Fall 2022

Lapnitor: A web service that protects your laptop from theft.

Michael Ameteku
ma368@uakron.edu

Lapnitor: A web service that protects your laptop from theft.

Michael Ameteku

Department of Computer Science

Honors Project

Submitted to

The Williams Honors College

The University of Akron

Table of Contents

Abstract

Laptop theft is becoming an issue worldwide. According to an article from 2018, Security Boulevard stated that a laptop is stolen every 53 seconds. Using a laptop's camera, we can monitor the surroundings of the laptop

and reduce a laptop's probability of being stolen. According to the University of Pittsburgh, a laptop has a 1-in-10 chance of being stolen and nearly half of these thefts occur in offices or classrooms. These thefts mostly occur when a laptop owner leaves their device unattended for a certain period of time to maybe go visit the restroom or attend to a call when they are in a quiet area like a library. Thinking about this problem of theft brought me to one of the main ways theft of property is curbed –Through surveillance and monitoring of a property and its surroundings. The Lapnitor, which is a system consisting of a desktop application for capturing a continuous stream of images from the camera of the laptop, sending the images captured over to the backend, a backend for processing and detecting suspicious activity from the images sent over from the desktop using the OpenCV(computer vision library) and a mobile application for notifying the user of any suspicious activity found by the backend through image evidence and notifications. This significantly reduces the probability of a laptop being stolen and even the possibility of being retrieved if the thief's face is captured in one of the images.

Chapter 1: Introduction

Having an interest in computers before University and pursuing a degree in computer science has been a fulfilling and learning-filled journey. I have grown through the lessons taught by lecturers, personal projects and competitions I have participated in throughout my years in college.

Through all these experiences, I have come to understand how the software and hardware of computers work and how they can be used to drive impact in our communities. I, therefore, decided to try to use my final year project as an opportunity to create a product that could potentially be used positively. The Lapnitor service is a web service that utilizes the power of the laptop camera together with computer vision and machine learning to notify the users through their phones when there is an attempted theft or unauthorized access.

1.1 Goals and Objectives

Lapnitor was a project I aimed to implement some of the most important knowledge I gained in school like machine learning, software engineering and development and object oriented programming whilst also creating something I would personally like to use. I knew it would be a challenge as it involved creating a sustainable system, designing and writing “non-spaghetti” code and understanding and using computer vision and machine learning.

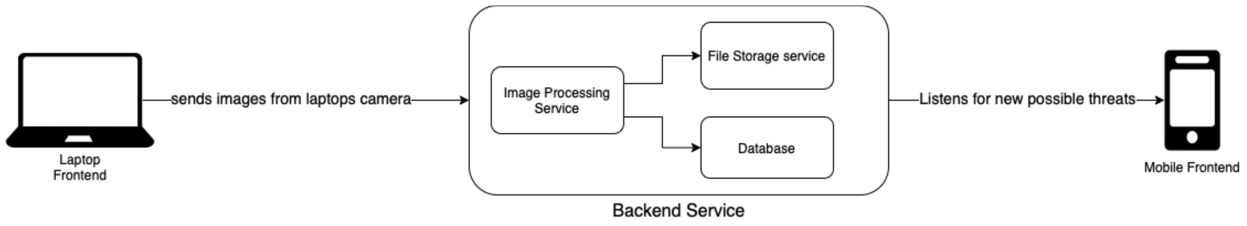
Lapnitor stemmed from two events;

- A hackathon at which two friends and I won two prizes by creating a python script that sends text messages to a user anytime an unwanted item (e.g. gun) is spotted in the user's specified camera.
- The fear of having my laptop stolen in my school's library anytime I had to leave it for a few minutes to go to the rest-room or guide a friend to where I was sitting.

Chapter 2: Design and Implementation

As a young software engineer, the most important lesson I have learnt so far when building software is to research and design before thinking about implementation. I therefore brainstormed through different designs and discussed them with friends and my supervisor before picking the one that had the best balance of scalability, could be built within a short period of time(a semester) and could notify users in real time.

