Conceptual Design

E Cubed Group (P2)

Deliverable D	Rares Constantin Serban	Emma Belal
October 13th	Louis-Philippe Keith	

Abstract

In this deliverable, we will be covering the different subsystems of our project. We will also show different design options for the subsystem and explain why we believe the subsystem we chose is the best for the project.

Table of Contents:

Table of Contents: 4
Introduction 5
Problem Statement 5
Subsystem 1 – Rares: 5
Evaluation Subsystem 1 13
Subsystem 2 (Facial recognition)– Louis-Philippe: 14
Evaluation Subsystem 2 16
Subsystem 3 – Emma 16
Option 1 18
Option 2 19
Subsystem 4: Boundary Definition - Rares and Emma 21
Subsystem 5: Notification system – Louis-Philippe and Emma 23
Evaluation Subsystem 5 24

Introduction

Our team has been tasked with building a Net-Aware application using Shabodi's network. Based on design criteria we have developed four subsystems that will form a restricted zone altering application. Considering the user's needs, we will implement a system that will provide real-time and accurate alters to administration. Our goal is to ensure members of the enterprise will not enter administration specified areas. The subsystems include video recognition, card access, GPS location, and boundary definition.

Problem Statement

Shabodi requires a NetAware application that allows for notification and provides location in the event where there's a UE in a restricted area. The system should also be adaptable and ready for use in multiple scenarios.

Subsystem 1 – Rares:

Card-access subsystem: Using Shabodi's Sandbox, we will use a card-access system for our application.

Our hardware requirements will be:

- Raspberry Pi (4 or 5)
- HID OMNIKEY 5427ck Gen 2 (card reader that connects via USB with Raspberry Pi)
- Smart cards or proximity cards compatible with the OMNIKEY reader
- MicroSD card with Raspberry Pi OS installed
- Power supply for the Raspberry Pi
- Wi-Fi setup

Table 1: Hardware Requirements for the Card-Access Subsystem

	Raspberry Pi (4) [16]	HID card-reader [17]	HID Prox cards [18]	MicroSD card [19]
Pros	1. Sustains Python script 2. Community support 3. General Purpose 4. Input/Output (GPIO) pins allow interface with multiple hardware devices. 5. Multiple connectivity options	1. Compatible with HID Global API 2. USB port 3. High security features 4. Multiuse (though we are only going to be using it for access control).	1. Compatible with our HID card reader 2. Cheap	1. Compatible with Raspberry Pi and other devices 2. High reputation 3. Decent storage space 4. High-speed class rating
Cons	1. Too expensive 2. Considerable size (compared to a dedicated chip) 3. Security features (unsecure coding practices)	1. Dependency on specific card types 2. Cost	1. High shipping costs 2. 2-week delivery time.	1. Expensive 2. Delivery period

Our software requirements will be:

- Raspberry Pi OS
- Python script (VSCode)
- HID Global API
- Shabodi's Sandbox
- Database (such as SQLite or MySQL)
- An operating system (Windows PC)

Table 2: Software Requirements for the Card-Access Subsystem

	Raspberry Pi OS	VSCode	HID Global API
Pros	1. Optimized for Raspberry Pi 2. User-friendly 3. Free	1. Extended compatibility (including Python) 2. Built-in terminal 3. Debugging Support 4. Free and open source	1. Compatible with our card access 2. Ease of integration with multiple systems 3. High-security platform
Cons	1. Performance limitations 2. Learning curve for beginners	1. Learning curve 2. Resource intensive (might be an issue on low-end machines or when opening large projects)	1. Unique to HID products 2. Costly

1. **Card reader setup on Raspberry Pi**

Then, we will need to install the required libraries to allow communication between the Raspberry Pi and the card reader:

```bash
sudo apt-get update
sudo apt-get install pcscd pcsc-tools libpcsclite-dev
```

We will use Python scripts to allow communication and user identification (UID) retrieval upon card swipe.

2. Integration with HID Global API

What the API allows us to do:

- Authenticate users based on card UID
- Save access events (who logged, when and where)
- Monitor and control access in real-time

We will need to sign up for the HID Global Developer Portal to obtain the API credentials. This process requires payment.

3. Setup of the Raspberry Pi

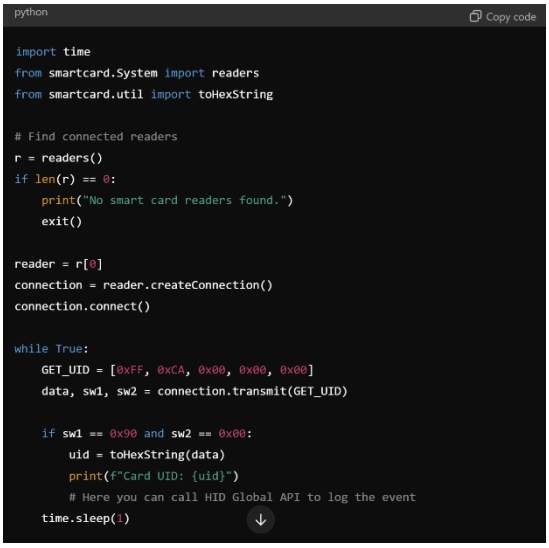
First, we need to install the Raspberry Pi OS. It is free and allows you to connect a Raspberry Pi device to a PC. Then, we will need to pair the card reader with our minicomputer. To achieve this, we connect it via USB port to our computer and install the necessary packages to permit communication with the card reader:

bash Copy code

```
sudo apt-get update
sudo apt-get install pcscd libpcsclite-dev python3-pip
pip3 install pyscard
```

4. Python script to capture card swipe

For smart card communication, card reader interaction and UID retrieval, we will use the PySCARD Python library. We can find examples of Python scripts online and/or with the help of AI.



python Copy code

```python
import time
from smartcard.System import readers
from smartcard.util import toHexString

# Find connected readers
r = readers()
if len(r) == 0:
    print("No smart card readers found.")
    exit()

reader = r[0]
connection = reader.createConnection()
connection.connect()

while True:
    GET_UID = [0xFF, 0xCA, 0x00, 0x00, 0x00]
    data, sw1, sw2 = connection.transmit(GET_UID)

    if sw1 == 0x90 and sw2 == 0x00:
        uid = toHexString(data)
        print(f"Card UID: {uid}")
        # Here you can call HID Global API to log the event
    time.sleep(1)
```

5. Authentication and log events with HID Global API

Upon registering HID Global Developer Portal, we must retrieve an API key or access token (under paid license). Then we must use this key in our HTTP headers for authentication when requesting information to the API.

