George Gabricht 56735102
Jonathan Huang 22516795
Blanca Lopez-Gonzalez 91245889
Benjamin Nolpho 48299842

SecuriSys: Phase 3

CS190 - Project 2 - Team 6

Github Repo - https://github.com/lonewolfyuri/SecuriSysRepo.git
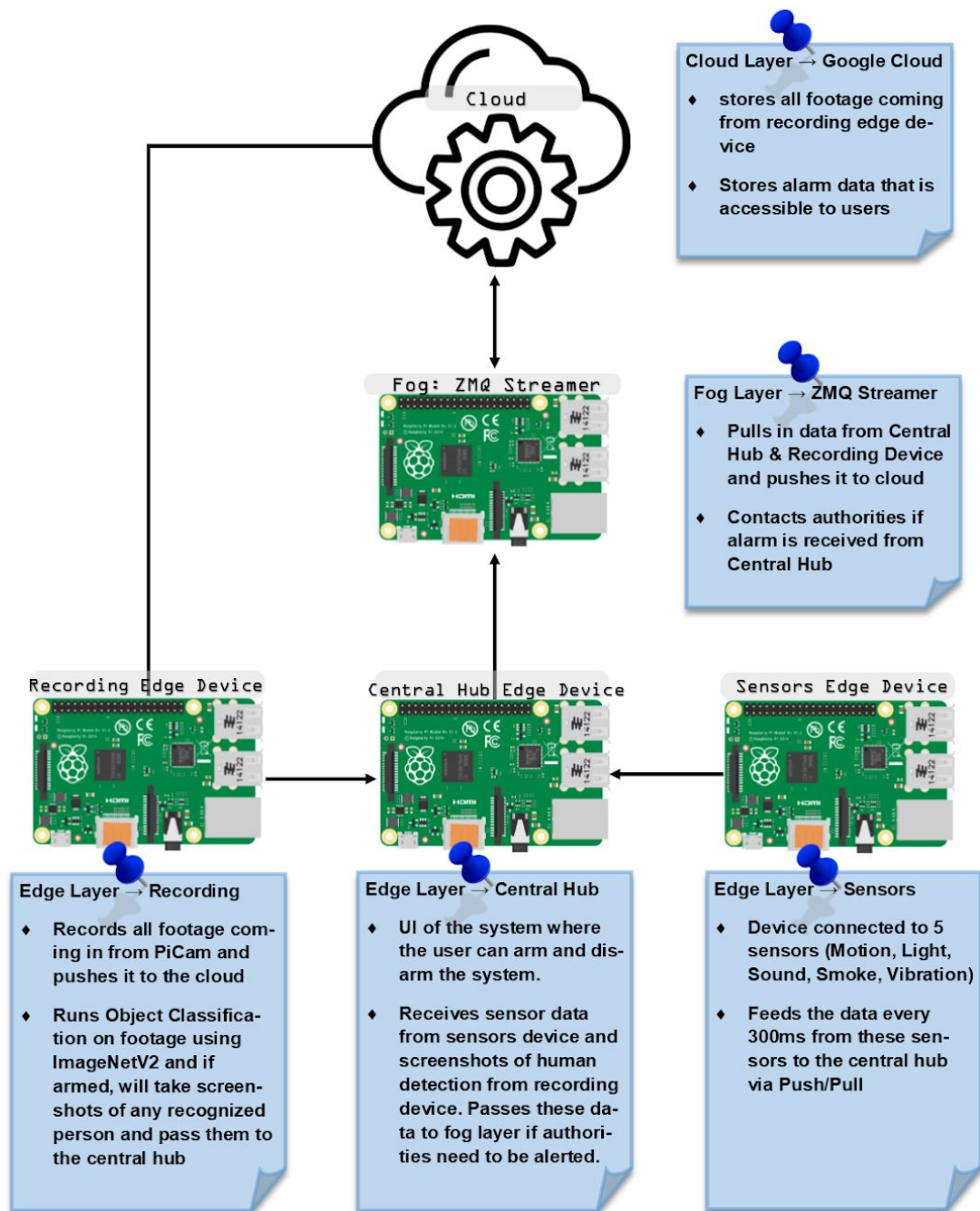
Introduction

Our project is a home security system that uses edge computing named SecuriSys. The system reads in feed from various outputs which include 5 sensors and a live camera feed. These five sensors are motion, smoke, sound, vibration, and light. The camera feed will not only push it's live footage directly to the cloud, but it will also take screenshots of its feed whenever it recognizes a face and pushes the screenshot to the fog. When the system is armed it will alert the appropriate authorities when it is triggered (sensor or facial recognition). All the camera and sensor data are being sent to the cloud (via the fog or directly from the camera edge device).