2.2 High Level Overview of Service



The laptops frontend connects with the laptops camera, captures video/image feed and sends the captured images to a backend REST API service. The backend service, which consists of the image processing service, a database and a file storage service, is responsible for processing the captured images for any potential threats and saving records of those potential threats. The mobile frontend then listens for any new threats, notifies and makes the image available for the users to view. All three parts work together to keep the users alert on what is going on around their laptop whilst they are away.

This design was chosen because of the following reasons:

1. Detaching the systems and having them communicate using APIs reduces both coupling and the risk of the entire service breaking down if one of them fails.

2. Having the mobile frontend listen for new notifications rather than poll using a specified cadence makes notifications real-time.

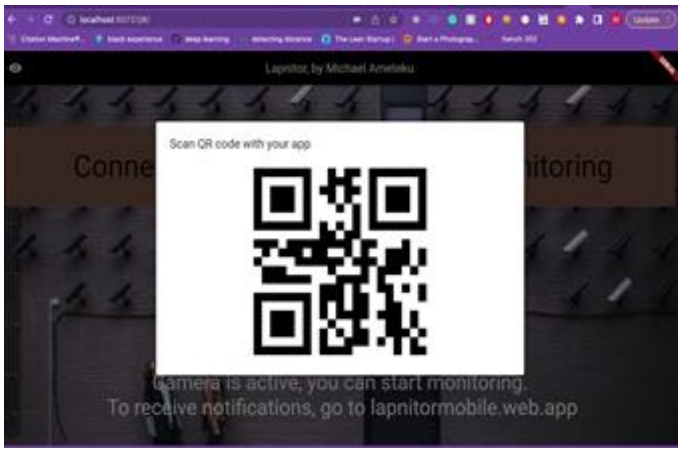
2.3 Laptop Frontend

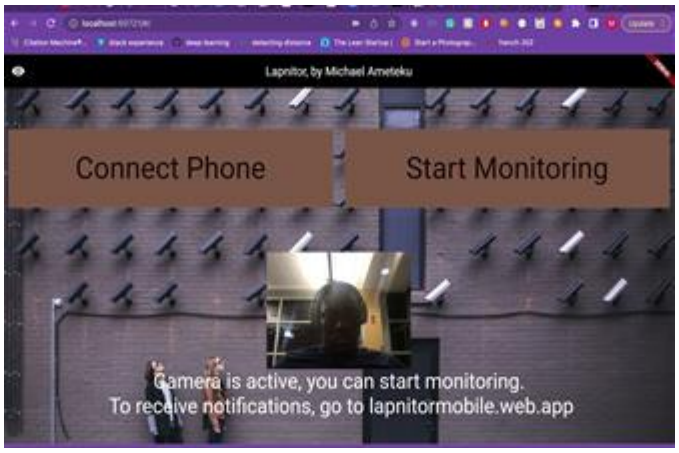
As mentioned before, the main function of this part of the service is to collect and send video feeds from the laptop's camera. Upon weighing options on what platform, language and frameworks to use, I decided on building a web application because it would be Operating System agnostic - which meant I didn't have to handle each operating system differently. The framework used for building the application was Flutter(a mobile application development framework) since Flutter's recently announced support for building web applications was something I wanted to experiment with. Since I used flutter, the programming language I had to use was dart.

The web app consists of a single page with two buttons:

1. **Connect Phone:** This displays the QR code used by the mobile frontend to connect and receive alert notifications from the backend. When the laptop web app has been fully loaded and connected to the backend, its IP address together with an internal mapping is used to locate specific laptop images and direct its notifications to the right phone.
2. **Start Monitoring:** This triggers the application to start recording and sending the feed to the backend. The library used for communicating with the laptops camera is WebRTC, which is a library that makes real time video and audio communication possible by acting as an interface between the code that needs the video/audio feed and the platform's camera among other functions. Upon the video feed being received, it is marshalled into JSON format and pushed to the backend using one of the endpoints the backend has made available.

Below are screenshots of the application.