When a card is swiped and the UID is captured, let us send a request to HID Global API to associate and retrieve the corresponding information of the user's details (name, access permissions, etc.) Here again, we can find more information about this code online. Here is an example:

```python
import requests

api_key = "your_api_key"
card_uid = "captured_card_uid"  # Replace with the actual UID captured

url = f"https://api.hidglobal.com/access/credentials/{card_uid}"

headers = {
    'Authorization': f'Bearer {api_key}',
    'Content-Type': 'application/json'
}

response = requests.get(url, headers=headers)

if response.status_code == 200:
    user_info = response.json()
    print(f"User Info: {user_info}")
else:
    print(f"Failed to get user info: {response.status_code}")
```

6. Logging of access events

Using Shabodi's Sandbox, we must create a database containing all user information that will be linked using the API to a card UID. We can also use the HID Global API to store event information and study or analyze it later. The database format will be the following:

```sql
CREATE TABLE access_logs (
    log_id          INTEGER PRIMARY KEY AUTOINCREMENT,
    user_name       TEXT,
    card_uid        TEXT,
    door            TEXT,
    access_time     TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status          TEXT
);
```

7. **Display data in NetAware application**

With all this preliminary information, we can now start displaying it on our NetAware application. With Shabodi's Sandbox, we can create a simple platform that illustrates this information on a simple front-end dashboard. It should include who accessed which door, the timestamp of each access event, and any unauthorized access attempts.

8. **Monitor and maintain**

Next, using HID Global API, we can monitor access in real-time and push notifications in the event of unauthorized access or system failure. For this, we will need to set up a logging and alerting system that can notify customers via text and/or email. We can also upload information to the Cloud, to which we will need access.

9. In summary

Figure 1: Summary of Subsystem 1

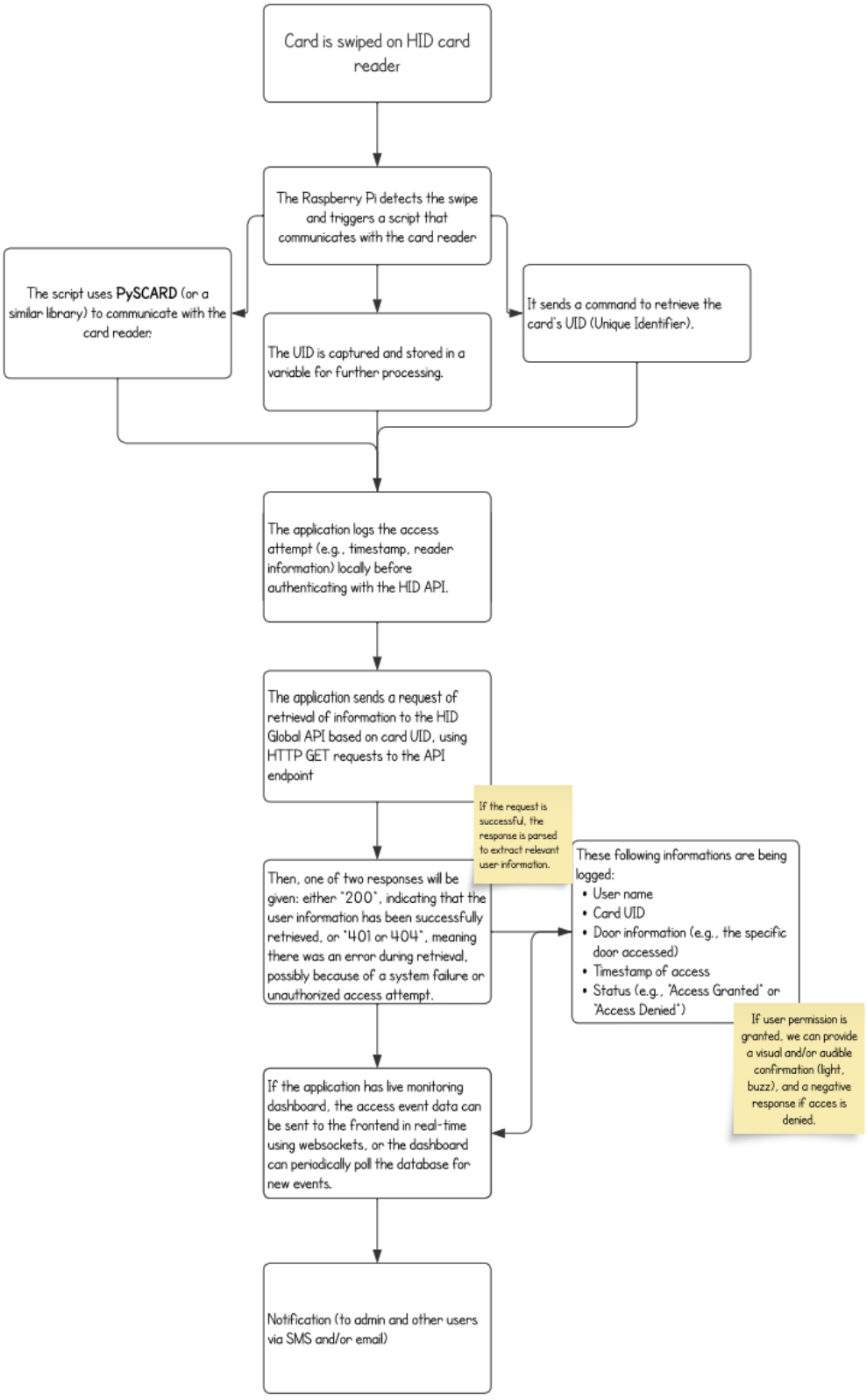


Table 3: API Options:

	libnfc	AWS IoT Core	HID Global API
Pros	1. Open source (free) 2. Operates offline	1. Provides real-time monitoring 2. Wireless device management (with Raspberry Pi and other devices) 3. Works well with AWS services to integrate data into Cloud	1. Works with our card reader 2. Elevated level of security 3. Enterprise-Level Features (offers features like multi-site management, detailed access logging, and integration with various HID products)
Cons	1. Limitations with compatibility (need to buy card reader) 2. Lacks security features 3. Requires more development efforts	1. Requires constant internet access, or else the system is down 2. Gets pricey 3. Security is up to us to integrate (added complexity)	1. Tied to HID ecosystem 2. Expensive 3. Limited range of compatibility 4. Complex (designed for enterprises)

Given the fact that we have an HID card reader, we will go ahead with the HID Global API. If the license is more expensive than our budget, we will pay the difference out of our own pockets, and its use will be limited for demonstration purposes exclusively.