Optimizations from Phase 2 Feedback

The feedback we received from phase 2 that we decided to implement are as follows:

1. The fog is being overloaded by having it take in so much data (constant video footage+all the data from the central hub)
    a. We are now having the recording edge device push its video feed directly to the cloud instead of the fog to ease how much input the fog is receiving.
2. This system isn't considering the privacy of the user.
    a. We are now having our whole system encrypt messages before sending and decrypt them upon receival to improve security.
3. Our system isn't reliable in the case of a power outage.
    a. Our system now checks if it has connection between devices and if there is no connection between devices for too long, the alarm will go off.
4. More powerful image recognition
    a. TPU was implemented to assist ImageNetV2 Inference
5. Improved Night Vision
    a. During night hours or if the room is too dark, the camera will now have gamma correction with image processing.
6. GUI loading screen
    a. Improved sleek UI

Diagram of the Network:

Cloud Layer → Google Cloud

- stores all footage coming from recording edge device
- Stores alarm data that is accessible to users

Fog: ZMQ Streamer

Fog Layer → ZMQ Streamer

- Pulls in data from Central Hub & Recording Device and pushes it to cloud
- Contacts authorities if alarm is received from Central Hub

Recording Edge Device

Central Hub Edge Device

Sensors Edge Device

Edge Layer → Recording

- Records all footage coming in from PiCam and pushes it to the cloud
- Runs Object Classification on footage using ImageNetV2 and if armed, will take screenshots of any recognized person and pass them to the central hub

Edge Layer → Central Hub

- UI of the system where the user can arm and disarm the system.
- Receives sensor data from sensors device and screenshots of human detection from recording device. Passes these data to fog layer if authorities need to be alerted.

Edge Layer → Sensors

- Device connected to 5 sensors (Motion, Light, Sound, Smoke, Vibration)
- Feeds the data every 300ms from these sensors to the central hub via Push/Pull

Device Catalog:

Device	Hardware Features
Recording Edge Device	Raspberry Pi 3B+, google TPU, Pi-camera
Sensor Edge Device	Raspberry Pi 3B+, breadboard, and 5 sensors (motion, light, sound, smoke, vibration)
Central Hub Edge Device	Raspberry Pi 4, Touch Screen, Speaker
Fog Edge Device	Raspberry Pi 3B+
Google Cloud Platform	Unknown - Simply Implemented API

Report of Network Operation:

➢ **Challenges:**
  ○ Receiving data from other devices
    ■ The way we overcame this is by all connecting to a VPN, sending our IP addresses on the VPN to each other, and using the ZMQ socket library to connect our devices with the IP addresses and ports we shared with each other.
  ○ Adjusting the sensitivity of the sensors to only go off when certain things occur (opening a door, hearing a loud noise)
    ■ The way we overcame this was by mainly trial and error by adjusting the sensitivity directly on the sensors and reading in the data when we wanted our sensors to go off.
  ○ Packaging screenshots + Video feed into a network packet
    ■ Cannot send numpy array that represents an image directly over the network
    ■ We discovered that we first needed to encode our image into bytes, ship the bytes in the network packet payload, then decode the bytes back into the original image on the receiving end
  ○ Managing the vast amount of video+screenshots in the cloud

- ■ The way we overcame this is by sorting the data into buckets as well as putting timestamps on the files being pushed to the cloud

➤ **Services:**
  - ○ Names and description of services you are using
    - ■ Google Cloud Storage API - This API allows us to access and modify storage buckets, pushing various files to these buckets and viewing or even deleting them later.
    - ■ Cryptography - This library allows us to encrypt and decrypt the passcode to keep the passcode secure internally.
    - ■ CV2 - This Library is used throughout all of our image processing and footage creation.
    - ■ Twilio - This API allows us to send text messages through a client to simulate the system contacting authorities in case of emergency.
    - ■ ZMQ library - This Library is used to implement Pub/Sub Behavior in communication and topic filtering throughout our edge system.

➤ **Quality Attributes:**
  - ○ Quality attributes (security, privacy, reliability) you are implementing on your network
    - ■ Reliability by using TCP to connect the fog devices and the edge devices
    - ■ Authentication & Encryption on passwords in Central Hub for arming+disarming

➤ **Messaging Patterns:**
  - ○ Using Pub/Sub to communicate between pis, fog, and the cloud.
    - ■ Surveillance & Sensors publish their data
      - ● Fog Subscribes to Surveillance + Central Hub devices
      - ● Central Hub device subscribes to Sensor + Surveillance devices
  - ○ Google Cloud Storage authenticates the account based on a system path variable, and if the provided credentials are valid, it proceeds to open a client.
    - ■ This client is used to process requests for different buckets, which can be used to push various kinds of data to.