VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Binh Nguyen

# Researching and AI Testing Strategies for Video Management System.

Information and Communication
2023

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

## ABSTRACT

| | |
|---|---|
| Author | Binh Nguyen |
| Title | Researching and AI API Testing for Video Management System |
| Year | 2023 |
| Language | English |
| Pages | 86 |
| Name of Supervisor | Kenneth Norrgård |

Embark on an insightful exploration of testing strategies for ITD Group's Smart Video Management System within the context of this thesis. The study delves into the realm where cutting-edge technology converges with boundless information. The Video Management System is simulated employing with Python, facilitating a comprehensive understanding of its intricate working.

This thesis also moves readers with the transition from manual testing API into more advanced methods like unit testing or real cam testing. Unveiling the features of AI APIs, system bugs, script tests. Moreover, the thesis will analyze similar video management system like Milestone, Axis for the valuable insights and will offer further knowledge on the field of Software Testing as well as contribute necessary value for the intelligent video management system (iVMS) industry.

With all the above activities, this study illuminates a pathway to be a responsible Software Engineer within the realm of Video Management System.

Keywords Video Management System, unit testing, AI APIs, system bugs, script tests, Milestone, Axis.

## CONTENTS

**LIST OF FIGURES AND TABLES**

# 1 INTRODUCTION

Computer vision in the AI industry is playing a crucial role in digital transformation revolution. That is also the great motivation for this thesis to be written. With the goal of providing software engineers different ways to approach practical projects related to computer vision in general or specifically video management systems with AI integrated.

This thesis comes from teaming up with 'Isecurity' project at ITD Vietnam Group. The hope is that it becomes a solid starting point for software testing engineers in the future. At current time, ITD Group is a big player in making cool things happen as Vietnam gets ready for a digital transformation in 2023. They have worked on many projects, including the creation of a smart camera management system that holds great potential in various domains such as time attendance management, home security, smart home, road traffic system, monitoring, highways, and so on.

In this thesis, readers will get a solid basis for entering the profession by examining a few intriguing problems and offering insight into the mindset of an AI software testing engineer. These concerns include of knowing how to build a camera management system, how to train an AI vision system, and how to test an AI system in a real-world corporate environment. Additionally, the thesis will cover the critical component of what a tester must comprehend regarding end user demands.

## 1.1 Smart Video Management System

The comprehensive nature of the Video Management System project is extremely extensive, as it is built using a microservice architecture[1]. Currently, during the

[1] Microservice architecture is like building with Lego blocks. Instead of one big piece, we have lots of smaller, independent pieces that do specific jobs. Each piece talks to the others using a clear set of rules, and we can work on, put in place, and grow each piece on its own.

This way of building things helps make software more flexible, scalable, and easier to take care of. It is like having a bunch of small, specialized teams working together instead of one huge team writing of this thesis, the project is still in progress and involves several distinct scopes, including the backend, backend API, AI training model (updated to YOLO v8 at the present), API of AI, and various frontend modules for administrator…

However, this thesis specifically focuses on in-depth testing of AI APIs for supporting improving recognition features purpose such as Face recognition, Person detection, writing script test for end user of object tracking feature. Additionally, it delves into VMS research, providing valuable references for the development of ITD Group's product as well as provide a strong point of view in how to make a pilot project related AI video management system which is a helpful knowledge for people who would like to become an AI testing engineer in the future.

The thesis's scope is related to AI, Data, and research. Therefore, there would be too many figures provided, it requires the readers to be patient for having a safe journey up to down sequentially.

## 1.2 API functionalities

First and foremost, it is necessary to understand how the API works. One important starting point for the thesis is described in ITD AI API Camera System Documentation (Nguyen & Le 2023d, 2-13).

Read and carefully understand the API documentation is always the most important thing-to-do in the rule list of any tester. This helps every tester to know what the system is talking about, how to request and the ways the system response for those requests.

trying to manage everything at once. For further exploration of microservice architecture principles, refer to foundational resource such as Microservices (2014) by Martin Fowler (Fowler 2014).

Moreover, reading the API documentation for an AI project is crucial for more reasons such as: Understanding functionality, integration, and development, use cases, data requirements, model performance, scalability, troubleshooting, security, cost and pricing, updates and versioning, legal and ethical considerations, collaboration, …

After knowing the API functionalities that we will work with, the thesis will implement several research related to the AI Camera System industry by moving to the next chapter where readers will expand the vision through the literature review of one of the most important developments in this generation.

# 2 LITERATURE REVIEW

In this chapter, the thesis will dive into the World of AI-powered camera systems. There are also investigations into these camera tools that is helpful to know how they can handle and process video recordings, the methods used to rigorously assess their proper functioning, as well as the specific software and hardware will be used.

Concluding the preliminary investigation, attention will be directed to similar Video Management Systems are *Milestone* and *Axis*. This chapter sets the stage for a deeper exploration into how AI impacts video management system while simultaneously laying the academic groundwork for researching analogous software products. It serves as a practical guide in crafting effective solutions for contemporary software challenges.

## 2.1 Overview of video management systems with AI capabilities

Video management system (VMS) are software platforms designed to manage, record, and analyse video footage from surveillance cameras. The integration of Artificial Intelligence (AI) within VMS has significantly transformed the landscape of video surveillance by enabling advanced functionalities that go beyond traditional video management system. (Samsara 2021).

AI in VMS allows the system with the functionality of real-time video analytics, such as object detection, tracking, and recognition. With these features, VMS can identify and classify objects, individuals, vehicles, or events easily in a video stream. For example, AI can monitor suspicious behaviour, monitor crowd density, detect unauthorized access, etc.

Besides, the integration of AI enables a VMS to predict security breaches by analysing historical data and identifying trends. Anomaly detection algorithms trigger alerts when unusual behaviour or events are detected, enabling for a proactive approach to security.

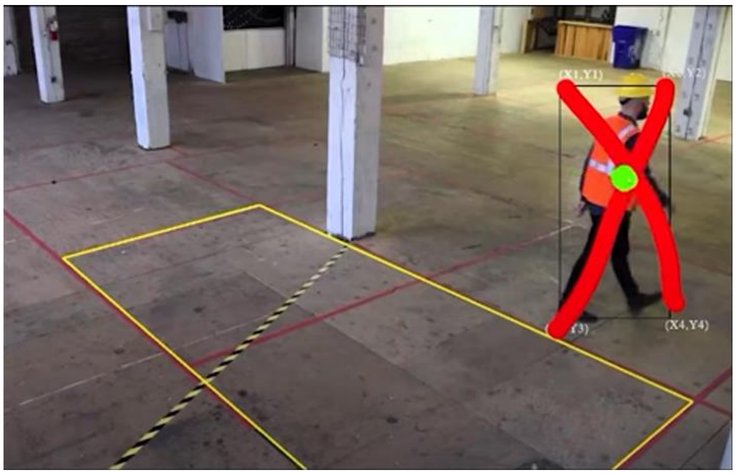


**Figure 1.** Analyse if the detected object is inside the restricted area. (Ngo 2022)

Additionally, AI-powered VMS can automatically generate alerts and notifications based on predefined parameters. This functionality guarantees swift response to critical situations so that the security professionals can receive alerts via email or mobile apps, allowing them to take prompt actions.

The integration of facial recognition technology into VMS also enables the Identification and verification of individuals. This helps with visitor management, access control, and strengthening security procedures.

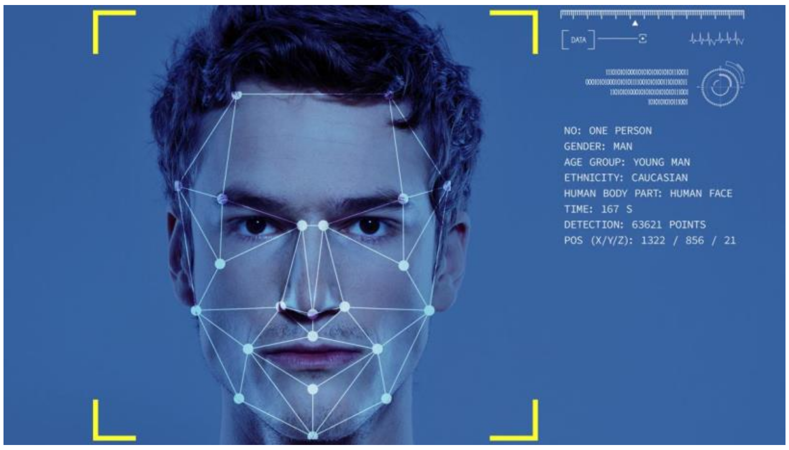


**Figure 2.** Facial recognition. (Refaces 2020)

Moreover, AI-driven VMS can extract useful information from video footage to support decision-making. The layout of stores and marketing tactics, for instance, can be improved by studying client mobility patterns in retail settings. (Barton & Court 2012).

## 2.2 Best practices for testing and quality assurance in VMS

To ensure the dependability and effectiveness of video management systems (VMS) with AI capabilities, stringent testing and quality assurance practices are required. Thorough testing ensures that the system works is functioning properly and provides accurate insights and secure monitoring. This section goes into the important best practices for AI-powered VMS testing and quality assurance.

Firstly, understanding the project documentation provides a solid basis. Understanding the system's architecture, technical specification, and expected functionality are all part of this. Clear project documentation establishes the context required for effective testing and quality assurance initiatives. As an AI Test engineer, at some companies this is also the responsibility to become one of the editors for those project documentations.

Besides, every tester needs to clearly know the survival way in any working environment. Let’s take some basic dive into how the system work, how to fetch camera from its IP address, RTSP (Real Time Streaming Protocol), how to connect it with the company API, how to connect the AI System Camera into the personal machine (Laptop, PC), how to draw on it for testing the AI by using OpenCV library, etc. Those concepts will be covered inside the chapter 3 and chapter 4 of this thesis. Therefore, if you are a beginner in any related project, do not hesitate to bother your boss or seniors about those doubts, understanding the project is very important.

When step into a project it is necessary to ensure that the software backend as well as the AI API function as expected. During the development process, testers or software engineers could employ a variety of technical tools, such as Postman, Pytest. If there is a system bug, AI testing engineers must provide relevant report to both the AI team and the IT team as soon as possible. In other words, AI testing engineers should serve as a bridge between a company's AI and IT departments. Those contribution would affect positively to further automation testing processes.
The understand of a comprehensive testing framework is also play an important role that spans functional, performance, and security testing. Functional testing assesses individual aspects like object detection and alert generating, performance testing measures system responsiveness under varying loads, and security testing detects exploitable flaws. (Nist 2023).

Furthermore, a good software testing strategy should consider concluding the process of Chaos testing. It is the process of deliberately putting a system to stressful and unexpected conditions. This is critical for robustness since it checks the system's resilience, fault tolerance, and capacity to recover gracefully from faults.

According to Shivani Naidu, “Chaos Testing, also known as Chaos Engineering, is a methodology that involves running controlled experiments to test software systems’ resilience to unpredictable, real-world events. “(Naidu 2023).

Chaos Testing is normally gone with a big project which is structed as microservice structure. The testing idea is turning off one of the services but still ensure that the other services as well as the whole system are still working. Based on the research on similar projects, almost intelligent camera management systems (iVMS) are built based on the microservice architecture. Therefore, applying strategies for chaos testing before it reaches the end user must be in the bucket list. The list of the top chaos engineering tools can be found at the article “The top chaos engineering tools” (Schillerstrom 2022).

Finally in any typical Software testing strategy, User Acceptance Testing (UAT) only accounts for 20 percent of effort but determines 80 percent of the success of a software on the market. End-users evaluate the system's usability and functioning during user acceptance testing. Real-world user feedback assists in finding areas for improvement and aligning the system with realistic requirements.

As a test engineer, “putting ourselves into the shoes of the end user” is something worth paying attention to.

## 2.3 Description of the software and hardware

This section provides an overview of the software and hardware components used in the thesis project's execution.

**Table 1.** List of Software and Hardware tools in project.

| **Software & Hardware Tools** | **Description** |
|---|---|
| Python | Python, a versatile programming language, acts as the foundation for the AI-powered video management system's development. Python's numerous libraries and frameworks enable rapid data handling and AI algorithm integration. |
| OpenCV | OpenCV (Open-Source Computer Vision Library) is a critical tool for image and video analysis. Its capabilities |

| | |
|---|---|
| | are used within the AI-infused video management system to process video streams, detect objects, and perform other analysis. |
| Pytest | To automate testing operations, Pytest, a popular testing framework, is used. Its simplicity and extensibility simplify the validation of numerous system functionalities. |
| Postman | Postman, a popular API testing tool, assists in testing the AI API's functioning. Its user-friendly UI makes it easier to validate API endpoints and answers. |
| Github | The collaborative platform GitHub is essential for version control and project management. It is helpful for cloning necessary code on the practical chapter like chapter 3 and chapter 4 of this thesis. |
| Yolov3 | A pre-trained AI model which contains several class names for label defined before, such as: person, bird, phone, … objects. (Keineahnung2345 2023). |
| Telegram | An application which is helpful for API integration for our simulation demo project. |
| IDE | Visual Studio Code (For building a simulation), Google Colab (For training an AI model). |
| Security Camera at ITD company, Laptop Webcam | For implementing the practical process as building project and testing AI functions. |

## 2.4 Overview of similar Video Management Systems

Doing research on similar camera management systems will allow AI software testing engineers to have a more intuitive perspective. Thereby, the integration of necessary features for company's products will be significantly improved. In this section, the thesis will focus mainly on some research on VMS Milestone and VMS Axis.

The reports below are based on real work done at ITD Group, which is the focus of this thesis. Summary of features and functions are recorded through the software product trial experience which was offered by client service of Axis Communication and Milestone Learning. It is worth noting that there will not be specific citations for each feature because Milestone and Axis do not offer detailed instruction resources in the way of user experience. Instead, they provide basic training to customers through tutorial videos or training courses as well as several guidance of basic to advanced configurations in their products.

### 2.4.1 VMS Milestone (Milesight)

Milesight VMS stands out as a versatile video management system designed for using across diverse environments, catering to both small businesses and larger scale enterprises.