Evaluation Subsystem 1

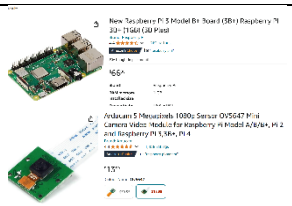
Overall, the main concern about this subsystem is its cost. It will exceed our budget, and we will need to accommodate the cost ourselves. Nevertheless, it is a high security diverse and expandable platform. The Raspberry Pi can be used to link other subsystems too, such as the facial ID that Louis is doing. The card reader has already been provided to us, so there is no added cost there. It is manufactured by a reputable company known for its high security and accessibility standards. The HID swipe cards are unexpensive and work seamlessly with our card reader. Lastly, for our hardware requirements, a micro-SD card is mandatory for Raspberry Pi operation. The one we found gives excellent value for the price, offering 256gb of storage for $32.

For our software requirements, we had to choose the API we wanted to use. Although it is not free, HID Global API provides a strong interface connection between our card reader and the user information we want to retrieve. It offers interesting features such as multi-site management, detailed access logging, and integration with various HID products. We can also monitor access in real-time and push notifications in the event of unauthorized access or system failure. Additionally, the entire system is easy to upgrade. You are free to use more card readers in various locations if you desire to do so, you can even use your phone to swipe in front of the card reader (not with the one we are using). The system is multi-use and utilitarian in design, it is quite easy to swap in-and-out features. The main drawback of the other free APIs is the lack of connectivity with our card reader and the poor security encryptions. As for our other software requirements, both Raspberry Pi OS and VSCode are free and essential for our system to work.

Finally, following these steps, we can build a utilitarian, high security and easy to manage interface. Though the license to HID Global API is expensive, we can arrange that cost by ourselves and still produce a demonstratable product.

Subsystem 2 (Facial recognition)– Louis-Philippe:

Table 4: Hardware for Facial Recognition

Potential Hardware option when building AI camera		
	Facial Recognition on used computer.	Facial recognition on Raspberry Pi camera
Hardware		
Software	Windows that can run python.	Uses python and has no pre-existing software
Cost	0 (used)	79.99$ before tax
Advantages	Since I am coding the program on this computer, I will not have to download it on another device. This is also the cheapest option as I would not have to spend any money.	Uses python, has a large community support and many people have already used Raspberry Pi 3 for facial recognition. Since the raspberry Pi is small it would be a lot more representative of a real camera.
Disadvantages	My computer has better hardware than most smart cameras, so it is less representative of the real application.	Expensive, could not have enough Ram for good facial recognition software.

Since our Budget for this project is 25$ we decided to go with the first option of using my computer for the software.

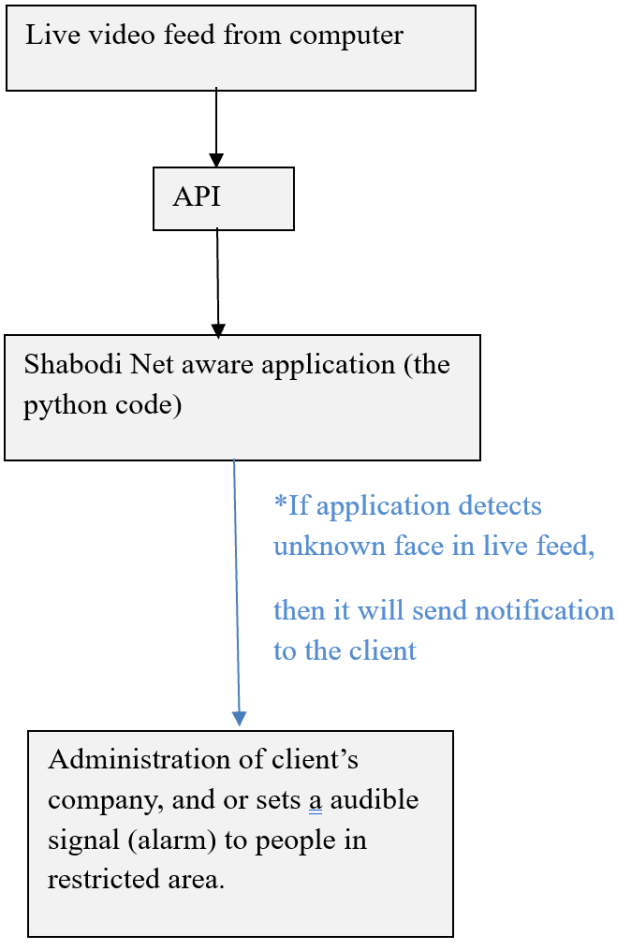
Software

Table 5: APIs for Facial Recognition

	OpenCV	OpenFace	FaceNet
Advantages	OpenCV is a very general API that has a lot of online support for building applications.	OpenFace is more accurate than OpenCV but lacks online support when it comes to coding the API.	FaceNet was developed by researchers at Google and is the most accurate and complex of all the other API's and has a moderate amount of support online.
Disadvantages	Since OpenCV is the most general out of all the APIs it makes it less accurate when it comes to detection and recognition.	Since it does not have a strong online community, and it is more complex, it makes it hard to implement in a student project.	Since the API is the most complex it makes it hard to work with, especially for someone who is new to coding.