Lapnitor, by Michael Ameteku
Scan QR code with your app
Connect Phone
Start Monitoring
Camera is active, you can start monitoring.
To receive notifications, go to lapnitormobile.web.app


Lapnitor, by Michael Ameteku
Connect Phone
Start Monitoring
Camera is active, you can start monitoring.
To receive notifications, go to lapnitormobile.web.app

QR Code used to connect mobile frontend

Single Page with Buttons

2.4 Backend, Database and Storage

The backend consists of three parts; an Image Processing Server, a NoSQL document database and a file storage system both from Firebase.

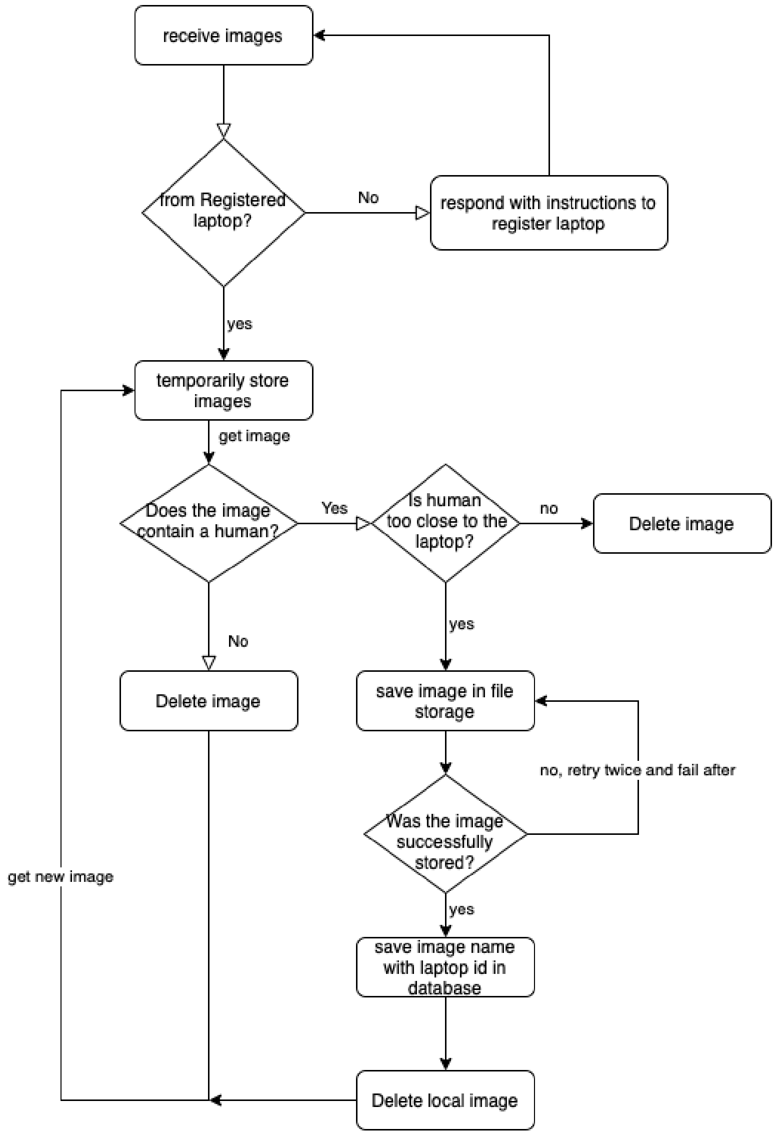
The server was built using Nodejs, Typescript(ensures JavaScript objects are typed), and the ExpressJs framework(a backend web application framework).

The server consists of a

1. ClientRegister class responsible for registering and managing the lifecycle of all connected laptops.
2. ClientDelegate class(which controls the database and file storage connector) that handles the storing and processing of a client’s images using the Tensorflow computer vision machine learning model, coco-ssd.

Google Firebase made well-documented libraries available for interacting with the database and file storage services. Connecting and using them were straightforward.