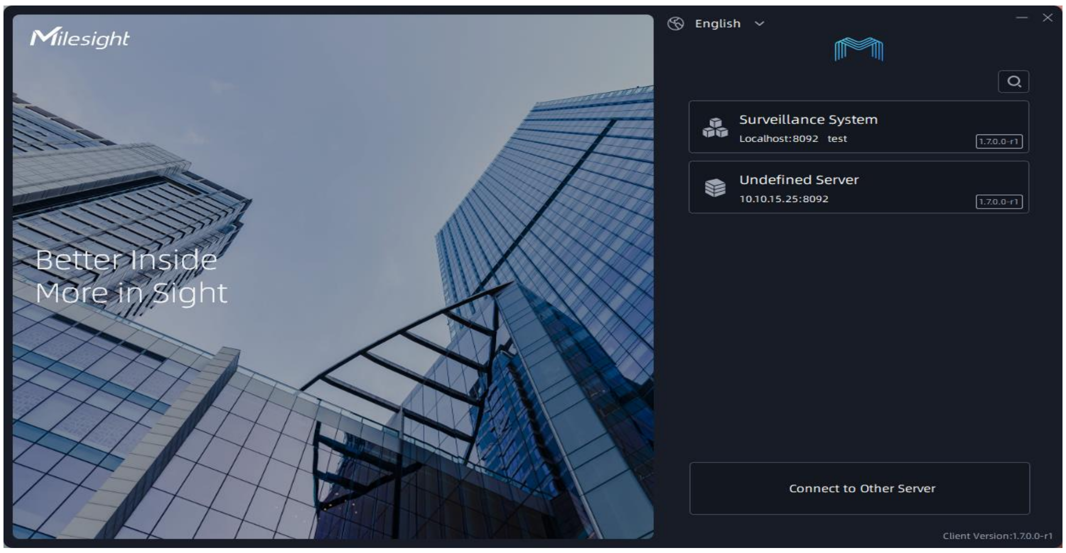


**Figure 3.** Front Page of Milesight VMS. (Screen capture)

In this section, the features of Milesight VMS are presented based on practical experience and software trial reports conducted at ITD Group in 2023. The information is derived from real-world usage and observations. Besides, the clients of Milesight VMS can also explore these features via the training program offer by Milestone Learning (Milestone Learning 2023a).

**System features**

**Video Recording and Storage:** Milestone VMS excels in recording and storing video footage from with customizable settings like frame rate, resolution, and compression. The free version supports up to 8 cameras, while more expensive versions accommodate a significantly larger number, possibly up to 13,000+ supported cameras and other devices (Milestone System 2024a).

**Video analysis:** Advanced video analysis capabilities include object detection, face recognition, motion changes, etc in different environments or contexts for example bank and finance, transportation, public facilities and buildings, city surveillance, intrusion warning, wrong way alarm, stolen object alarm, etc. (Milestone System 2022e).

**Remote Access:** Users can access video footage and analytics remotely through web browser or mobile app, ensuring monitoring and management from anywhere at any time (Milestone System 2024b).

**User Management:** The software incorporates an user management module allowing administrators to control access and permissions for each user. Ensuring the security of sensitive video data. Milestone offer detailly useful information in configuring user profile on server as well as other administrator features in their product documentation (Milestone System 2023a).

**Customizable:** Highly customizable and can be tailored to meet the specific needs of different organizations, including customization of the user interface, video analytics settings, and integration with third-party software. More information of configurations and system settings can be found out on Milestone's user manual (Milestone Learning 2023b).

**Reports:** Advanced reporting capabilities enable the analysis of video data, user access, and system performance (Milestone System 2024a).

For more details, Milestone VMS offer user manual for experiencing as well as configurating the application on their official documentation (Milestone Learning 2023b).

**AI Integrated Features**

Milestone VMS seamlessly integrates with various features, offering a comprehensive security solution.

**Motion Detection:** Detects motion in a specific area of camera view and trigger alerts or recordings, distinguishing between different types of motion, such as human movement or vehicle movement. (Milestone System 2023b).

**Face Recognition:** Recognize faces in real-time and compare them with a known database, enhancing security (Milestone System 2022c).

**Object Detection:** Detects and tracks specific objects, triggering alerts or logs based on predefined rules (Milestone System 2023b).

**People Count:** Count people entering and exiting a specific area, such as a store or building. This feature is useful in managing crowds , analysing purchase data, checking attendance, improving customer service (Milestone System 2022d).

**Audio Analysis:** Analyse audio data from the cameras and microphones, detecting specific sounds such as screaming, vehicle crashes, gunshots, or broken glass, and activate alarm or recording. This feature is useful for solving crime contexts even in blind spots. (Milestone System 2022a).

**Heat Map:** Generate heatmaps showing activity frequency and duration, valuable for analysing customer behaviour (Milestone System 2022b).

**Anomalous Behaviour Detection:** Detects unusual behaviour, such as a person loitering in a restricted area or a vehicle driving in the wrong direction. Enhancing security (Milestone System 2022c).

**Disappearance object detection:** Detects when an object is removed from a specific, preventing theft and increases security through stolen object alarm (Milestone System 2022e).

**Queue management:** Detect and monitor queues or queues in real time. Useful for optimizing queue management (Milestone System 2022d).

**Fall detection:** Detect and track when someone falls, such as in a hospital or nursing home, train station, dangerous area, … Or simply slippery streets during Finnish winter and generate alerts or notifications to aid in emergency response (Milestone System 2022c).

**Smoke and fire detection:** Detect and monitor smoke and fire in real time, enhancing safety (Milestone System 2022c).

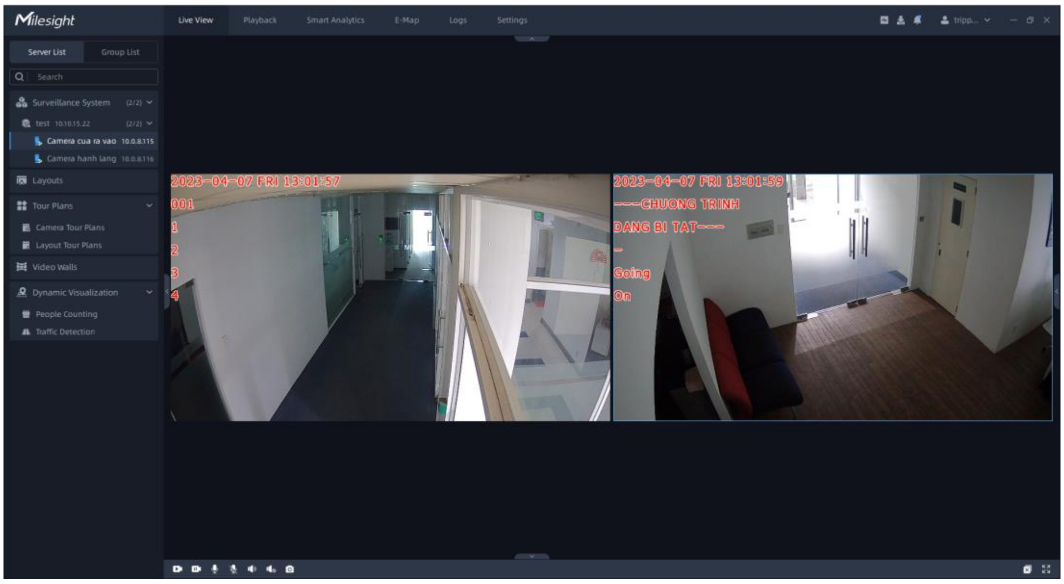


**Figure 4.** Camera view of Milesight VMS. (Screen capture)

**Features that need improvement**

The discussion on areas that could be enhanced for a better user experience is presented as personal opinions and recommendations based on the CTO of ITD Software department's assessment.

**Simplify advanced features:** Make advanced feature more accessible to all users.

**Improve app accessibility:** Provide alt text for images and icons to enhance app accessibility.

**Resolve Technical issues:** Address crashing and freezing issues reported by some users.

**Adjust costs:** Consider adjusting costs and implementing promotional campaigns to attract and retain customers.

**Optimize Smartphone Integration:** Improve integration with smartphones or mobile devices for better performance.

**Distance monitoring AI integrated feature:** Monitors and detects if people maintaining a safe distance from each other and trigger an alert or record if the social distancing rules are violated.

### 2.4.2 VMS Axis

AXIS Camera Station stands out as a comprehensive video management software tailored for small and medium businesses. Drawing parallels with Milesight, Axis offers an user-friendly integration feature, particularly notable for its seamless integrating smartphones.

In this section, the features of AXIS Camera Station are presented based on practical experience and software trial reports conducted at ITD Group in 2023. The information is derived from real-world usage and observations. Additionally, the clients of Axis communication can also explore these features via the training program offer by Axis communication (Axis Communications 2023). Unlike the documentation or the video courses offer by Milestone Learning, Axis Communication offer their clients training schedules on their training program.

*Note:* In this section, we will primarily delve into the intelligence integration and advanced features provided by Axis Communications, highlighting its similarities with Milesight VMS in system features.

**AI Integrated and advanced Features**

**Motion Detection:** Supports motion detection, triggering recording and alert in the monitored area (Axis Products 2024a).

**Alarm Management:** Supports alarm management, allowing users to configure and manage alarms for various events, such as motion detection, fake camera impersonation. network problems, or even set excluded objects to avoid sending unnecessary alarm (Axis Products 2024a).

**License Plate Recognition (LPR):** LPR technology for recognizing and capturing license plate information from vehicles, enhancing applications like parking management and security monitoring (Axis Solutions 2024a).

**Behavioural Analytics:** Utilizes AI-based behavioural analytics for detecting suspicious or unusual behaviours, such as loitering, object left behind, or crowd detection. This feature is very helpful for example traffic management, or parking lot services. (Axis Products 2024b).

**Audio Analytics:** Analyses audio data using AI, detecting specific sounds or anomalies, such as gunshots, breaking glass, or raised voices. Thereby enhancing situational awareness (Axis Communications 2024).

**Privacy Masking:** AI can be used to automatically blur or obscure sensitive information, such as faces and license plates, to address privacy concerns. (Axis Solutions 2024b).

**Integration with Access Control:** Integrate with access control systems, allowing seamless interaction between video surveillance and access control, facilitating actions like automatically unlocking doors based on visual identification. Those features can be applied to for example smart home projects. (Axis Developer Community 2024).

**Mobile App:** The Axis Camera Station mobile app provide convenient access to the surveillance system from mobile devices (Axis Support 2024a).

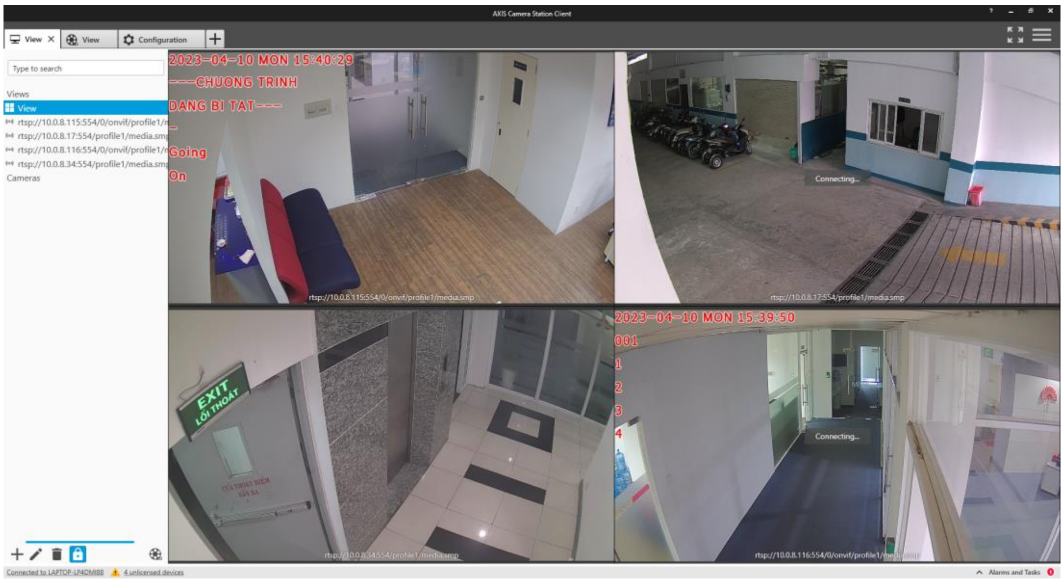


**Figure 5.** Streaming view of VMS Axis. (Screen capture)

**Integration Capability:**

**Third-party device integration:** Supports integration with third-party devices, offering a comprehensive security solution managed from a single interface. Supported third-party devices list could be found on Axis Communication user manual (Axis Public 2023).

**API:** Provides a wide range of APIs that allow integration with other software solutions for example network video, audio systems, radar, and so on. (Axis Vapix Library 2024).

**Cloud integration:** Integrates with cloud-based solutions, such as Microsoft Azure, enabling remote access to a cloud-based video surveillance and storage system (Axis Support 2024b).

**Video Analytics Integration:** Supports integration with video analytics solutions, enhancing video footage insights (Axis Products 2024b).

**Features that need improvement:**

The discussion on areas that could be enhanced for a better user experience is presented as personal opinions and recommendations based on the CTO of ITD Software department's assessment.

**Resource-intensive live view:** Live view can be resource-intensive, impacting system performance. Optimizing the live view feature to reduce resource usage is recommended.

**Storage Management:** Continuous writes can result in large amounts, which can be challenging to manage.  Improving storage management can enhance user experience, especially for those interested in integrating data with cloud services. There would be more requirement for the users in other cloud services like Google Cloud Platform or Amazon Web Services.

**False alarms:** Motion detection can be prone to false alarms, causing frustration for users. Axis Camera Station can work to improve the accuracy of motion detection to reduce the frequency of false alarms.

**Alarm Management:** Setting up and managing alarms can be time consuming. Improving alarm management capabilities to make the alarm setup and management process more efficient.

**Mobile app functionality:** Mobile apps may not provide the same level of functionality as the desktop version. Enhancing the functionality of mobile applications will contribute to a more comprehensive remote access experience for users.

## 2.5 Key Insights from Literature Review

In the second chapter, the thesis generally has offered an overview of several software testing concepts as well as has showed the outstanding features that current

AI-integrated video management system software have through research of Axis and Milestone VMS.