This system will use OpenCV API for this because it has a lot of support online and it is easy to find resources for creating a face recognition program. It is important to our group that we have a functioning product on design day, but we are all new to coding in python. Working with more complex APIs would take too much time to develop before design day. In addition, one of the criteria of our system is it must be adaptable and OpenCV has other functions that can be implemented or removed depending on the client.

Figure 2: Face Recognition System



Evaluation Subsystem 2

In Subsystem 2, we will code a face recognition system coded on python using OpenCV. When the program does not recognize someone, it will send a notification to whoever the client wants to.

Subsystem 3 – Emma

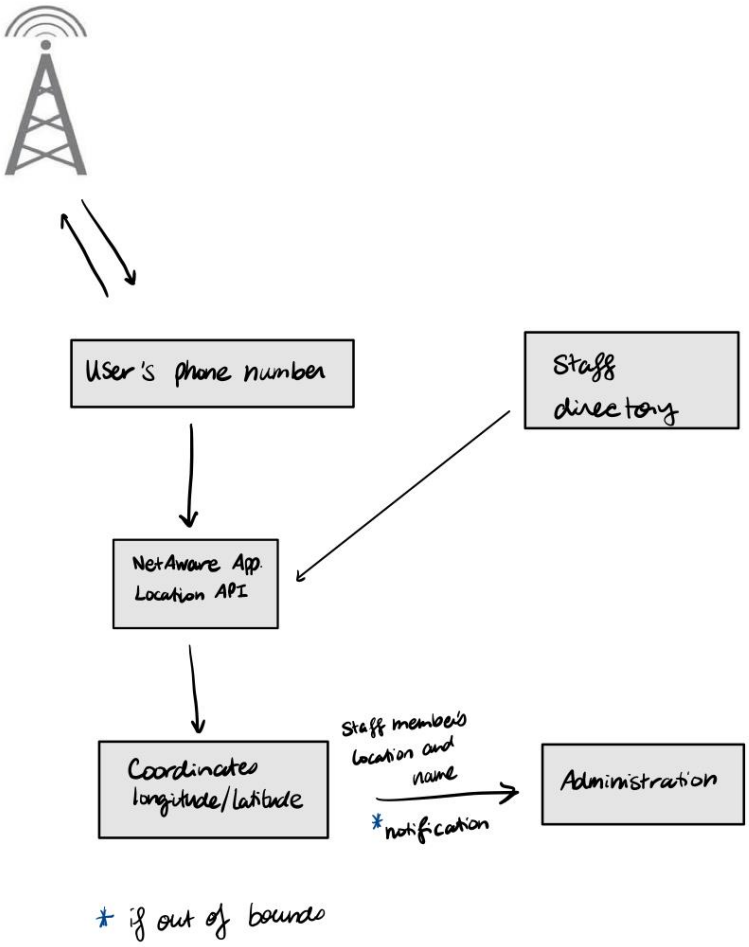
Both systems will leverage the same location API. The location API will decide if a phone number tracking or Global Positioning System (GPS) would be functional options. To note is that APIs will determine accuracy of location.

Table 6: Location Tracking APIs

	Google Geolocation [2]	HERE [3]	OpenLayers [4]	Distancematrix [5]
Pros	1. It has the most developed map, multiple features beyond longitude and latitude, ex: route tracking. 2. Can be modified to work indoors and outdoors.	1. Works indoors and outdoors. 2. Can build interactive maps. 3. Can return longitude and latitude. 4. Can create maps for autonomous vehicles	1. Location update time can be modified. 2. Can return the longitude and latitude of the device.	1. It has developed maps, comparable to Google 2. Can return to longitude and latitude of locations 3. Variety of language options to receive information
Cons	1. Only updates if the location has significantly changed or after a few minutes. [1]	1. Code built for JavaScript (would need to be bridged).	1. Code built for JavaScript (would need to be bridged). Works best outdoors only	1. Mostly used for routes, and best suited for addresses.
Cost	1. Depending on the number of alerts the price can increase.	1. Depending on the number of alerts the price can increase.	1. Open source – no additional cost.	1. Low cost, where pricing is scalable.