Below is the workflow of the backend:


receive images
from Registered laptop?
No
respond with instructions to register laptop
yes
temporarily store images
get image
Does the image contain a human?
Yes
Is human too close to the laptop?
no
Delete image
No
Delete image
yes
save image in file storage
no, retry twice and fail after
Was the image successfully stored?
yes
save image name with laptop id in database
Delete local image
get new image

2.5 Mobile Frontend

The mobile frontend was initially supposed to be implemented as a native mobile application. However, the complexity that came with accessing an app not in the app store for testing by others was an inconvenience. I therefore built the mobile frontend also as a progressive web application(a web application that can be saved and used as a native mobile application).

The frontend was also built in Flutter and depends on the file storage and database libraries for receiving any new threat alerts.

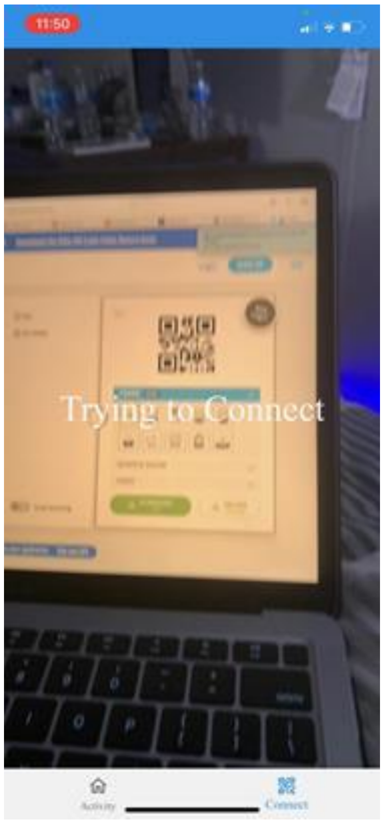
The mobile frontend consists of two screens:

1. **Scan screen:** By default, this is the screen shown when the user first opens the web app. It requests access to the camera and uses it to scan the QR on the laptop frontend. Using the QR code content, it

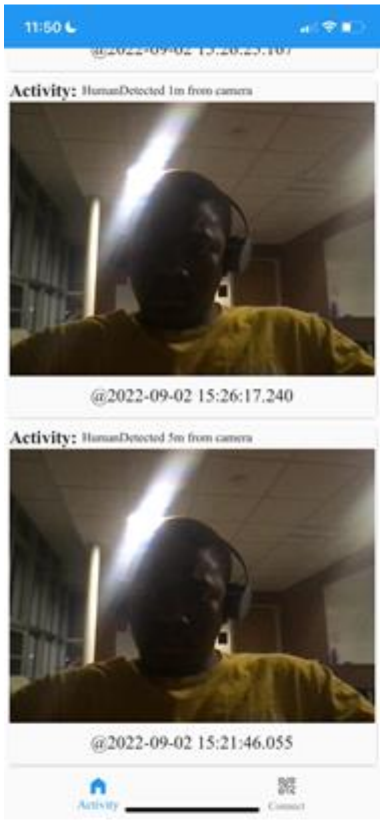
accesses the database and verifies that a laptop with that connection id exists and if so, connects. After a successful connection, it automatically switches over to the feed page.

2. **Feed Page:** The page listens to the database for any new notifications and displays the image with the time it was captured. The Firebase Firestore database has a constantly updated stream object that other objects can subscribe to. Using this stream, new notifications with images are received in real time.

Below is what the web application looks like.



Scan Page



Activity: HumanDetected 1m from camera

@2022-09-02 15:26:17.240

Activity: HumanDetected 5m from camera

@2022-09-02 15:21:46.055

Feed Page

Chapter 3: Mistakes and Learnings

This project has brought to light how much work it can take to build a product that is scalable and can be used by the general public. I have learned a lot through some of the mistakes I made and solidified some of the principles I learned in school. My most important take away was that studying computer science and hoping to have a career in software engineering is a choice I do not think I will regret.

3.1Mistakes:

1. Underestimating the scope of the project and amount of time needed to finish sections of the project.
2. Picking a technology(Flutter web) based on it being flashy and new.
3. Sometimes getting deep into the code before going back to design.

