Blueprint



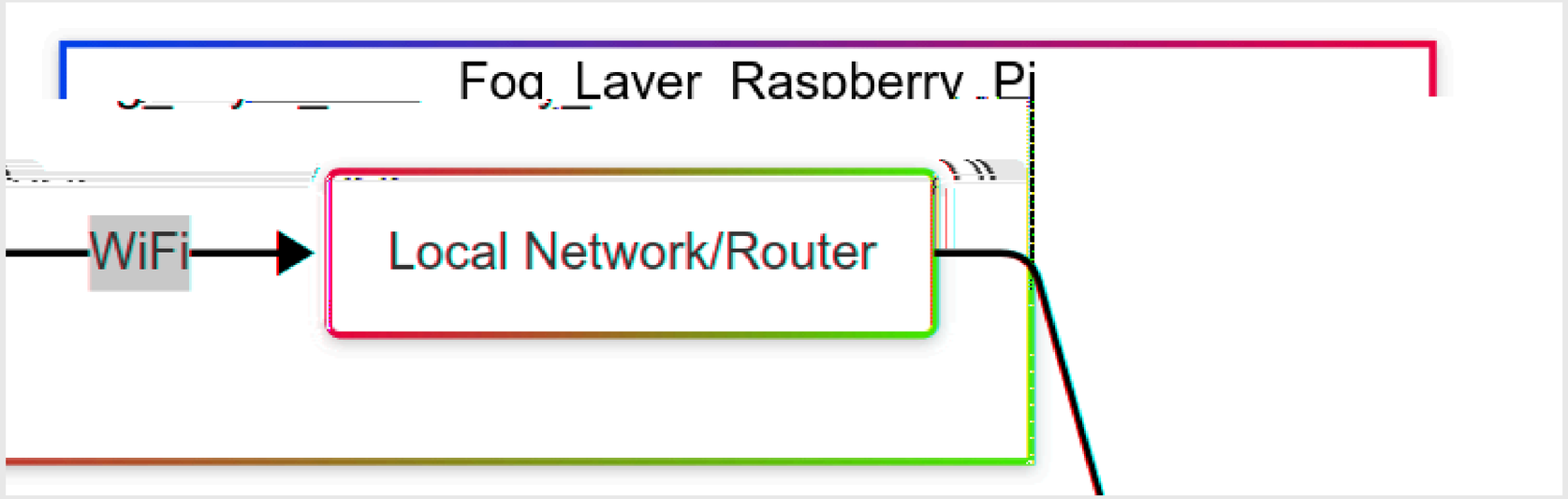

Edge_Layer_Sensing_Layer
Fridge Door Reed Switch/Photoresistor
Camera Module
ESP32-CAM
WiFi/MQTT/HTTP
Fog_Layer_Raspberry_Pi
RPi
WiFi
Local Network/Router
Internet
Cloud_Layer_AWS
AWS IoT Core / API Gateway
AWS Lambda (Event Processing)
Amazon DynamoDB (Users/Items/Logs)
Amazon S3 (Store Photos)
Web Dashboard (Read Data)
Amazon EventBridge (Scheduled Trigger)
Amazon SNS (Send Notification)
Web App / Mobile App


Edge_Layer_Sensing_Layer
Fridge Door Reed
ESP32-CAM
WiFi/MQTT/HTTP
RPi

Edge Layer (Perception Layer)

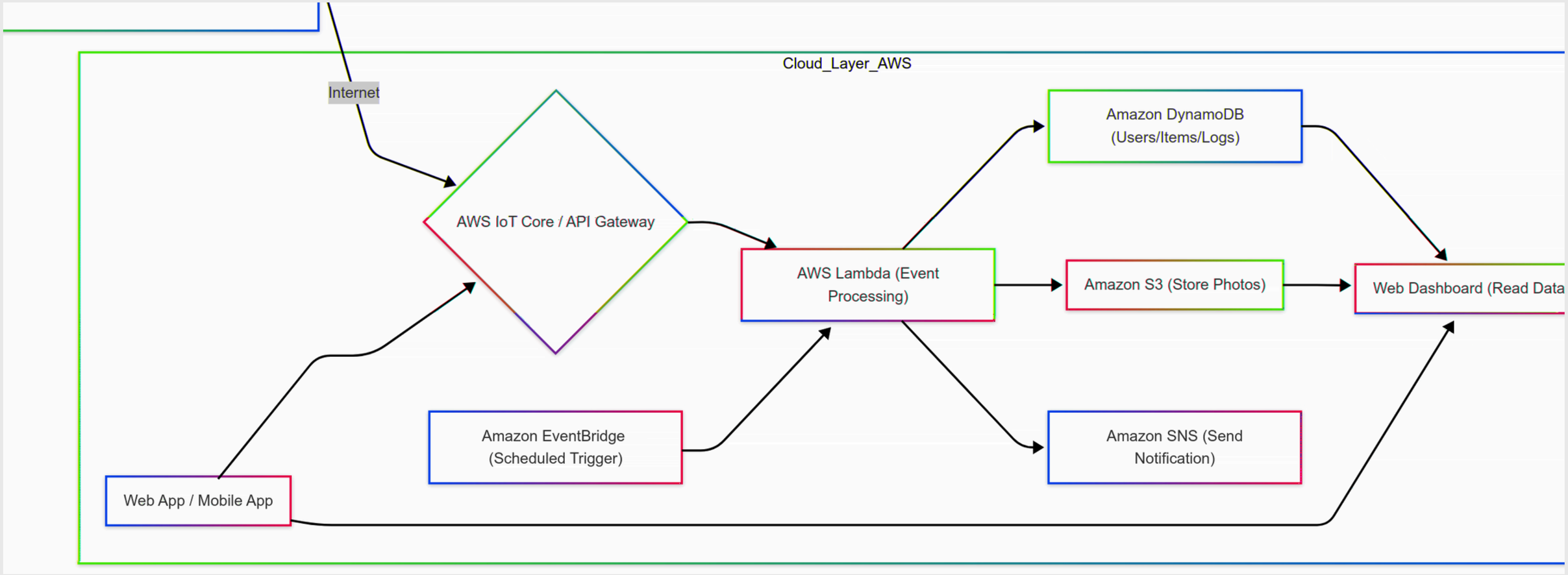
- **Core Device:** A B
- **Sensing Components:**