Option 1

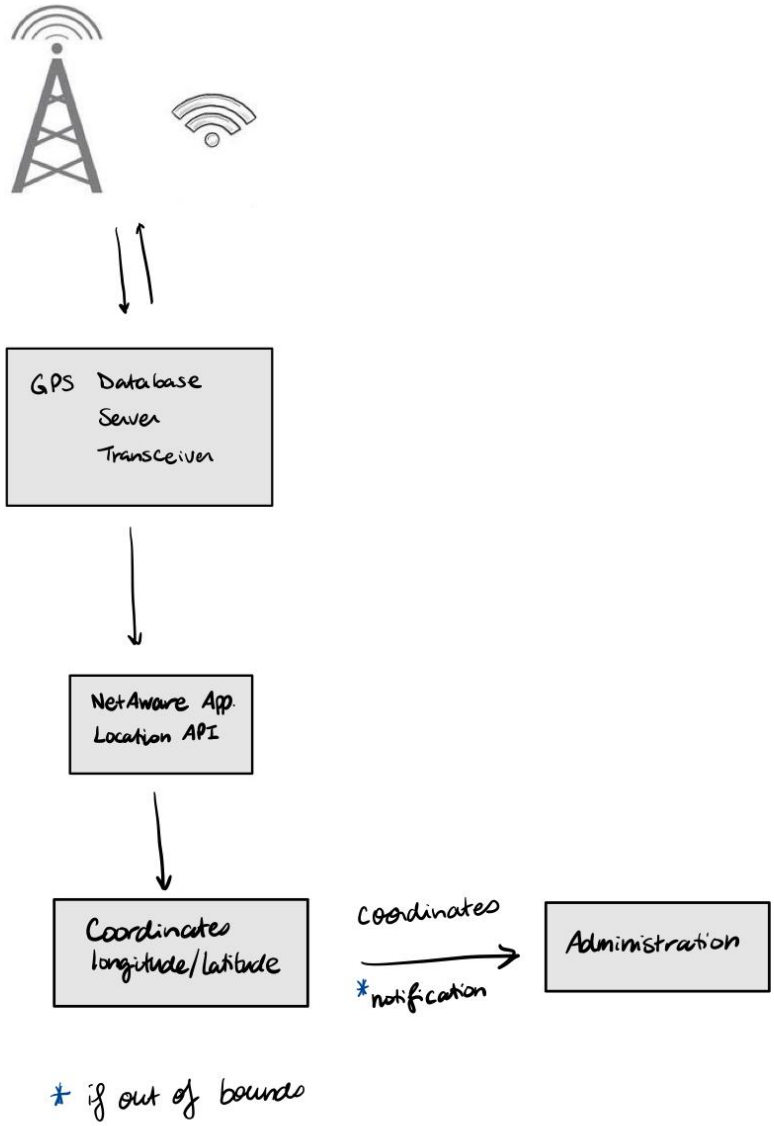
Figure 3: Phone Number-Based System



This system uses the staff members' phone numbers to track where they are within an enterprise's given boundary. It relies on open networks to gather information. The NetAware application will leverage an API to gather information based on location and then send it to the administration. The location of the phone may not be accurate. If the cellular device enters a restricted area the administration will be notified of the staff members' name and location. The notification will provide a visual and audible alert to ensure recognition.

Option 2

Figure 4: GPS Based System



This system will use a GPS tracking device to gather information. Once the information has been received by the NetAware application, a location API will process the location and route of the device. If the device enters a restricted area a notification will be sent to administration. The notification will provide a visual and audible alert to ensure recognition

Table 7: Evaluation of Hardware

	Phone	Raspberry Pi	GPS System
Pros	1. Accessible since everyone uses a phone for communication. 2. Provide accurate location. 3. Would be indoors and outdoors.	1. Supports python 2. Preforms well in multiple use cases 3. Can simulate many different subsystems. 4. Accessible since it is used in another subsystem	1. Works well on vehicle tracking. 2. Optimal for outdoor use.
Cons	1. It would rely on phone battery. 2. Must be provided by the employer to have correct tracking application installed. 3. Does not rely on a private network	1. May not be as accurate as phone or GPS. 2. Bulky item – not representative of how an enterprise would leverage these features	1. Code depends on the device. 2. Not suitable for indoor use.

When comparing between the options in subsystem design and hardware necessities. The best subsystem choice would be to use the **(API)** with phone tracking. Since the phone is already available to us, removing any aspect of cost. The **(API)** will be able to provide the most accurate map available, and the phone will provide adequate location accuracy. When this information is coupled with a notification API, the administration will have real-time alerts of unauthorized entrance events.

Requirements:

- Cellular Phone
- Python script (VSCode)
- Geolocation API
- Shabodi's Sandbox
- Database (Enterprise Directory)
- An operating system

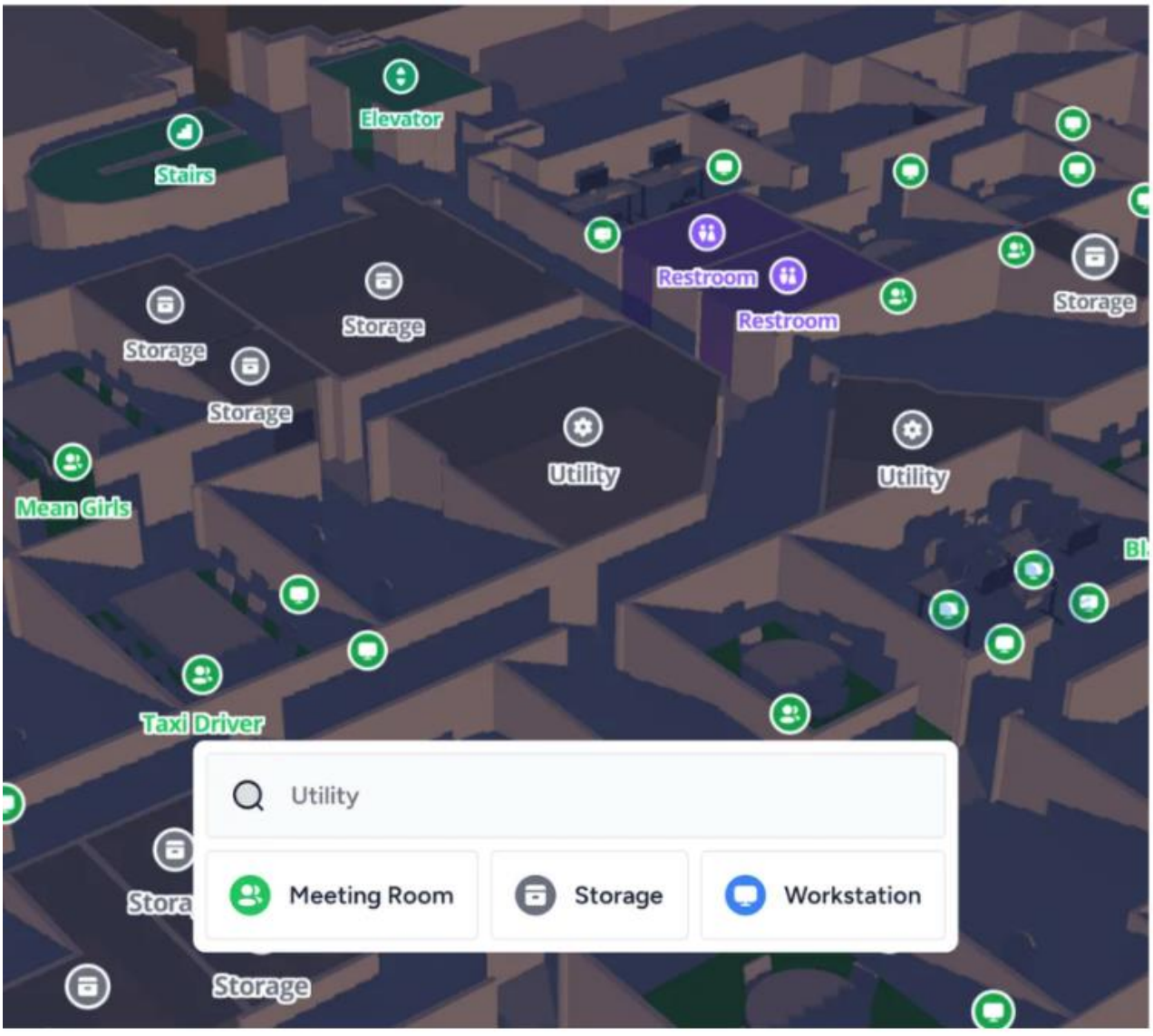
Subsystem 4: Boundary Definition - Rares and Emma

A significant portion of restricted zone alerting is being able to define boundaries. The administration needs to be able to fluidly define the permittable and non-permittable zones. Certain APIs offer diverse ways to define an area.