With all the precious information, in the next two chapters, this thesis hopes to bring the stated theories and apply them into practice by training a sample model, simulating an intrusion warning project, conducting some testing for an AI-integrated video management system at ITD company, and writing a sample of script test that will give an AI test engineer a more realistic view in this industry.

# 3 SIMULATION OF AN AI CAMEARA APPLICATION

Gaining insight into the workflows of both the AI and IT teams is critical as an AI testing engineer. This chapter seeks to provide a thorough overview of the tools and procedures used to accelerate AI model training. In addition, readers will be guided to implement an Intrusion Warning project. This exploration not only deepens comprehension but also proven invaluable for future endeavors in smart camera system projects. Let's take the first step in any workflow in the AI industry today "Implementing a pilot project to gain momentum". (Ng 2023).

## 3.1 Training visual intelligent model - step by step

*Preparation: Google Drive, Google Colab, training dataset image, training dataset label.*

Firstly, setting up a drive folder and ensuring the folder structure look like the folder structure for training AI model of the project belong to this thesis (Nguyen 2023h).

To initiate this "cake-eating from the size down" process. Let's focus on the data folder, the standard file structure for training data in computer vision AI model is depicted as below.

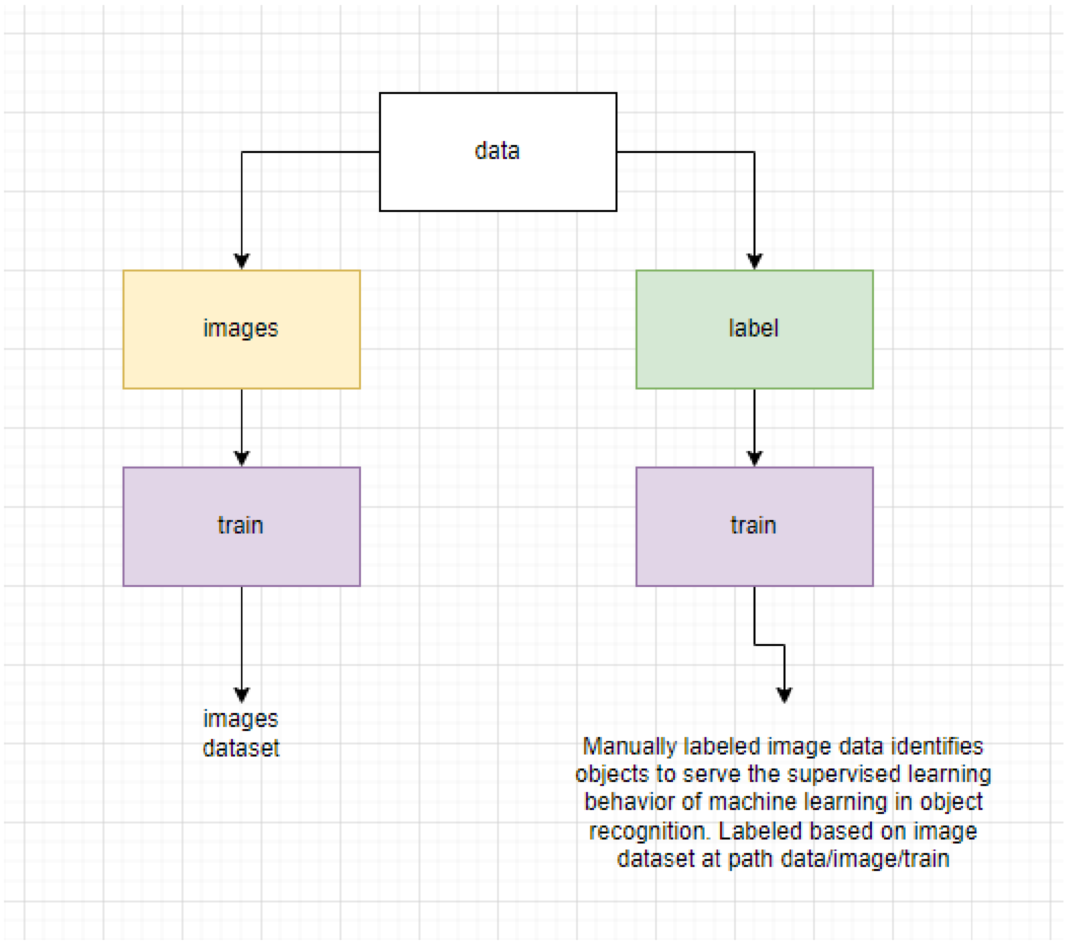
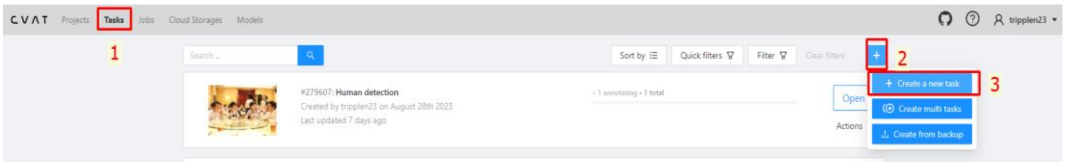

data
images
label
train
train
images dataset
Manually labeled image data identifies objects to serve the supervised learning behavior of machine learning in object recognition. Labeled based on image dataset at path data/image/train

**Figure 6.** file structure for using in training data. (Draw.io screen capture)

*Step 1: Configuration for Labeling Task.*

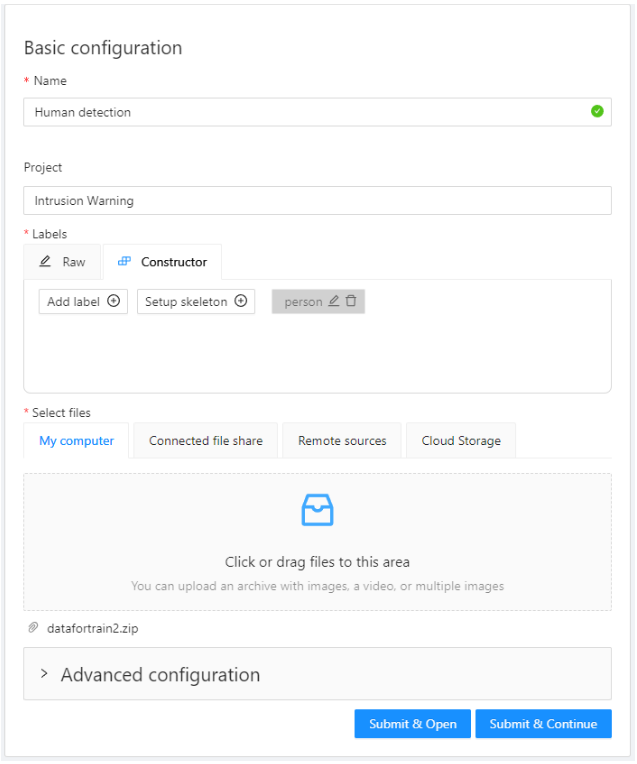
Labeling image datasets for supervised learning can be executed through various tools. For a fast and straightforward approach, explore the UI web-based platform at cvat (Cvat 2024).

After registering and signing in, click into the “Task” (1) on the web navbar, then interact to the “+” button (2) and choose “Create a new task” (3) for creating a new task to label from an uploaded dataset.



#279607: Human detection
Created by tripplen23 on August 28th 2023
Last updated 7 days ago
• 1 annotating • 1 total

**Figure 7.** Create task for labelling the image dataset. (CVAT screen capture)



Basic configuration

* Name: Human detection

Project: Intrusion Warning

* Labels

Raw | Constructor

Add label | Setup skeleton | person

* Select files

My computer | Connected file share | Remote sources | Cloud Storage

Click or drag files to this area

You can upload an archive with images, a video, or multiple images

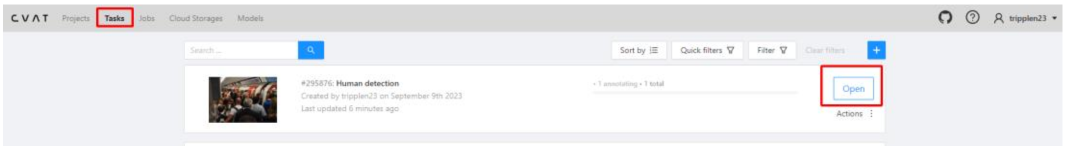
datafortrain2.zip

Advanced configuration

Submit & Open | Submit & Continue

**Figure 8.** Basic configuration for labelling task. (CVAT screen capture)

After configuring the label task, go back to the link in the navigation bar as “Tasks” and click on “Open” button on the chosen task.



#295876: Human detection
Created by tripplen23 on September 9th 2023
Last updated 6 minutes ago
• 1 annotating • 1 total

**Figure 9.** Opening the task project for labelling. (CVAT screen capture)

*Step 2: Labeling Interface.*

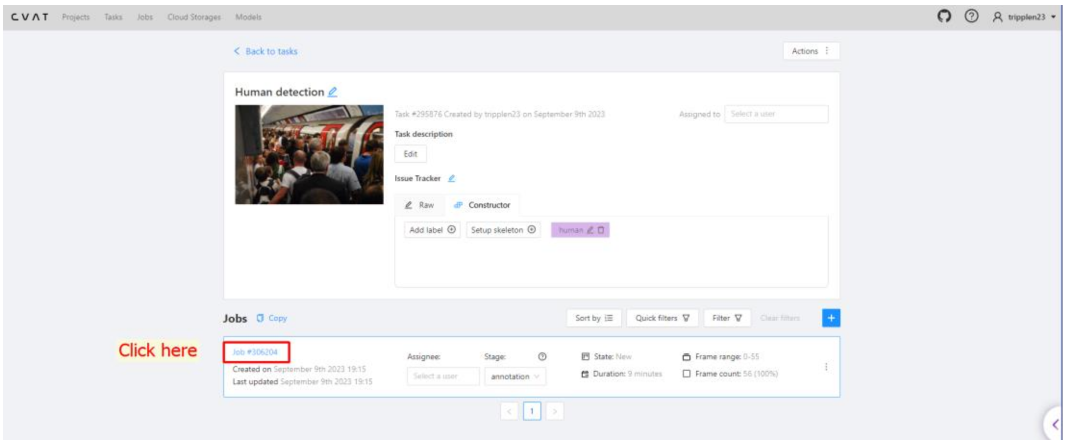


**Figure 10.** Going to the job#[id] for labelling. (CVAT screen capture)

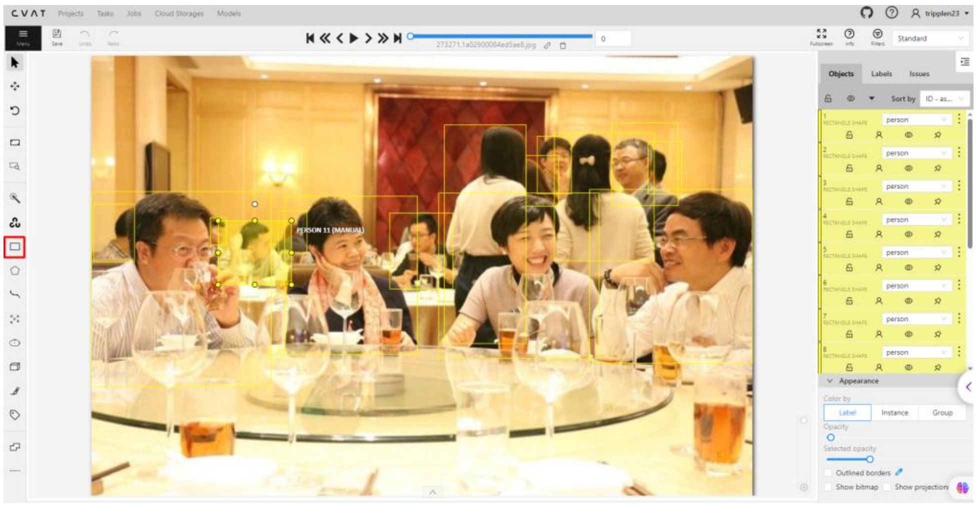
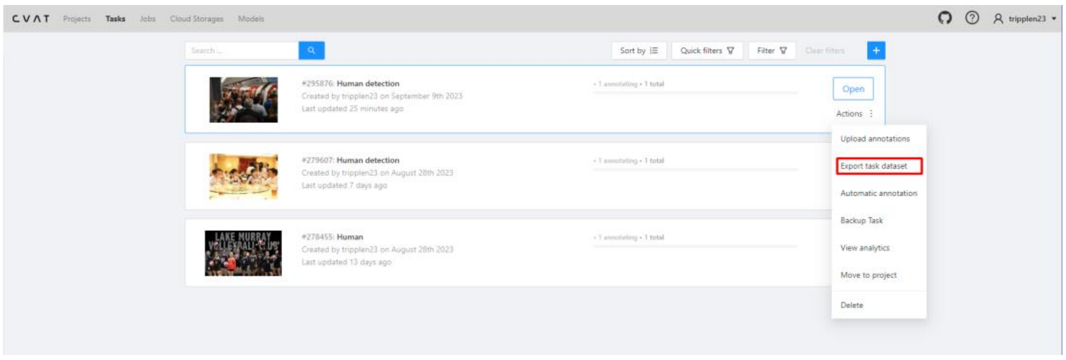


**Figure 11.** Labelling the objects manually. (CVAT screen capture)

It is recommended to implement the labelling process as much as possible. It is because, the more dataset, the more in accuracy rate for machine to detect the defined object.

*Step 3: Exporting Labeled Dataset*

After labelling throughout the dataset and saving, go back to the “Tasks” on the navigation bar, hover mouse to “Action”, then click on “Export task dataset”.