- **Software:** A
- **Main Functions:**
  -
  -
  -
  -


Fog_Layer_Raspberry_Pi
WiFi
Local Network/Router

Fog Layer (Fog Computing Layer)

- **Hardware:** Raspberry Pi
- **Software:** Python Scripts + MQTT Broker (Mosquitto) / Web Server (Flask/Django)
- **Main Functions:**
  - **Data Relay and Buffering:** Receive data from Edge Layer, buffer during offline periods
  - **Preliminary Processing:** Data validation/format conversion (can be simplified for PoC)
  - **Security Gateway:** Security interface between internal network and cloud
  - **Cloud Upload:** Securely send processed data (events, photos) to the cloud (AWS IoT Core/API Gateway) via AWS SDK (boto3) or MQTT

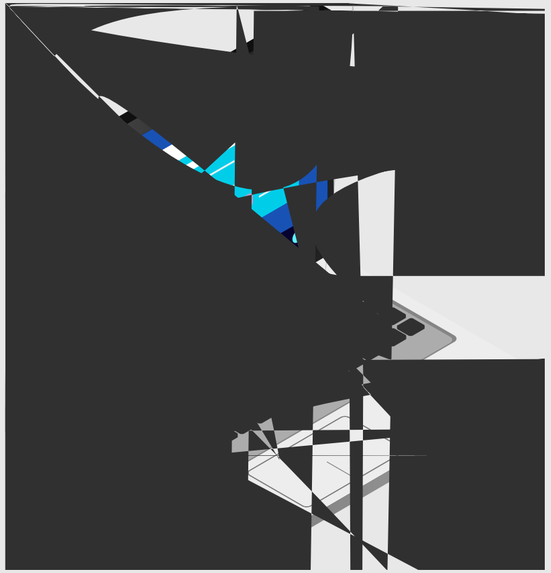

Cloud_Layer_AWS
Internet
AWS IoT Core / API Gateway
AWS Lambda (Event Processing)
Amazon DynamoDB (Users/Items/Logs)
Amazon S3 (Store Photos)
Web Dashboard (Read Data
Amazon SNS (Send Notification)
Amazon EventBridge (Scheduled Trigger)
Web App / Mobile App