Option 1: MapsPeople

This application utilizes many of their internal APIs to create 3D maps of buildings, roads, and rooms. Targeted towards enterprises being able to label locations and update information as needed. [8]

Figure 4: MapsPeople Labeling Location Example



Option 2: OpenCage API

This API allows users to define a radius of acceptable zoning. It leverages a map-based system, which allows us to see beyond the enterprises' borders.

Figure 5: OpenCage map (where the red zone is the authorized/unauthorized zone)

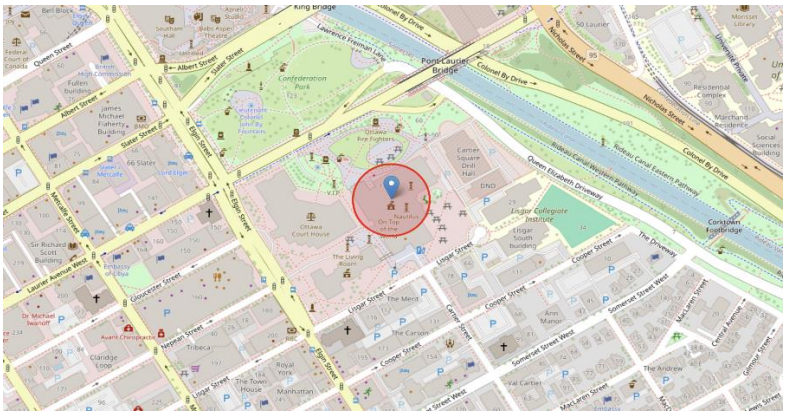


Table 8: Comparisons Between Options

	MapsPeople	OpenCage
Pros	1. Can map indoor and outdoor areas 2. User friendly 3. Clear visuals	1. Supports python 2. Able to redefine areas 3. Useful for restricting areas external to the enterprise.
Cons	An application and not API, can restrict the ability to modify for different enterprises	Not as accurate or as clear as MapsPeople
Cost	High Cost	Lower cost, has free trial options

Due to budgeting restrictions implementing the OpenCage API will be the best choice for this project. The enterprises will be able to implement the restricted zones and change them if necessary. The boundaries will guide the rest of the subsystems, allowing for real-time notification of unauthorized entrance into space. OpenCage API can also be modified to include other systems (at a cost) to improve map accuracy and tailor the functionality to the enterprise's needs.

Subsystem 5: Notification system – Louis-Philippe and Emma

Figure 6: Notification Block Diagram

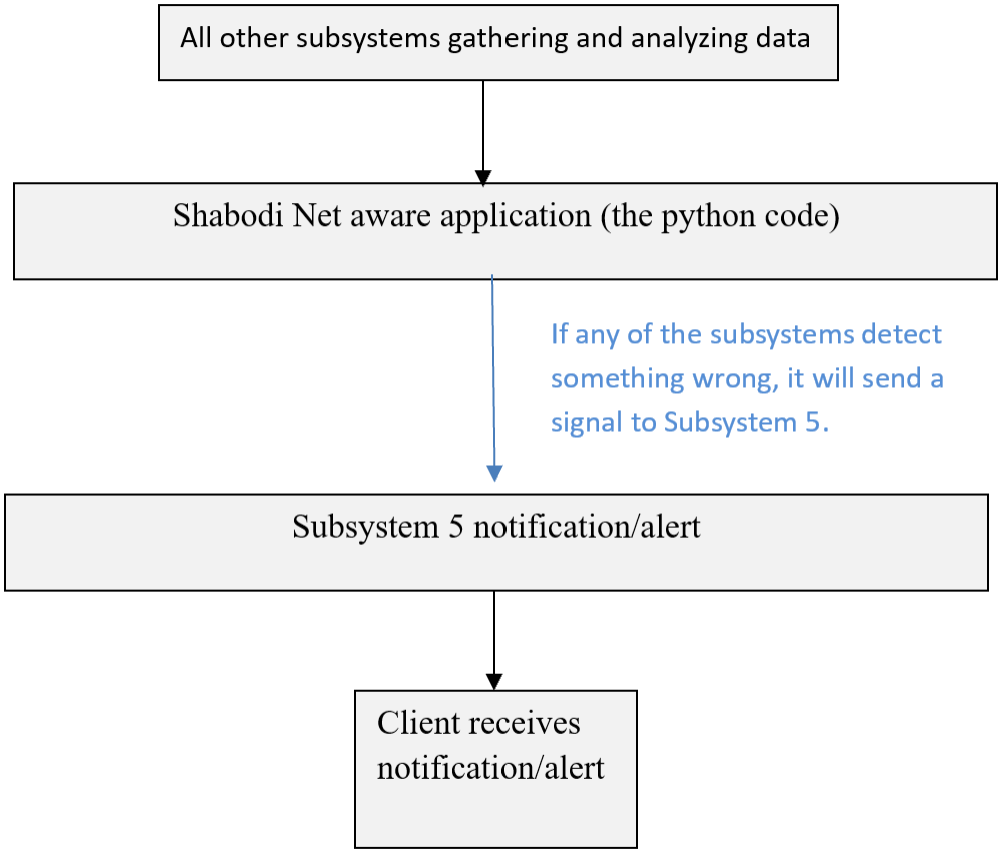


Table 9: Notification APIs

	Pushbullet	Pushover.net	NotificationAPI.com [9]
Pros	Push bullet uses encryption and is easy to use on python.	Can send up to 10,000 notifications per month, which should be the reason for most companies	can send up to 10,000 notifications, the API is simple so it should be easier to integrate with aps.
Cons	Limited free plan can only get 500 notifications a month.	Cost 5$ which is a lot for our team's budget of 25$, has less multiplatform capability than Pushbullet	Less online support compared to pushbullet and Pushover.net.
Cost	Free	5$ per month	Free

Subsystem 5 will be using a notification API (notificationapi.com) to send out an email, SMS, or another alert type to notify whomever the client wants. We will be using notification API because even though it has less community behind it, its simple nature might come in handy when working with Shabodi sandbox.