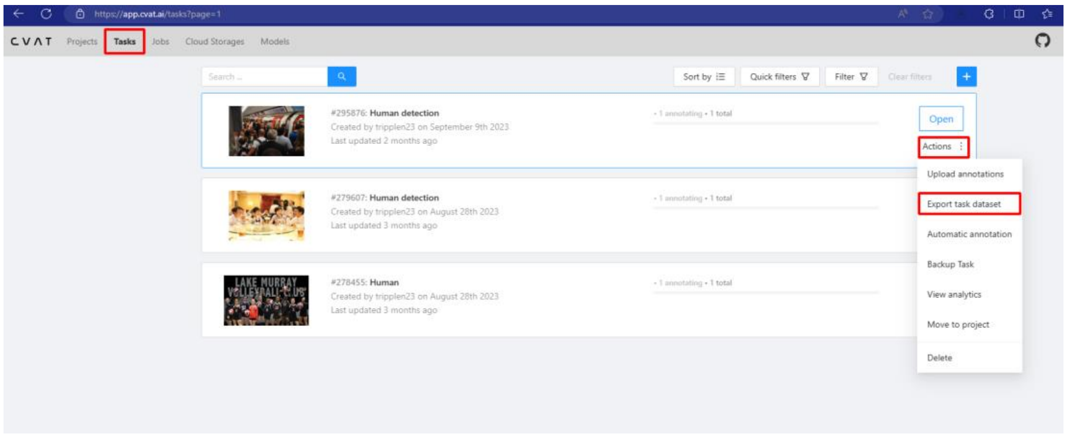
#295876: **Human detection**
Created by tripplen23 on September 9th 2023
Last updated 25 minutes ago
• 1 annotating • 1 total

#279607: **Human detection**
Created by tripplen23 on August 28th 2023
Last updated 7 days ago
• 1 annotating • 1 total

#278455: **Human**
Created by tripplen23 on August 28th 2023
Last updated 13 days ago
• 1 annotating • 1 total

**Figure 12.** Exporting labelled dataset. (CVAT screen capture)

*Step 4: Exporting Task Dataset*

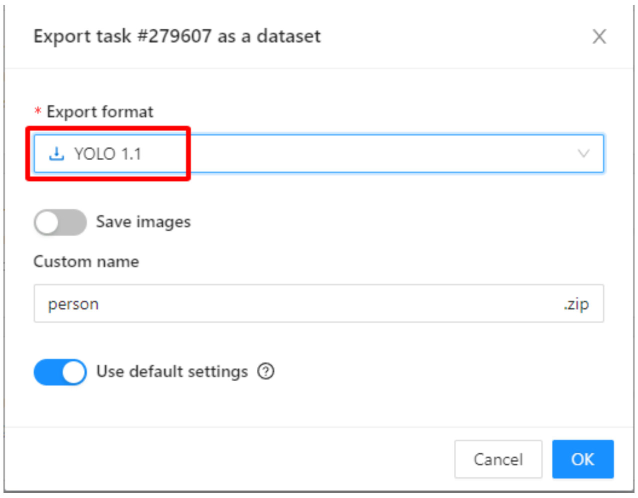


#295876: **Human detection**
Created by tripplen23 on September 9th 2023
Last updated 2 months ago
• 1 annotating • 1 total

#279607: **Human detection**
Created by tripplen23 on August 28th 2023
Last updated 3 months ago
• 1 annotating • 1 total

#278455: **Human**
Created by tripplen23 on August 28th 2023
Last updated 3 months ago
• 1 annotating • 1 total

**Figure 13.** Export task dataset. (CVAT screen capture)



Export task #279607 as a dataset

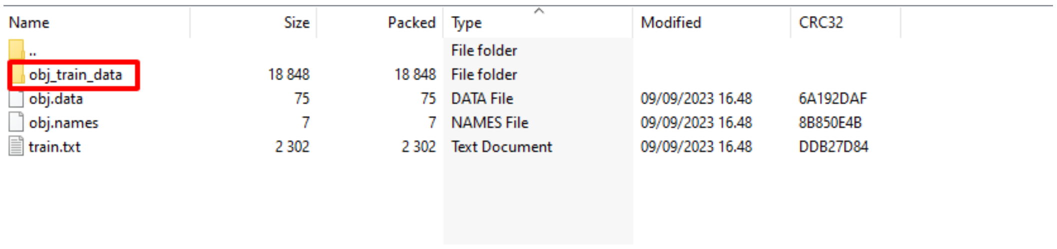
* Export format: YOLO 1.1

Save images: off

Custom name: person .zip

Use default settings: on

**Figure 14.** Modifying the export format into YOLO1.1. (CVAT screen capture)



| Name | Size | Packed | Type | Modified | CRC32 |
|---|---|---|---|---|---|
| .. | | | File folder | | |
| obj_train_data | 18 848 | 18 848 | File folder | | |
| obj.data | 75 | 75 | DATA File | 09/09/2023 16.48 | 6A192DAF |
| obj.names | 7 | 7 | NAMES File | 09/09/2023 16.48 | 8B850E4B |
| train.txt | 2 302 | 2 302 | Text Document | 09/09/2023 16.48 | DDB27D84 |

**Figure 15.** `obj_train_data` in the person.zip that we exported. (Screen capture)

Extract the downloaded "person.zip," revealing a folder named `obj_train_data`. Place this folder into the designated path on Google Drive as: **My Drive/ComputerVision/TrainYoloV8/data/labels/train.**

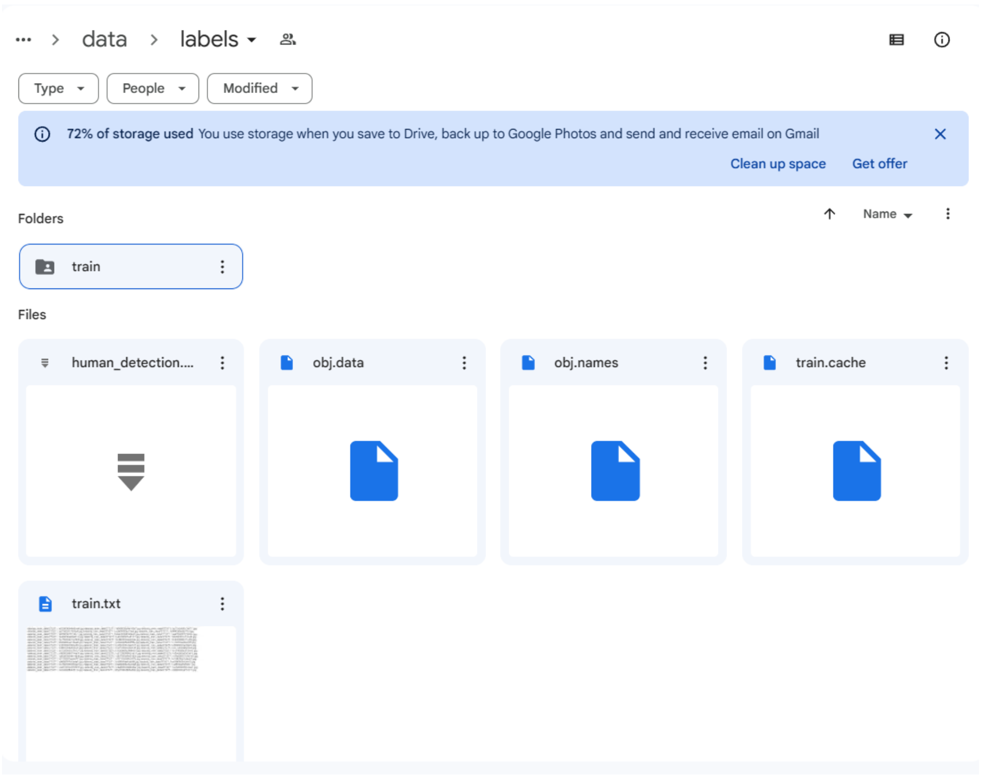
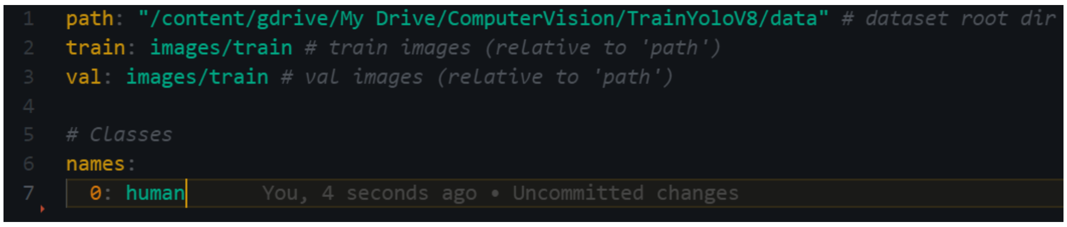


**Figure 16.** Set up folder for labelled training data. (Google Drive screen capture)

*Step 5: Creating `google_colab_file.yaml`*

Now, create a `google_colab_file.yaml` file which play a role as a pipeline between the dataset and the defined classes:



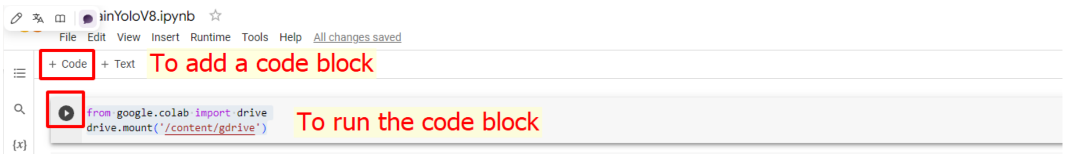
```yaml
path: "/content/gdrive/My Drive/ComputerVision/TrainYoloV8/data" # dataset root dir
train: images/train # train images (relative to 'path')
val: images/train # val images (relative to 'path')

# Classes
names:
  0: human        You, 4 seconds ago • Uncommitted changes
```

**Figure 17.** `google_colab_file.yaml` file  content. (Source code screen capture)

*Step 6: Modifying Google Colab Python notebook*

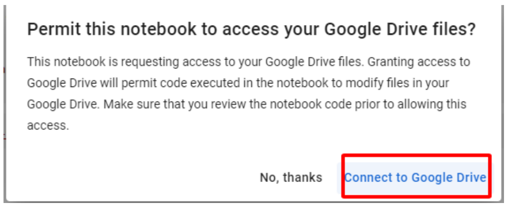
Initiate a connection between Google Drive directory and the current Google Colab Python notebook. Set up the folder structure refer to the project belong to this thesis (Nguyen 2023i).

After accessing to the python notebook file (.ipynb) in google colab. The first 2 lines are created in the first cell or first code block is importing class 'drive' in the `google.colab` library and then use it to mount the current Python Notebook to the created drive folder.



```
from google.colab import drive
drive.mount('/content/gdrive')
```

**Figure 18**. Connecting to Google Drive. (Google Colab screen capture)

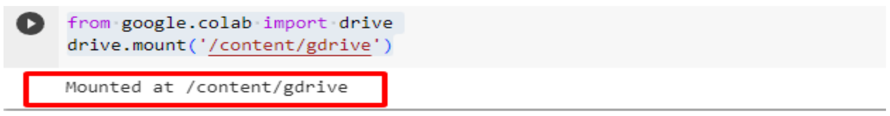


Permit this notebook to access your Google Drive files?

This notebook is requesting access to your Google Drive files. Granting access to Google Drive will permit code executed in the notebook to modify files in your Google Drive. Make sure that you review the notebook code prior to allowing this access.

**Figure 19.** Permission request for connecting to Google Drive. (Google Colab screen capture)

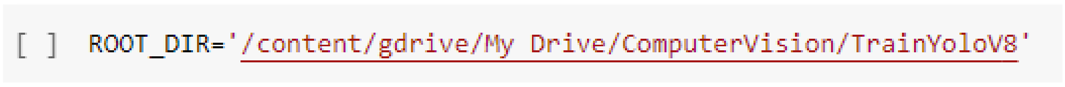
After signing in successfully on the Drive with the account that we used for creating necessary folder, the output in the terminal would be "Mounted at /content/gdrive".



```
from google.colab import drive
drive.mount('/content/gdrive')
```

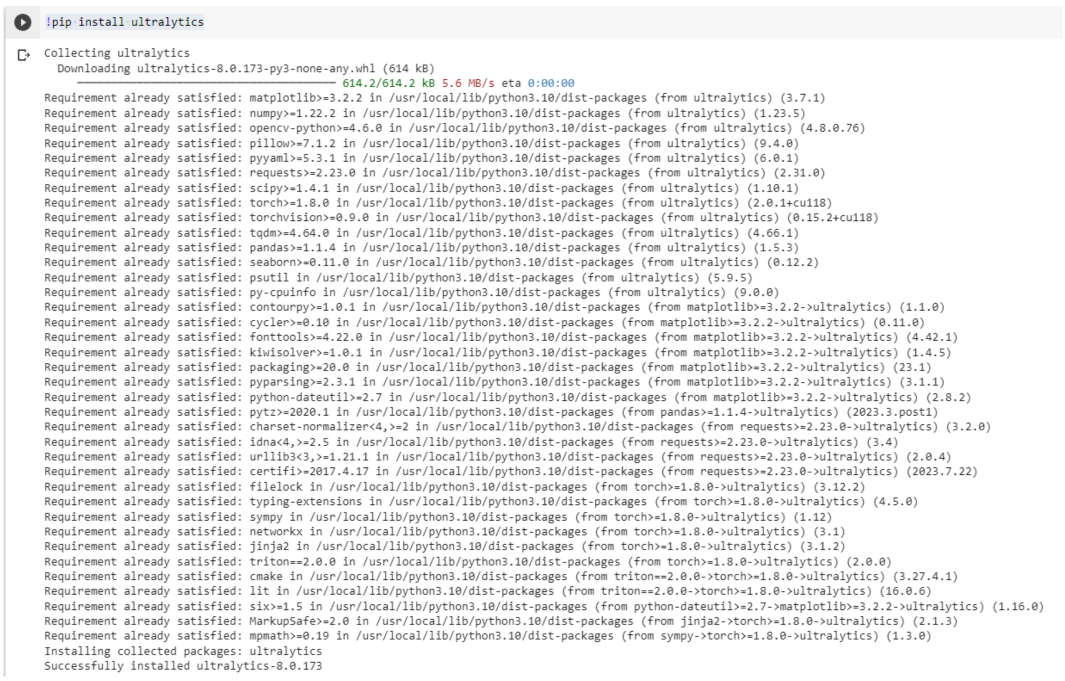
```
Mounted at /content/gdrive
```

**Figure 20.** Mounting to Google Drive successfully. (Google Colab screen capture)



```
[ ]  ROOT_DIR='/content/gdrive/My Drive/ComputerVision/TrainYoloV8'
```

**Figure 21.** Initialize 'ROOT_DIR' variable in Google Collaboratory. (Google Colab screen capture)



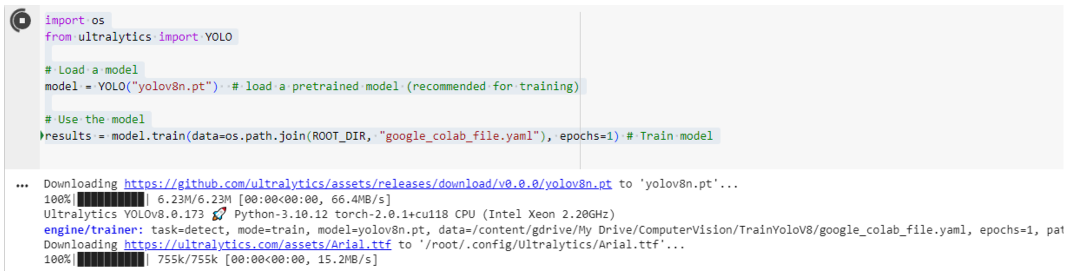
```
!pip install ultralytics
```

```
Collecting ultralytics
  Downloading ultralytics-8.0.173-py3-none-any.whl (614 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 614.2/614.2 kB 5.6 MB/s eta 0:00:00
Requirement already satisfied: matplotlib>=3.2.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (3.7.1)
Requirement already satisfied: numpy>=1.22.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.23.5)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.8.0.76)
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.4.0)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (6.0.1)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.31.0)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.10.1)
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.0.1+cu118)
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.15.2+cu118)
Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.66.1)
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.5.3)
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.12.2)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from ultralytics) (5.9.5)
Requirement already satisfied: py-cpuinfo in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (23.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.2.2->ultralytics) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->ultralytics) (2023.3.post1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2023.7.22)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.12.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch>=1.8.0->ultralytics) (3.27.4.1)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch>=1.8.0->ultralytics) (16.0.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.2.2->ultralytics) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.8.0->ultralytics) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.8.0->ultralytics) (1.3.0)
Installing collected packages: ultralytics
Successfully installed ultralytics-8.0.173
```

**Figure 22.** Installing ‘ultralytics’ library in Google Colab Python notebook. (Google Colab screen capture)

Modify the YOLOv8 training code to work within the Google Colab environment like the figure 23.