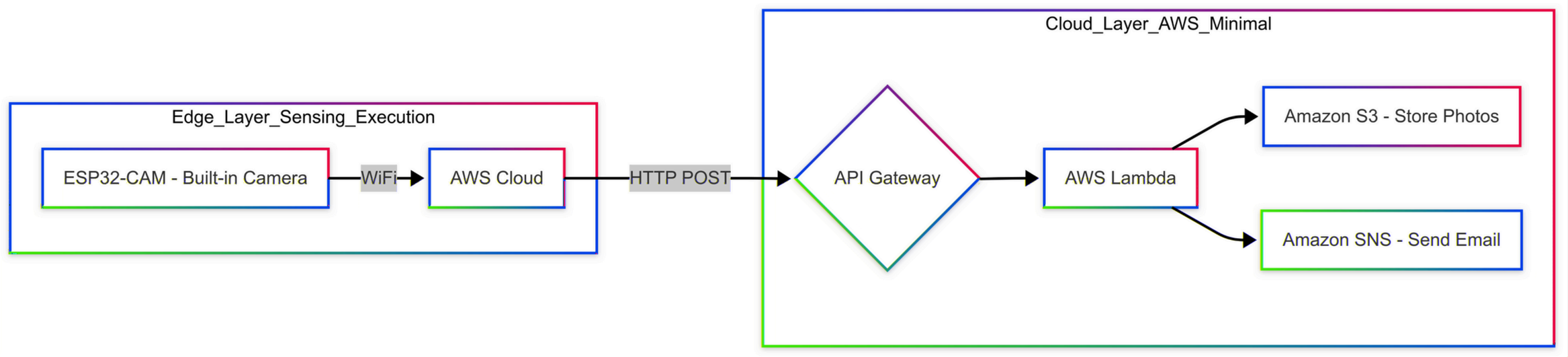
Cloud Layer (AWS) - Core Services

- **AWS IoT Core:** Sec

Cloud Layer (AWS) - Core Services

- **Amazon DynamoDB:** NoSQL database for storing system data
  - **Users Table:** User information
  - **Items Table:** Item information (ID, name, expiry date, status, photo link...)
  - **Events Table:** Door open event records (timestamp, ESP32 ID, photo link...)
- **Amazon S3:** Store photos captured by ESP32-CAM (object storage)
- **Amazon SNS:** Notification service (expiry reminders, suspicious activity alerts)
- **Amazon EventBridge:** Scheduled task triggering (e.g., daily expiry check Lambda)




Edge_Layer_Sensing_Execution
ESP32-CAM - Built-in Camera
WiFi
AWS Cloud
HTTP POST
Cloud_Layer_AWS_Minimal
API Gateway
AWS Lambda
Amazon S3 - Store Photos
Amazon SNS - Send Email

Edge Layer

- **Device:** ESP32-CAM (Integrated Camera, WiFi, Processing) and switch
- **Role:** Performs both data sensing and initial processing at the edge.
- **Functions:**
  - **Trigger Event Detection:** Detects door open events (button).
  - **Image Capture:** Captures photo of refrigerator interior upon event trigger.
  - **Data Packaging:** Packages image data for cloud upload.
  - **Cloud Communication:** Transmits data directly to AWS Cloud via WiFi.
- **Communication:** WiFi, using HTTP POST protocol.
- **Streamlined Edge Processing:** Keeps edge logic simple for this PoC.

Cloud Layer (AWS)

- **AWS Cloud as Backend:** Utilizes essential AWS services for cloud processing and storage.
- **Core Services:**
  - **API Gateway:** Receives HTTP POST requests from ESP32-CAM, entry point to AWS.
  - **AWS Lambda:** Serverless function, processes incoming data, orchestrates actions.
  - **Amazon S3:** Storage for captured refrigerator images (object storage).
  - **Amazon SNS:** Sends email notifications upon door open events.
- **Data Processing Flow:**
  - API Gateway receives image data.
  - Lambda function processes the data.
  - Image saved to S3.
  - Email notification sent via SNS.
- **Serverless Architecture:** Leverages serverless services for scalability and cost-efficiency.

Hardware

- ESP32-CAM: Built-in WiFi + Camera, takes pictures
- Raspberry Pi 4: Fog layer node for processing & forwarding
- Reed Switch: Detects door state
- Jumper Wires: Connects components
- Micro USB Cable: Powers ESP32, uploads code
- USB-C Adapter: Powers Raspberry Pi
- Micro SD Card: For RPi OS & buffer storage
- AWS Services: Free Tier covers API, S3, SNS, Lambda, etc.

Software-Edge Layer

- Hardware: ESP32-CAM + Reed Switch
- Software: MicroPython or Arduino

- **Functions:**
  - Monitor reed switch (door open detection)
  - Trigger camera to capture photo
  - Package photo + timestamp
  - Send data via WiFi (MQTT/HTTP POST)

Software-Fog Layer

- Hardware: Raspberry Pi (3B+ or 4)
- Software: Python script + MQTT broker or Flask/Django server

- **Functions:**
  - Receive & buffer ESP32 data
  - Format/validate payload (optional)
  - Securely forward to AWS (SDK or MQTT)
  - Adds a secure intermediate layer

Software-Cloud Layer

- API Gateway: Receives HTTP requests
- AWS Lambda: Processes image, metadata
- Amazon S3: Stores captured photos
- Amazon SNS: Sends Email/SMS alerts
- DynamoDB: Stores users, items, events
- EventBridge: Triggers expiry checks
- Frontend:
  - Web/App for item registration and dashboard viewing

Why we choose to use these

- MicroPython + ESP32-CAM:
  - Python is relatively simple, ESP32 has WiFi + camera
- Fog + Cloud:
  - RPi acts as buffer + bridge
  - AWS handles logic, alerts, storage
  - Modular design = easier to build & scale
- Serverless (Lambda):
  - No need to run servers
  - Free under AWS Free Tier
  - Easy to plug into other AWS services

Assignment

ESP32-CAM wiring, sensor integration, MicroPython/Arduino programming, image/data packaging, and communication with Raspberry Pi.

Raspberry Pi setup, data buffering/forwarding logic , secure connection to AWS, data upload.

AWS service configuration , Lambda, DynamoDB schema design.

Building a simple web app/dashboard with user registration and data visualization features, and integrating APIs from the backend.

Writing and maintaining GitHub documentation, organizing technical notes, and compiling development progress and reports.

Thank you!