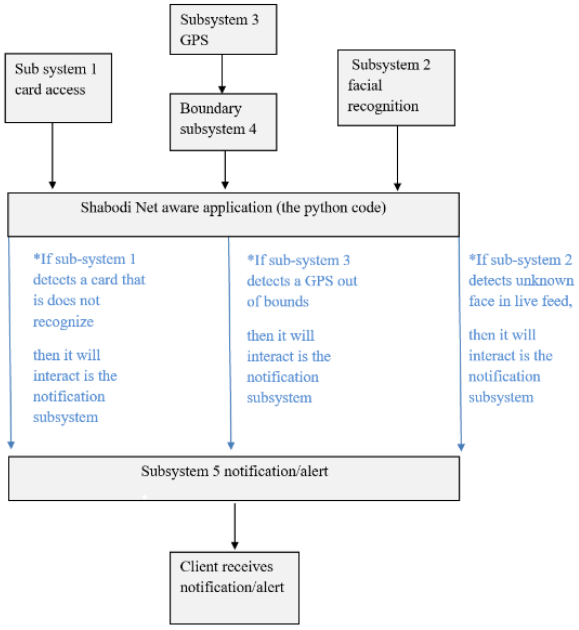
Evaluation Subsystem 5

Subsystem 5 does not require any hardware as it entirely relies on other API's and Shabodi's sandbox. Since the system is already complex enough, we chose NotificationAPI.com to send out notifications to alert users when people are going to restricted areas. NotificationAPI will allow us to send visual alerts to the administrator.

Evaluation:

The final solution will leverage all subsystems to give the most information to administration as possible. The basis of the restricted zone alerting system will be the boundary definition, GPS tracking, and notifications. These Three subsystems will allow the administration to have the essential information needed to track unauthorized access. In addition, card access and face recognition will provide the administration with extra security when needed. Which can help improve security and alert detection.

From subsystem 4 we can get the outline of the process:



For the notification subsystem we will use notificationapi.com For GPS tracking we will use Google Geolocation API with boundary definition by OpenCage since it is open source. The three previous subsystems will allow us to use card access and facial recognition. For card access we will use HID Global API for facial recognition, we will use OpenCV as it is the easiest to use and has the biggest online community.

Conclusion:

We have gathered the necessary information from brainstorming to choose a system that will be beneficial for any enterprise's needs. While making our choices we considered all aspects of our gathered information. We tried to choose the options that were the most accessible and which allow for future development. The subsystems presented can work together in multiple ways. For our next steps we will focus on integrating our ideas using Shabodi's sandbox and their developer tools.

References:

[1] “How Accurate Is Google Timeline for Mileage Tracking?” Timeero, Timeero - How Accurate is Google Timeline for Mileage Tracking?. Accessed 12 Oct. 2024.

[2] *Google*, Google, Geolocation API overview | Google for Developers. Accessed 12 Oct. 2024.

[3] “Here Maps API for JavaScript: Build Feature-Rich Web Apps: Here.” *HERE Developer*, https://developer.here.com/develop/javascript-api. Accessed 12 Oct. 2024.

[4] Geolocation, Geolocation (openlayers.org). Accessed 12 Oct. 2024.

[5] AI, Twine, et al. “Top 10 Open Source Facial Recognition Libraries and Tools.” *Twine Blog*, 22 Nov. 2023, https://www.twine.net/blog/top-10-open-source-facial-recognition-library-and-tools/.

[6] “IPhone 16: Bell Mobility: Bell Canada.” *Www.Bell.Ca*, www.bell.ca/Mobility/Products/iPhone-16 . Accessed 12 Oct. 2024.

[8] “Indoor Mapping for Smart Building Products.” *MapsPeople*, www.mapspeople.com/?utm_term=indoor+mapping+api&utm_campaign=NA%2B-%2BIndoor%2BMapping&utm_source=adwords&utm_medium=ppc&hsa_acc=1756747029&hsa_cam=18137476208&hsa_grp=141829672793&hsa_ad=688689425503&hsa_src=g&hsa_tgt=kwd-344801684785&hsa_kw=indoor+mapping+api&hsa_mt=p&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=CjwKCAjwmaO4BhAhEiwA5p4YL9PSggfUwdeFqBgIrgzqfStJCf_4bMQVslfKtHOg-0IoQh6pJwTSkRoCZ04QAvD_BwE . Accessed 12 Oct. 2024.

[9] “Notificationapi.” *NotificationAPI*, www.notificationapi.com/ . Accessed 12 Oct. 2024.

[10] Qureshi, Asim. “10 Best GPS Module for Raspberry Pi.” *Wonderful Engineering*, 31 July 2019, 10 Best GPS Module For Raspberry Pi (wonderfulengineering.com).

[11] *Raspberry Pi OS – Raspberry Pi*, www.raspberrypi.com/software/ . Accessed 13 Oct. 2024.

[12] “OpenCage Geocoding API Documentation.” *OpenCage Geocoding API Documentation (opencagedata.com)*. Accessed 12 Oct. 2024.

[13] Nfc-Tools. “NFC-Tools/LIBNFC: Platform Independent near Field Communication (NFC) Library.” GitHub - nfc-tools/libnfc: Platform independent Near Field Communication (NFC) library. Accessed 12 Oct. 2024.

[14] *Secure IOT Gateway, IOT Gateway Device - AWS IOT Core - AWS*, Secure IoT Gateway, IoT Gateway Device - AWS IoT Core - AWS (amazon.com). Accessed 13 Oct. 2024.