Overally, this cell loads a pre-trained YOLO model specifies a custom dataset and training settings through a YAML configuration file, and then starts the training process for one epoch.



```
import os
from ultralytics import YOLO

# Load a model
model = YOLO("yolov8n.pt")  # load a pretrained model (recommended for training)

# Use the model
results = model.train(data=os.path.join(ROOT_DIR, "google_colab_file.yaml"), epochs=1) # Train model
```

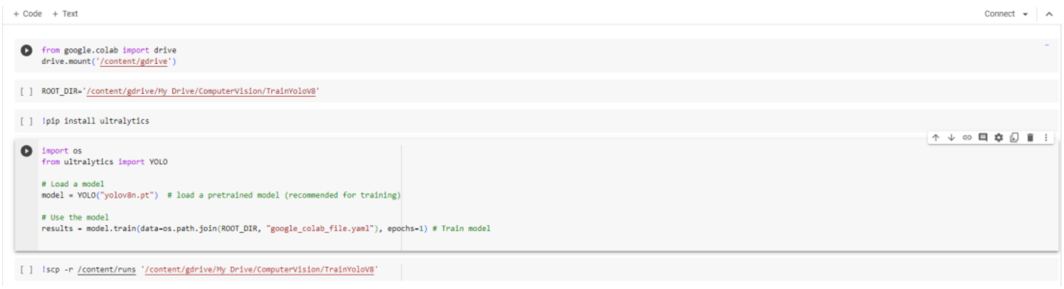
```
Downloading https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8n.pt to 'yolov8n.pt'...
100%|██████████| 6.23M/6.23M [00:00<00:00, 66.4MB/s]
Ultralytics YOLOv8.0.173 🚀 Python-3.10.12 torch-2.0.1+cu118 CPU (Intel Xeon 2.20GHz)
engine/trainer: task=detect, mode=train, model=yolov8n.pt, data=/content/gdrive/My Drive/ComputerVision/TrainYoloV8/google_colab_file.yaml, epochs=1, pat
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100%|██████████| 755k/755k [00:00<00:00, 15.2MB/s]
```

**Figure 23.** Importing `os` module and integrate `ultralytics` to train the model. (Google Colab screen capture)

After successfully training data, export the model file and use the trained model with the implementation in the following cell:

```
!scp -r /content/runs '/content/gdrive/My Drive/ComputerVision/TrainYoloV8'
```

Generally, the Google Collaboratory Python notebook should look like as the following figure:



```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
ROOT_DIR='/content/gdrive/My Drive/ComputerVision/TrainYoloV8'
```

```
!pip install ultralytics
```

```
import os
from ultralytics import YOLO

# Load a model
model = YOLO("yolov8n.pt")  # load a pretrained model (recommended for training)

# Use the model
results = model.train(data=os.path.join(ROOT_DIR, "google_colab_file.yaml"), epochs=1) # Train model
```

```
!scp -r /content/runs '/content/gdrive/My Drive/ComputerVision/TrainYoloV8'
```

**Figure 24.** Google Collaboratory Python notebook for training YOLOv8 AI model. (Google Colab screen capture)

For a detailed visit the Google Python notebook that this thesis provides in the list of reference (Nguyen 2023g).

The clarity on creating a pilot project to train an AI model raises the next question is: "How can a software engineer initiate an AI camera system pilot project to gain momentum?" The next sub chapter will dig deeper into this inquiry using the pre-trained YOLOv3 model to provide comprehensive insights into its functionality.

## 3.2 Intrusion Warning Project

### What is intrusion warning?

Intrusion Warning system in the context of this thesis is a compact AI Camera System, which firstly detect an object, specific here is person object defined by a pre-trained YOLOv3 model (Keineahnung2345 2023). Additionally, the system enables the drawing of a polygon area to allocate the restricted zone via the monitored camera. It dynamically updates the warning status whenever the centroid of the detected object resides within the drawn polygon area.

Moreover, the integration of a Telegram bot API into our application ensures notifications are sent whenever the system detects a person within the restricted area.

This application is not limited to the security industry; it has versatile applications, including marketing strategy to understand customer product preferences. Its utility extends to various industries such as Agriculture, Health Care, Military, and more.

The workflow of the Intrusion Warning project is illustrated in Figure 24, depicting a clear representation of the project's intricacies.

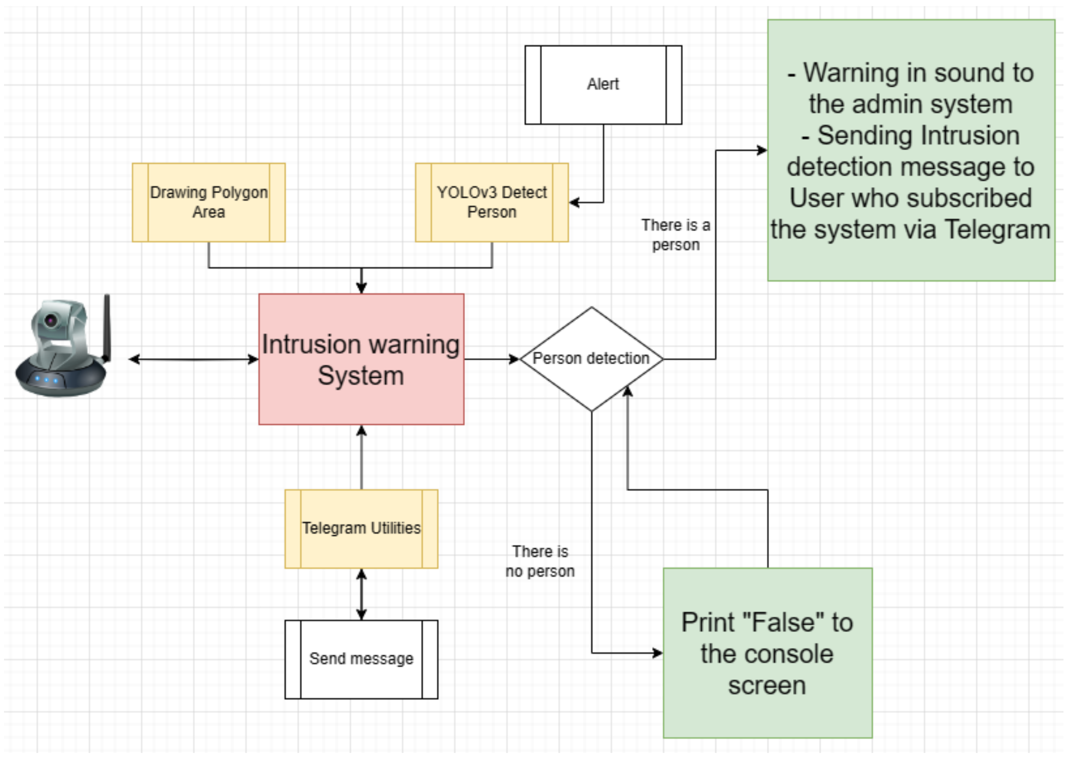

Alert
Drawing Polygon Area
YOLOv3 Detect Person
- Warning in sound to the admin system
- Sending Intrusion detection message to User who subscribed the system via Telegram
There is a person
Intrusion warning System
Person detection
Telegram Utilities
There is no person
Print "False" to the console screen
Send message

**Figure 25.** Workflow of Intrusion Warning project. (Draw.io screen capture)

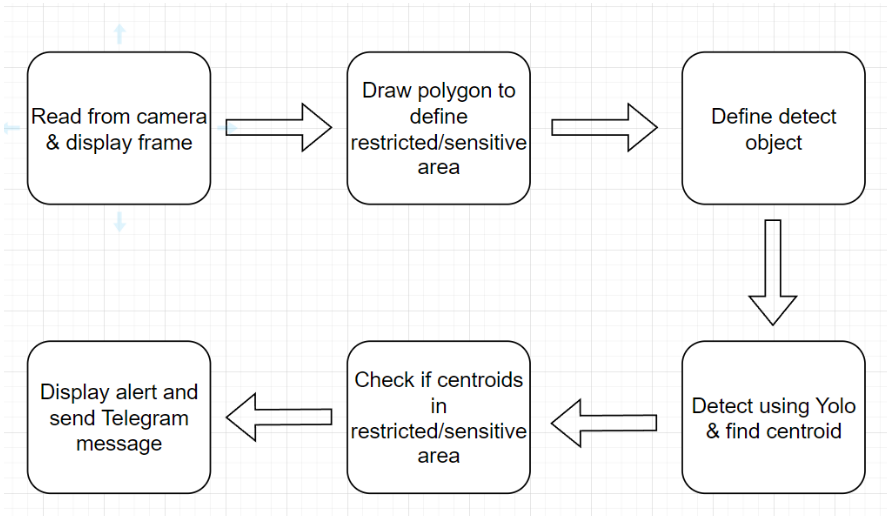

Read from camera & display frame
Draw polygon to define restricted/sensitive area
Define detect object
Detect using Yolo & find centroid
Check if centroids in restricted/sensitive area
Display alert and send Telegram message

**Figure 26.** Pipeline of the Intrusion Warning project. (Draw.io screen capture)

The centroid is a geometric property that represents the centre of mass or point of balance of an object. For shapes like the human body, the centroid is the average position of all the individual points that make up the object.

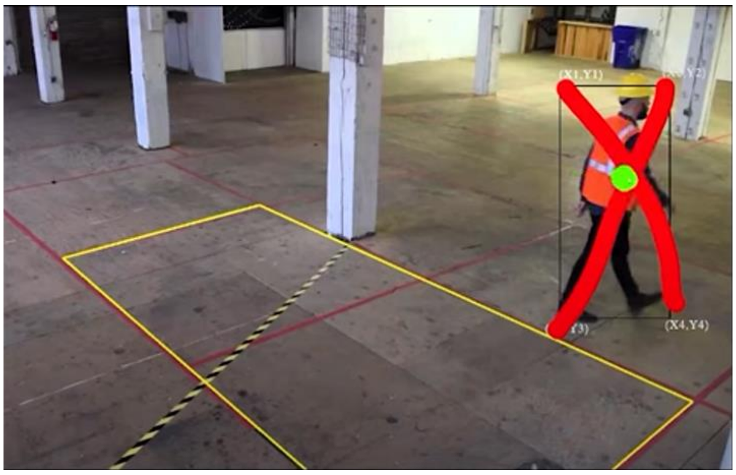


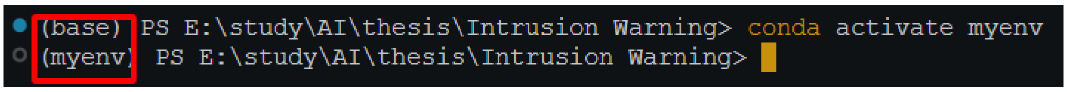
**Figure 27.** Centroid of human example. (Ngo 2022)

**Building Intrusion Warning System application (Simulation):**

Source code of the project could be found on the list of references at the end of this thesis. (Nguyen 2023e).