3.2 Learnings:

1. Defining and going back to refine a design is a part of the development process.
2. Setting aside time to design a system and its code before implementation saves a lot of time in the long term.
3. Using tools with a medium-to-large supporting community also saves a lot of time.
4. Sometimes it is not worth it to reinvent the wheel.

Chapter 4: Future Work and Improvements

As mentioned in my mistakes, the original scope of the project would need more time than a semester(3 months) to complete. The following are features that were trimmed but could possibly be added in the future:

1. **Add notifications to the mobile frontend:** The mobile frontend only listens for and displays any new potential threats when the user has opened the application. A more useful feature would be to have the application send a ping(notification) to the phone when the user is out of the application.
2. **Set off alarms on the laptop frontend:** When a threat is detected and the laptop owner is notified, the only available option is to rush back to where the laptop is. Adding the ability to set off an alarm or voice warnings could scare off the threat.
3. **Ability to detect movement of the camera:** The current implementation of the service has blind spots outside of the camera's view. If an intruder approaches the laptop from the back of the laptop, there is no way to detect and notify the owner. In order to tackle this I plan on training the machine learning model to detect when the camera(and laptop lid) is being moved.
4. **Improve the UI of both frontends:** The UI of both frontends has only the needed functions and buttons. I plan on improving the UI and UX.

5. **Pay for backend hosting:** The image processing server is currently being hosted on Heroku, a hosting service with a free tier that provides very limited processing power. With the server needing a lot of memory and processing power for processing each image frame, I may need to upgrade to a plan that better supports the load.

Chapter 5: How to Use the Live Version

1. On your laptop, go to lapnitor.web.app
2. Accept camera permission when asked.(Sometimes your browser's laptop has permissions turned off by default so you will have to go into settings to change that)
3. Upon accepting permissions, click on the **connect phone** button and a dialog box with a QR code will open. (If a dialog box does not display, this means my Heroku free tier limit has been reached and I have to restart it.)
4. On your phone or where you want to receive notifications, go to lapnitormobile.web.app
5. Accept camera permissions(will be used to only scan the QR code)
6. Use the mobile web app to scan the QR code displayed on your laptop.
7. Upon a successful connection, go back to the laptop, click out of the dialog box to exit and click on the **start monitoring** button.
8. The service has started working and will send any new images to the mobile web app.

A special thanks to my supervisor, Dr. Zhong-Hui Duan for helping me throughout this project.

References

Sharpe, N. (2018, September 10). *7 shocking statistics that prove just how important laptop security is*. Security Boulevard. Retrieved September 18, 2022, from https://securityboulevard.com/2018/09/7-shocking-statistics-that-prove-just-how-important-laptop-security-is/

M., & Shiff, L. (2020, May 13). *Real time vs batch processing vs stream processing*. BMC Blogs. Retrieved September 9, 2022, from https://www.bmc.com/blogs/batch-processing-stream-processing-real-time/

Dignan, M. (2016, January 25). *Making blocking functions non-blocking in JavaScript*. Medium. Retrieved September 9, 2022, from https://medium.com/@maxdignan/making-blocking-functions-non-blocking-in-javascript-dfeb9501301c

Martin, E. M. (2022, September 9). *WebSockets vs server-sent events: Ably blog: Data In Motion*. The Ably Blog. Retrieved September 9, 2022, from https://ably.com/blog/websockets-vs-sse

Birnbaum, R. (1966, March 1). *Converting PNG to tensor tensorflow.js*. Stack Overflow. Retrieved September 9, 2022, from https://stackoverflow.com/questions/53231699/converting-png-to-tensor-tensorflow-js

Asadullahdal. (2022, July 12). *Realtime distance estimation using OpenCV - Python*. GeeksforGeeks. Retrieved September 9, 2022, from https://www.geeksforgeeks.org/realtime-distance-estimation-using-opencv-python/

University of Pittsburg. (2022, August 5). *Laptop and mobile device theft awareness*. Laptop and Mobile Device Theft Awareness | Information Technology | University of Pittsburgh. Retrieved September 18, 2022, from https://www.technology.pitt.edu/security/laptop-theft