[15] “Opin API.” OPIN API | HID SDK, OPIN API | HID SDK (hidglobal.com). Accessed 12 Oct. 2024.

[16] “Raspberry SC15184 PI 4 Model B 2019 Quad Core 64 Bit WIFI Bluetooth (2GB): Amazon.ca: Electronics.” *Raspberry SC15184 Pi 4 Model B 2019 Quad Core 64 Bit WiFi Bluetooth (2GB) : Amazon.ca: Electronics*, www.amazon.ca/Raspberry-Model-2019-Quad-Bluetooth/dp/B07TD42S27/ref=sr_1_6?adgrpid=1364494457248639&dib=eyJ2IjoiMSJ9.BZmapo9ErmNt1ZrvSlNDRB6--RiRzs-pjKdtN28bw4ufuoLJDrgfYOeSCvSlT-ITVYGdUZ2yfkwgyJIBzU4HF6qeO6ZctzZpXQuuzbAxwvILAnao0kr3qQPS7wrtUy1CeA8aBJHj6auFHMpoNu7vKZATpXLw4L226hHybDiDDWpTFYRkFjpWKtWiVyg_iMuDaLFAHDlvA34WCLXj0WI9DHFd8CZu35_IKtA_YkpVrsi21mYRgcKaceJxBUXnmp88cszz0682zqbbT-7FQcy6gdlvvz9IrVR6qLwAQ8sZuC4.RvOFIxOvz9nYOwC272F3GYAIHOg3s1NnKJaOd8O3cF8&dib_tag=se&hvadid=85281424927189&hvbmt=be&hvdev=c&hvlocphy=124909&hvnetw=o&hvqmt=e&hvtargid=kwd-85281272966503%3Aloc-32&hydadcr=24945_13712585&keywords=raspberry%2Bpi%2Bamazon&msclkid=ea6dc070e9361c4e6f7fa77bed445293&qid=1728775283&sr=8-6&th=1. Accessed 12 Oct. 2024.

[17] “HID OMNIKEY 5427ck Gen 2 - Smart Card Reader - USB, Black, Light Gray.” *Amazon.ca: Electronics*, www.amazon.ca/GLOBAL-R54270101-OMNIKEY-Interface-Reader/dp/B07FD7V6KB/ref=sr_1_6?crid=53CJKKFYUBPR&dib=eyJ2IjoiMSJ9.0t3gPXSUALv7S_NjCdsJ0tvGRiHyZK3_JQ1caJ-9O0OaYe2pcAi114Q_sVJ_0MiTYuoGShotSg8jA_Ksg5kDp8mNpsxp7QYLAt73vJ5POYet2GaO1H_NdUrLvm_M7Bm-ttRcITajtZdSBGXI7mSkOdDdiv-6S5WG0Q-XclOc3Q6_F2VqSax-ScrrN9rsjaE2lOgRbwpFivyyJzrITk4JKk-sYdqtiTW324D4UQEJqsxr0uZP09ZndNI0J1LPl3FbMVxy-qp7yUXlY86Z3PbUX2T9s9psD_EriSgantt9POc.adcluaZfl77hVf5QS6QnKRhEV4lwbzT70U59ZJtY6Go&dib_tag=se&keywords=hid%2Bomnikey%2Bcard%2Breader&qid=1728775459&sprefix=hid%2Bomnikey%2Bcard%2Breader%2Caps%2C85&sr=8-6. Accessed 12 Oct. 2024.

[18] “25 New Hid ProxCard II 1326 Clamshell Access Control Security Card 1326LMSMV 37.” *eBay*, www.ebay.ca/itm/146041639295?_skw=hid%2Bprox%2Bcard&epid=2124766986&itmmeta=01JA1GGECARESRY98X37555D2S&hash=item2200c2917f%3Ag%3A0lAAAOSwc0Fm7eKV&itmprp=enc%3AAQAJAAAA0HoV3kP08IDx%2BKZ9MfhVJKlKuHjmI8keJqX9%2BDhAWN5Ep7lOHHewGaYXTe%2FXNUVSxdpBJOByjfC%2BtwMY31X2xwPIDJX2rtFoId10CBmuMAuKK8AANV6uv0lT2NTcF1P%2FUgPCOgbVJPPTjILeqPWk8%2FjE2X4blQB3Vd8Gg81JAgqiyIqL

3MbTdL9ZIQ3HNCtvpF%2FBUIjbdgvRUlVA4QWMQZdFurDccy594srGJy4nyD7357H4JXe4%2FJfv%2BkTec58Bz%2F8UsIZVlfdsCxSJhCWwqF0%3D%7Ctkp%3ABk9SR6DmwbDQZA. Accessed 12 Oct. 2024.

[19] “SanDisk 256GB Extreme Microsdxc UHS-I Memory Card - C10, U3, V30, 4K, A2, Micro SD - SDSQXA1-256G-GN6MN.” *Amazon.ca: Electronics*, www.amazon.ca/SanDisk-256GB-Extreme-MicroSDXC-Memory/dp/B082WNC4NK/ref=sr_1_18?crid=24ONCUYOFOTCX&dib=eyJ2IjoiMSJ9._nqLdYHlrAMIJc4YNuvOfw2kFn9qdRTLSSCyGvHQzHJyg8ucKnxi6TXeNKYIf4ZzGDoePpFcNHqu8B9FDsnqL2iXCUxk2xED5oVll4JmiPGFKdft-yPG7QDwxjUsawzYFCx4jr8ynzn7czZ4hbAXYgd6aVSfCQbeGnmcVewieixTeoQH-UvKO0laTYXQ8u7TtdxuhWttg_VDfY77ZwvGRD1swGdZSMmTLBziZhIAtkNnBo9JKcdBdlLAcaQsRe-fkCi6IX35qw0femwlMoGreUBV5C2P0hjjK-lDabHU6Xo.1lf-54JivjE6fxrgmFW6Xt3gl1ktRhQsHJOTAUPeF4Q&dib_tag=se&keywords=sandisk%2Bsd%2Bcard&qid=1728775501&sprefix=sandisk%2Bs%2Caps%2C116&sr=8-18. Accessed 12 Oct. 2024.