**Getting Started:**

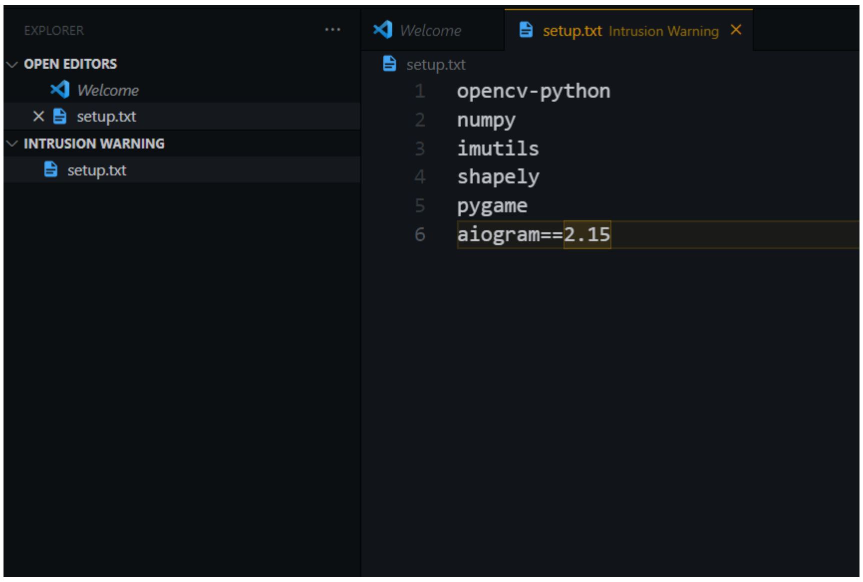
Create an empty folder named ´Intrusion Warning´ in Visual Studio Code. Open new Terminal and enter the command *´conda activate myenv´* to activate the virtual environment.



```
(base) PS E:\study\AI\thesis\Intrusion Warning> conda activate myenv
(myenv) PS E:\study\AI\thesis\Intrusion Warning>
```

**Figure 28.** Activating Miniconda virtual environment. (Terminal screen capture)

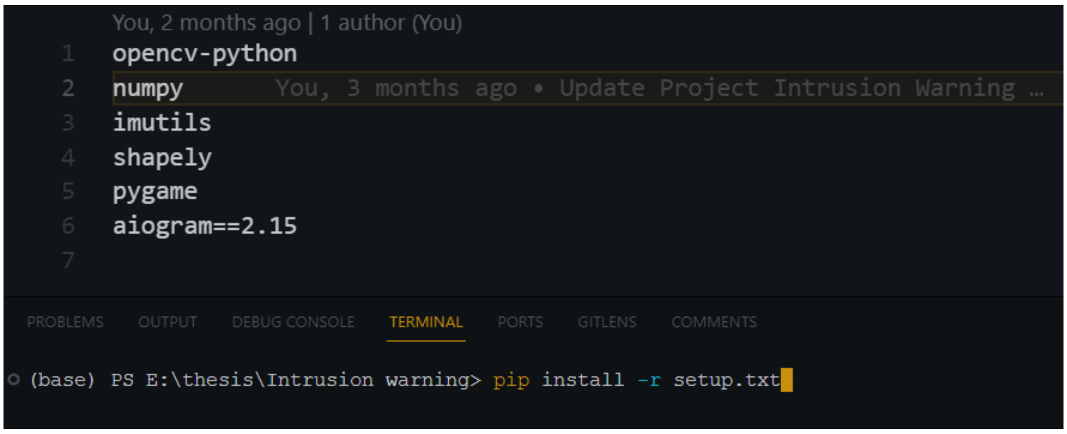
Create a ´setup.txt´ file indicating the required external libraries for the project.



```
opencv-python
numpy
imutils
shapely
pygame
aiogram==2.15
```

**Figure 29.** Required external libraries for our project. (Source code screen capture)

Run the command *'pip install -r setup.txt'* to install all the needed libraries.

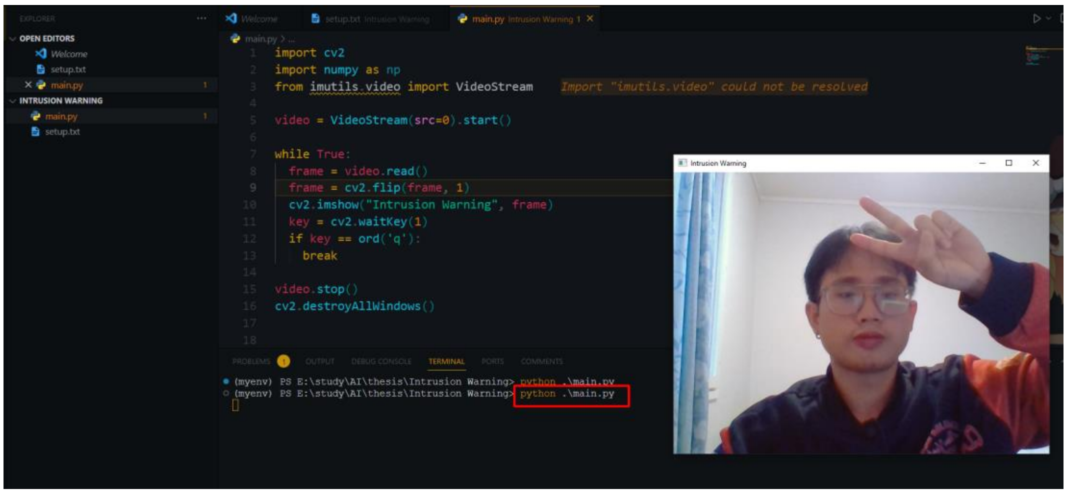

You, 2 months ago | 1 author (You)
You, 3 months ago • Update Project Intrusion Warning …
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GITLENS
COMMENTS

```
opencv-python
numpy
imutils
shapely
pygame
aiogram==2.15
```

```
(base) PS E:\thesis\Intrusion warning> pip install -r setup.txt
```

**Figure 30.** Installing necessary libraries for Intrusion Warning project (screen capture)

Ensure that the application can connect to the real camera. This report uses the personal webcam for this purpose via the ´main.py´ file with the following initial code block:



```
1  import cv2
2  import numpy as np
3  from imutils.video import VideoStream    Import "imutils.video" could not be resolved
4
5  video = VideoStream(src=0).start()
6
7  while True:
8    frame = video.read()
9    frame = cv2.flip(frame, 1)
10   cv2.imshow("Intrusion Warning", frame)
11   key = cv2.waitKey(1)
12   if key == ord('q'):
13     break
14
15 video.stop()
16 cv2.destroyAllWindows()
17
18
```

```
(myenv) PS E:\study\AI\thesis\Intrusion Warning> python .\main.py
(myenv) PS E:\study\AI\thesis\Intrusion Warning> python .\main.py
```

**Figure 31.** Use project to turn on webcam. (Source code screen capture)

*Code Overview in 'main.py' file:*

*Line 1-3:* Import necessary libraries.

*Line 5:* Start a video stream from the default camera (src = 0 is laptop webcam as default).

*Line 7-13:* Enter a loop to continuously capture and display video frames 'while True'**.**

*Line 15:* Stop the video stream using 'video.stop()'.

*Line 16:* Close all OpenCV windows with 'cv2.destroyAllWindows()'.

**Using the YOLOv3 AI model for object detection:**

Next, download the pre-trained model from Pre-trained YOLOv3 model for human detection (Nguyen 2023f). Then put the model folder into the current project.

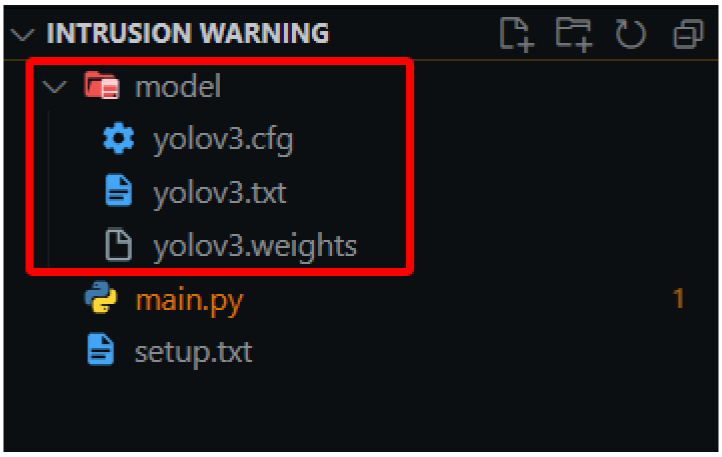
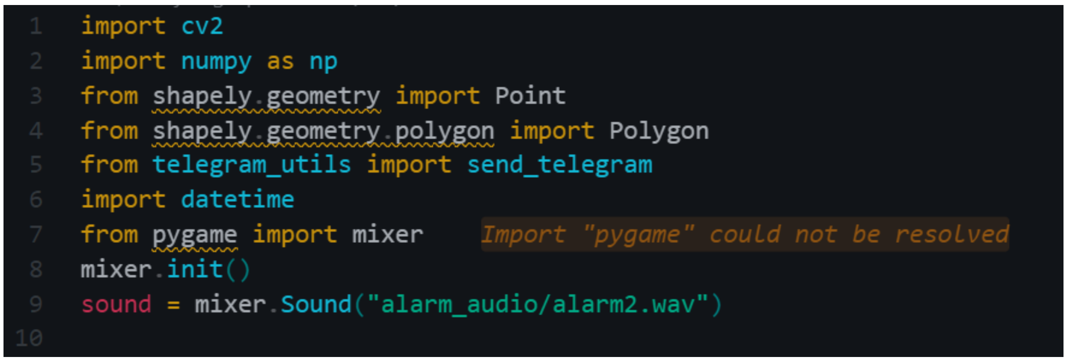


**Figure 32.** YOLOv3 pre-trained model folder structure. (Visual Studio Code screen capture)

**Implement of yolodetect.py:**

In this step, create a new file *yolodetect.py* and copy the whole content in the file yolodetect.py of intrusion warning project (Nguyen 2023e).

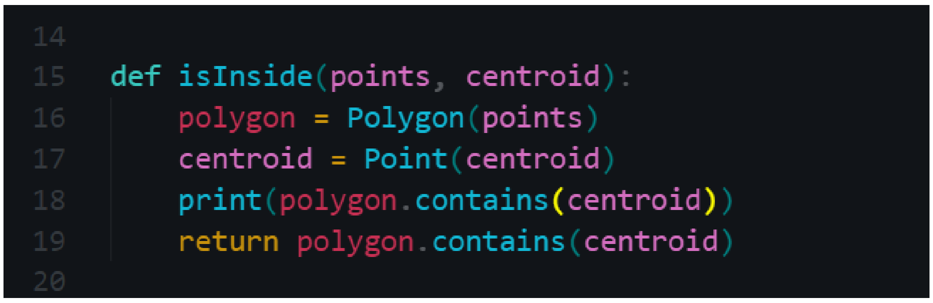
The provided Python code is part of a program designed for real-time object detection, specifically for detecting humans using YOLO model. It integrates various functionalities, including object detection, drawing bounding boxes, calculating centroids, and triggering alerts.



```
import cv2
import numpy as np
from shapely.geometry import Point
from shapely.geometry.polygon import Polygon
from telegram_utils import send_telegram
import datetime
from pygame import mixer     Import "pygame" could not be resolved
mixer.init()
sound = mixer.Sound("alarm_audio/alarm2.wav")

```

**Figure 33.** Importing external libraries for file 'yolodetect.py'. (Source code screen capture)

Take a look at the function *´isInside(points, centroid)´* where will check if the centroid of detected person is inside a specified polygon area using Shapely geometry.

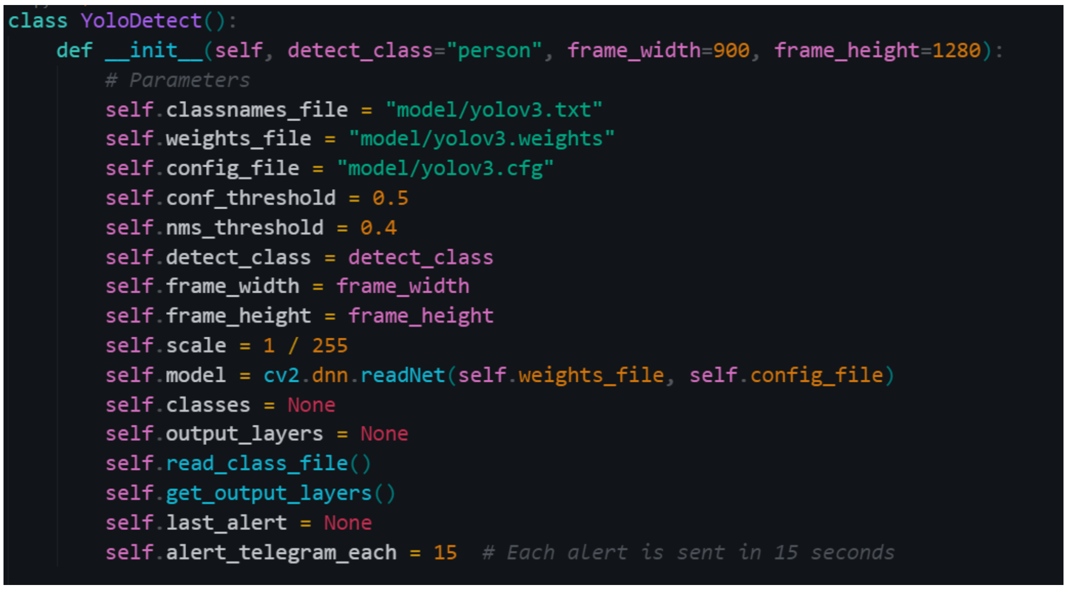


```python

def isInside(points, centroid):
    polygon = Polygon(points)
    centroid = Point(centroid)
    print(polygon.contains(centroid))
    return polygon.contains(centroid)

```

**Figure 34.** Defining the 'isInside(point, centroid)' function. (Source code screen capture)

On the following step, initializes parameters, such as YOLO model files, confidence thresholds, and frame dimensions…

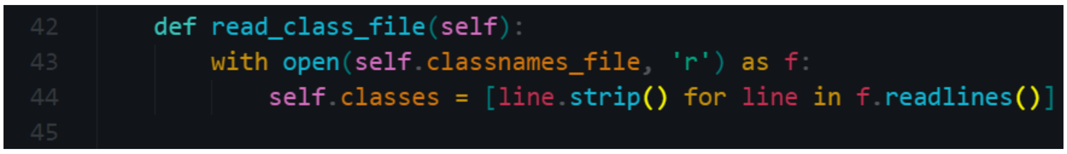
*Note:* Modify the sending telegram alert in each 15 second (in line *`alert_telegram_each = 15`*) so that whenever an object is detected in the restricted area. This help to avoid sending notification messages uncontrollably repeated, causing negative impacts on server resources as well as the inconvenience of users who subscribed for the service of the application.



```python
class YoloDetect():
    def __init__(self, detect_class="person", frame_width=900, frame_height=1280):
        # Parameters
        self.classnames_file = "model/yolov3.txt"
        self.weights_file = "model/yolov3.weights"
        self.config_file = "model/yolov3.cfg"
        self.conf_threshold = 0.5
        self.nms_threshold = 0.4
        self.detect_class = detect_class
        self.frame_width = frame_width
        self.frame_height = frame_height
        self.scale = 1 / 255
        self.model = cv2.dnn.readNet(self.weights_file, self.config_file)
        self.classes = None
        self.output_layers = None
        self.read_class_file()
        self.get_output_layers()
        self.last_alert = None
        self.alert_telegram_each = 15  # Each alert is sent in 15 seconds
```

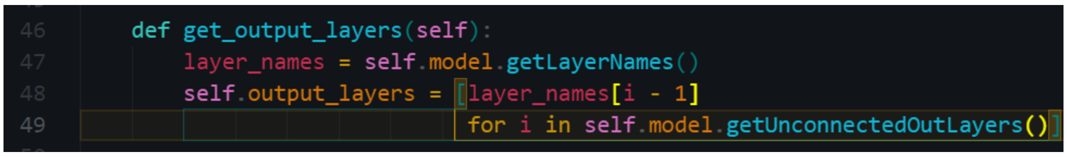
**Figure 35.** Defining a class ‘YoloDetect’ class for object detection. (Source code screen capture)

The upcoming lines of code show the implementation of several foundational configuration of the YOLO model via figure 36 and figure 37.



```python
    def read_class_file(self):
        with open(self.classnames_file, 'r') as f:
            self.classes = [line.strip() for line in f.readlines()]
```

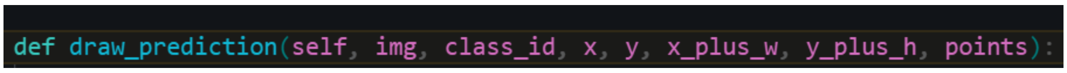
**Figure 36.** Defining ‘read_class_file’ function in the ‘YoloDetect’ class. (Source code screen capture)



```python
    def get_output_layers(self):
        layer_names = self.model.getLayerNames()
        self.output_layers = [layer_names[i - 1]
                              for i in self.model.getUnconnectedOutLayers()]
```

**Figure 37.** Defining ‘get_output_layers’ function in the ‘YoloDetect’ class. (Source code screen capture)

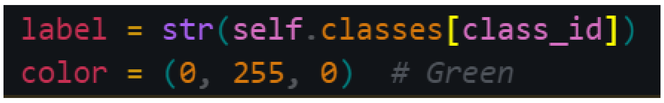
Next, the *´draw_prediction´* function inside the *´YoloDetect´* class is responsible for drawing predictions on a given frame based on the results of object detection. It specifically draws bounding boxes around detected objects, labels them, calculates the centroid of each bounding box, and performs additional actions when a centroid is inside a specified polygon area.



```
def draw_prediction(self, img, class_id, x, y, x_plus_w, y_plus_h, points):
```

**Figure 38.** Defining 'draw_prediction' function in the 'YoloDetect' class.

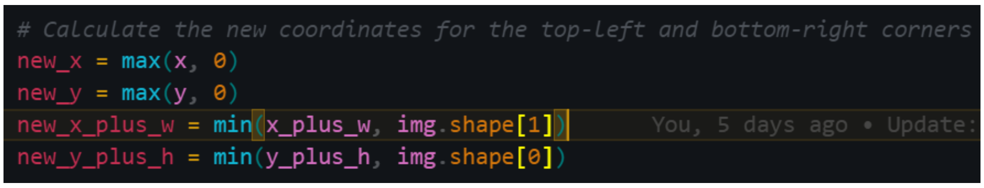
Inside '*draw_prediction´* function, retrieve the class name based on the *'class_id'* and define the *'color'* for drawing the bounding box (in this case, green).



```
label = str(self.classes[class_id])
color = (0, 255, 0)  # Green
```

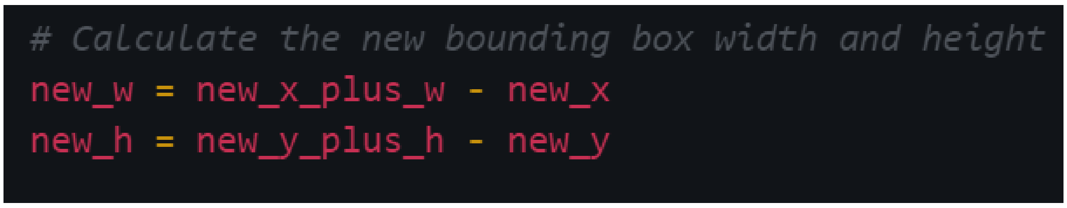
**Figure 39.** Declaring variables 'label' and 'color' inside the 'draw_prediction' function. (Source code screen capture)

Additionally, '*draw_prediction´* function ensure that the bounding box coordinates *(´new_x´, ´new_y´, ´new_x_plus_w´, ´new_y_plus_h´)* do not extend beyond the image boundaries to avoid errors when drawing.



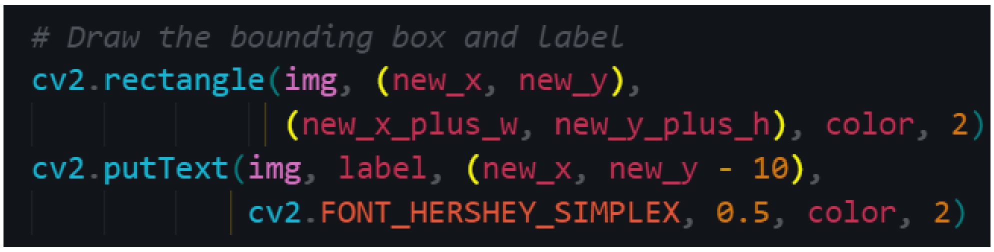
```
# Calculate the new coordinates for the top-left and bottom-right corners
new_x = max(x, 0)
new_y = max(y, 0)
new_x_plus_w = min(x_plus_w, img.shape[1])        You, 5 days ago • Update:
new_y_plus_h = min(y_plus_h, img.shape[0])
```

**Figure 40.** Declaring variables for Calculating the coordinates inside the 'draw_prediction' function. (Source code screen capture)



```
# Calculate the new bounding box width and height
new_w = new_x_plus_w - new_x
new_h = new_y_plus_h - new_y
```

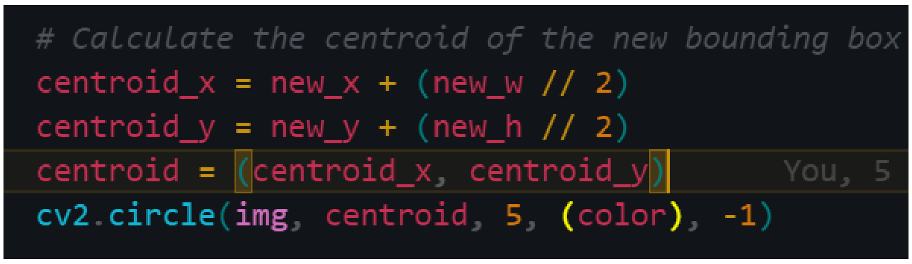
**Figure 41.** Declaring variables for Calculating the width and height of the adjusted bounding box inside the 'draw_prediction' function. (Source code screen capture)



```python
# Draw the bounding box and label
cv2.rectangle(img, (new_x, new_y),
                    (new_x_plus_w, new_y_plus_h), color, 2)
cv2.putText(img, label, (new_x, new_y - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
```

**Figure 42.** Drawing the bounding box inside the ‘draw_prediction’ function.

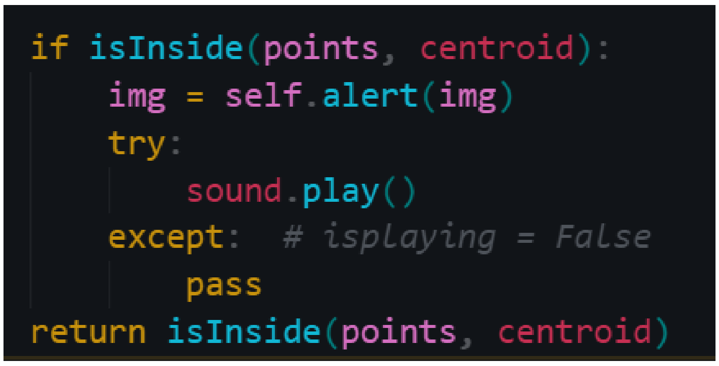
Calculate the centroid of the new bounding box by finding the midpoint of its width and height and mark this centroid point with a small green circle.



```python
# Calculate the centroid of the new bounding box
centroid_x = new_x + (new_w // 2)
centroid_y = new_y + (new_h // 2)
centroid = (centroid_x, centroid_y)          You, 5
cv2.circle(img, centroid, 5, (color), -1)
```

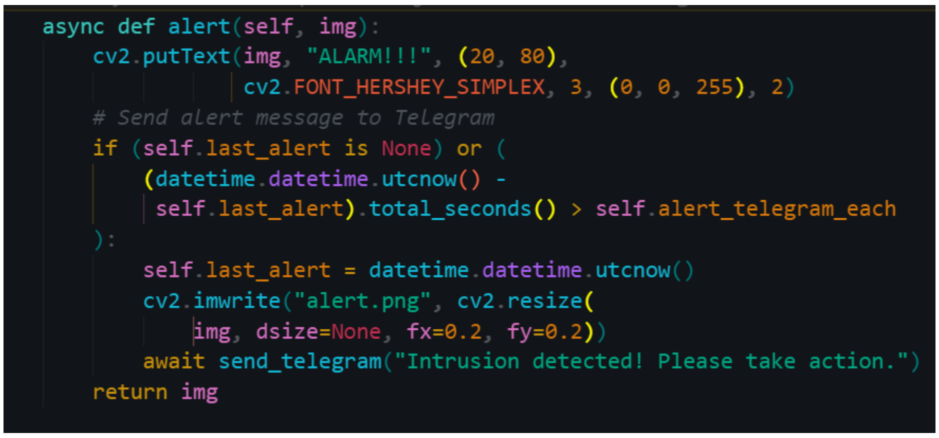
**Figure 43.** Calculating the centroid point.

Then, the `*isInside*` function is called to check if the centroid of the bounding box is inside a specified polygon area (covered by `*points*`). If the centroid is inside the polygon, an alert is triggered, and the alert sound will be played.



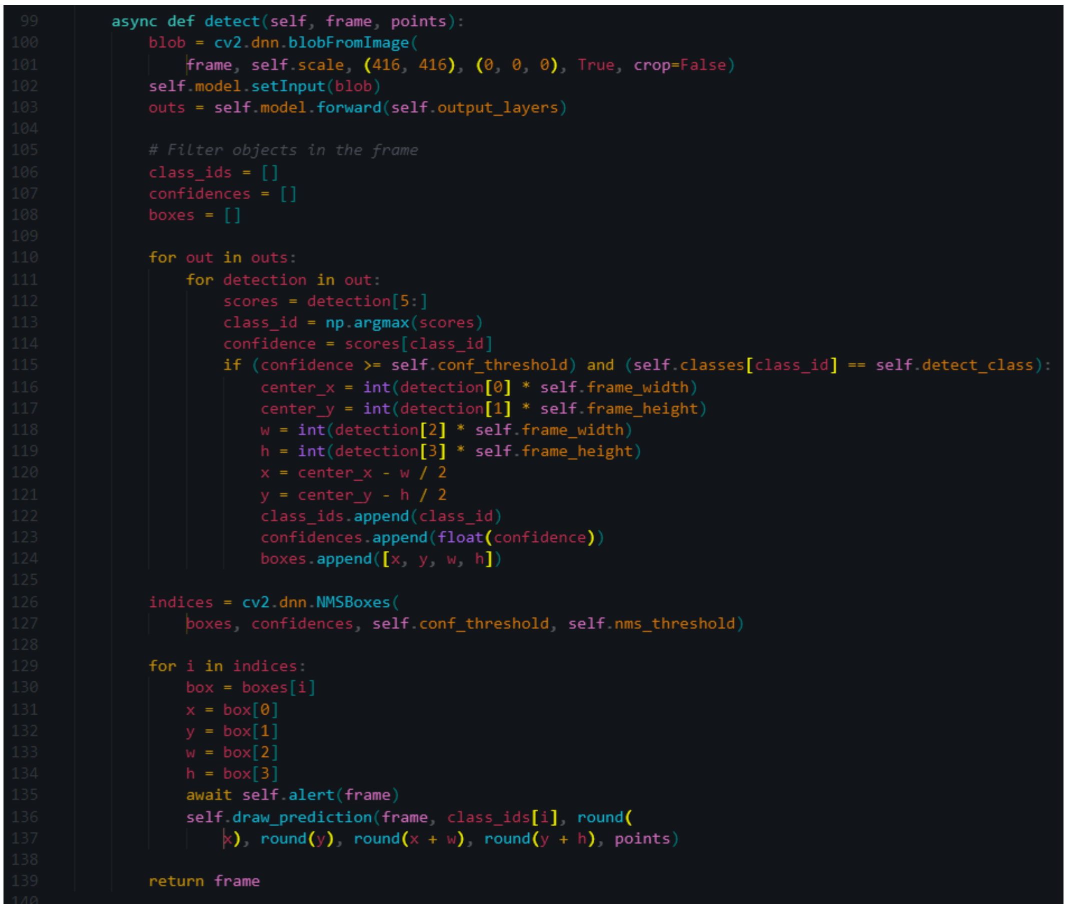
```python
if isInside(points, centroid):
    img = self.alert(img)
    try:
        sound.play()
    except:  # isplaying = False
        pass
return isInside(points, centroid)
```

**Figure 44.** Conditional statement for sound alerting. (Source code screen capture)



```python
async def alert(self, img):
    cv2.putText(img, "ALARM!!!", (20, 80),
                cv2.FONT_HERSHEY_SIMPLEX, 3, (0, 0, 255), 2)
    # Send alert message to Telegram
    if (self.last_alert is None) or (
        (datetime.datetime.utcnow() -
         self.last_alert).total_seconds() > self.alert_telegram_each
    ):
        self.last_alert = datetime.datetime.utcnow()
        cv2.imwrite("alert.png", cv2.resize(
            img, dsize=None, fx=0.2, fy=0.2))
        await send_telegram("Intrusion detected! Please take action.")
    return img
```

**Figure 45.** Defining ‘alert’ function in the ‘YoloDetect’ class for triggering an alert. (Source code screen capture)



```python
    async def detect(self, frame, points):
        blob = cv2.dnn.blobFromImage(
            frame, self.scale, (416, 416), (0, 0, 0), True, crop=False)
        self.model.setInput(blob)
        outs = self.model.forward(self.output_layers)

        # Filter objects in the frame
        class_ids = []
        confidences = []
        boxes = []

        for out in outs:
            for detection in out:
                scores = detection[5:]
                class_id = np.argmax(scores)
                confidence = scores[class_id]
                if (confidence >= self.conf_threshold) and (self.classes[class_id] == self.detect_class):
                    center_x = int(detection[0] * self.frame_width)
                    center_y = int(detection[1] * self.frame_height)
                    w = int(detection[2] * self.frame_width)
                    h = int(detection[3] * self.frame_height)
                    x = center_x - w / 2
                    y = center_y - h / 2
                    class_ids.append(class_id)
                    confidences.append(float(confidence))
                    boxes.append([x, y, w, h])

        indices = cv2.dnn.NMSBoxes(
            boxes, confidences, self.conf_threshold, self.nms_threshold)

        for i in indices:
            box = boxes[i]
            x = box[0]
            y = box[1]
            w = box[2]
            h = box[3]
            await self.alert(frame)
            self.draw_prediction(frame, class_ids[i], round(
                x), round(y), round(x + w), round(y + h), points)

        return frame
```

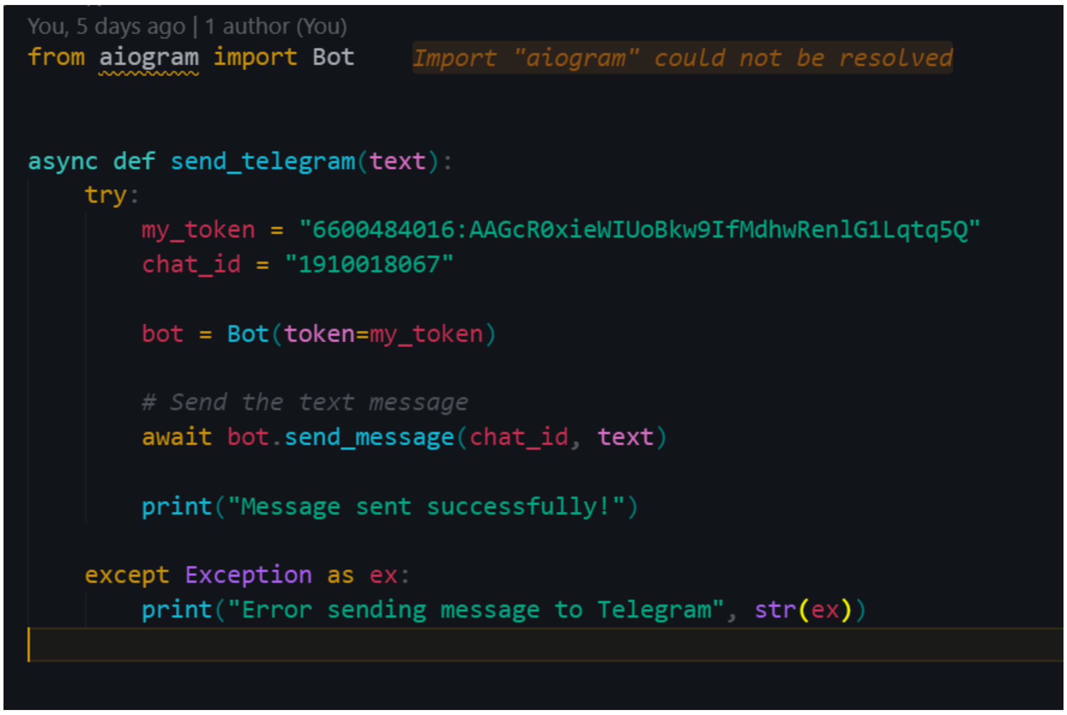
**Figure 46**. Defining ‘detect’ function in the ‘YoloDetect’ class. (Source code screen capture)

This comprehensive analysis of the *´YoloDetect.py´* file demonstrates its indispensable role in our project. The functions in the *´YoloDetect´* class are critical to the core functionality of the program. A deeper exploration of these functions will provide a deeper understanding of their implementation and importance in the class.

**Telegram Integration:**

After finishing the *'yolodetect.py'* file, follow the instruction of creating a telegram bot with BotFather. (Nguyen 2023c).

After following the given instruction and creating a Telegram bot, get Bot ID and User ID via telegram API. Then, create another file in the project named *'telegram_utils.py'* with the following content, it is notable to assign the User ID with 'my_token' variable and Bot ID with 'chat_id' variable.



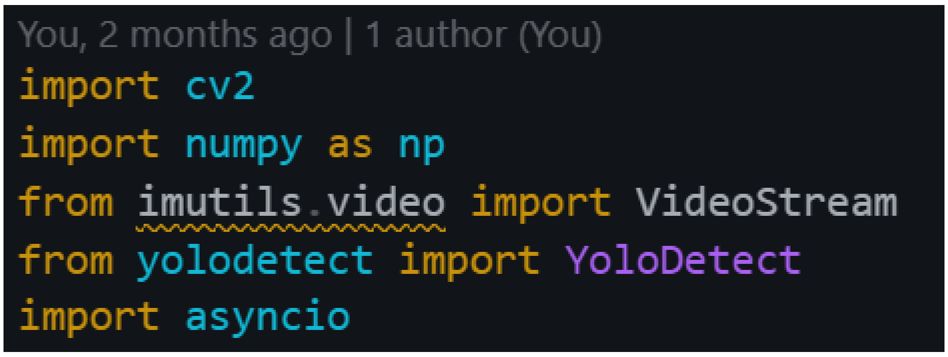
```
You, 5 days ago | 1 author (You)
from aiogram import Bot    Import "aiogram" could not be resolved


async def send_telegram(text):
    try:
        my_token = "6600484016:AAGcR0xieWIUoBkw9IfMdhwRenlG1Lqtq5Q"
        chat_id = "1910018067"

        bot = Bot(token=my_token)

        # Send the text message
        await bot.send_message(chat_id, text)

        print("Message sent successfully!")

    except Exception as ex:
        print("Error sending message to Telegram", str(ex))
```

**Figure 47.** Modifying the 'telegram_utils.py' file. (Source code screen capture)

**Implementation of ´main.py´:**

Finally, modify the *'main.py'* file like the relevant file in intrusion warning project which belong to this thesis (Nguyen 2023e).

First and foremost, import necessary libraries.

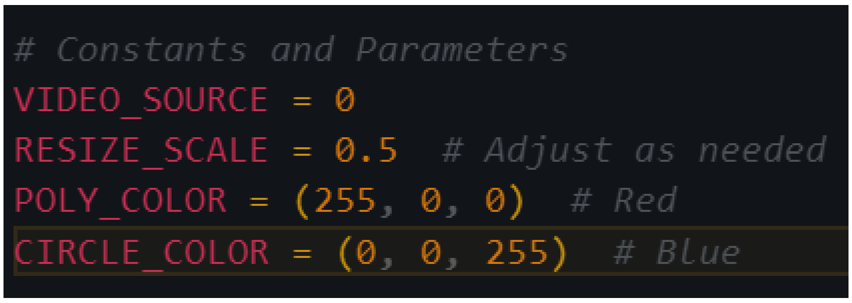


```
You, 2 months ago | 1 author (You)
import cv2
import numpy as np
from imutils.video import VideoStream
from yolodetect import YoloDetect
import asyncio
```

**Figure 48.** Import various libraries in file main.py. (Source code screen capture)

Next, the declaration of used constants and parameters in main.py should be explained as:

*'VIDEO_SOURCE'*: Specifies the video source (0 as default for laptop webcam).
*´RESIZE_SCALE'*: Sets the scale by which the video frame will be resized.
*'POLY_COLOR'* and *'CIRCLE_COLOR'*: Define the colours (Red and Blue) used for drawing the polygon and circle points.



```
# Constants and Parameters
VIDEO_SOURCE = 0
RESIZE_SCALE = 0.5  # Adjust as needed
POLY_COLOR = (255, 0, 0)  # Red
CIRCLE_COLOR = (0, 0, 255)  # Blue
```

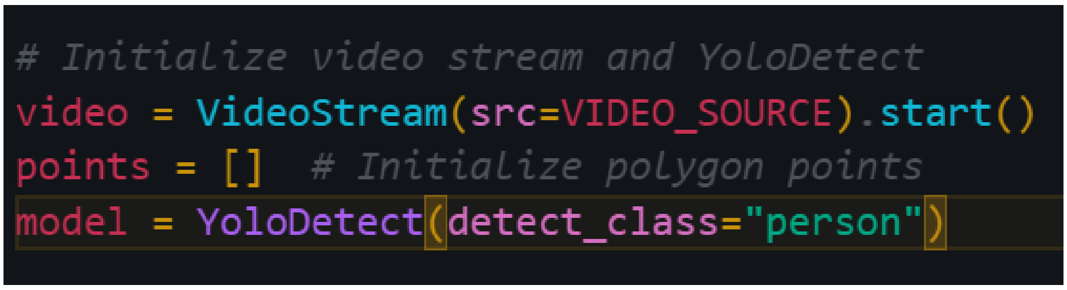
**Figure 49.** Declare necessary constants and parameters in file main.py. (Source code screen capture)

In the following step, the code block offers the initialization of video stream and Yolo model.

*'video'*: A video stream is initiated using the specified video source (0 as default camera, normally is the laptop webcam).

*'points'*: An empty list, points, is initialized to store polygon points.

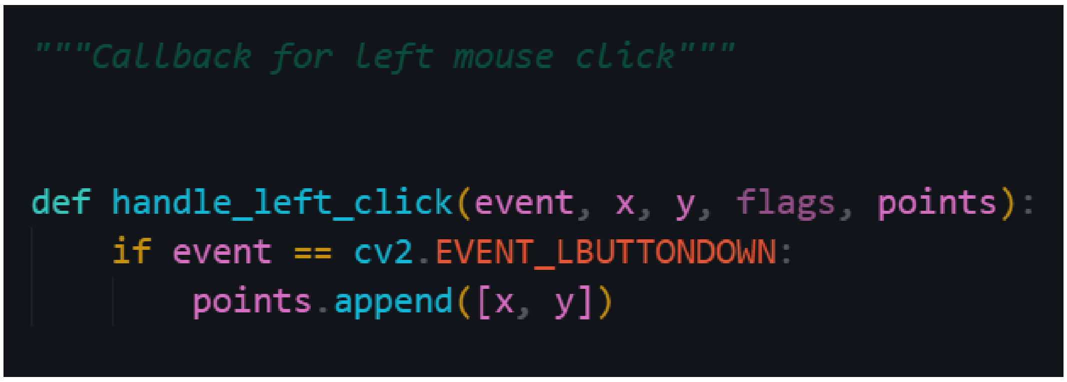
*'model'*: The YoloDetect model is initialized for detecting "person" objects.



```python
# Initialize video stream and YoloDetect
video = VideoStream(src=VIDEO_SOURCE).start()
points = []  # Initialize polygon points
model = YoloDetect(detect_class="person")
```

**Figure 50.** Initialize video stream and YoloDetect model and list for drawing points constants. (Source code screen capture)

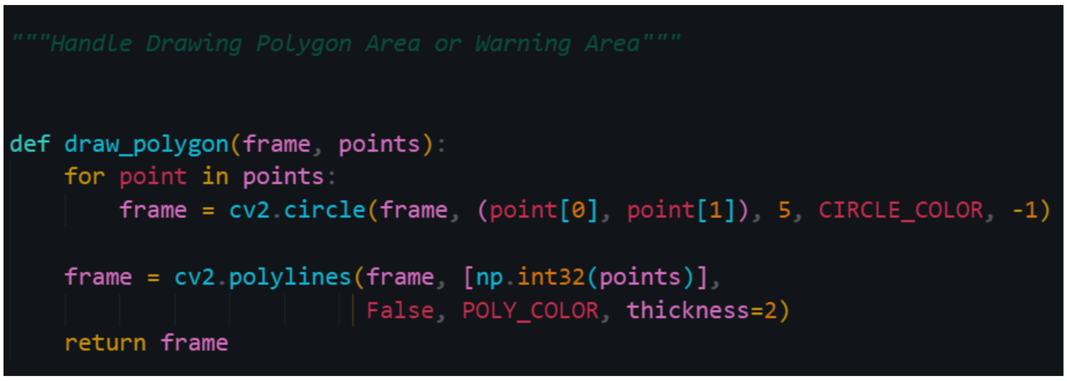
On the next lines, the 'handle_left_click' function is triggered when the left mouse button is clicked. This function records the x and y coordinates in the points list.



```python
"""Callback for left mouse click"""


def handle_left_click(event, x, y, flags, points):
    if event == cv2.EVENT_LBUTTONDOWN:
        points.append([x, y])
```

**Figure 51.** Define 'handle_left_click' function in file main.py. (Source code screen capture)

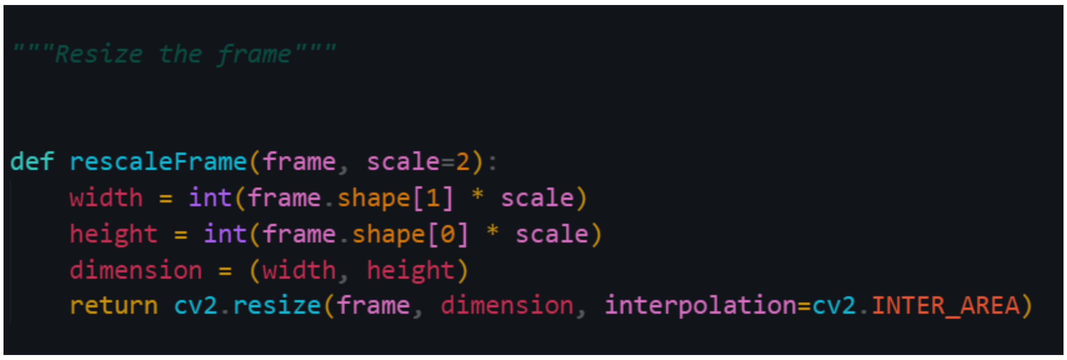
The 'draw_polygon(frame, points)' function draws a polygon based on the points in the points list and marks the *'points'* with circles. This function then returns the updated frame based on drawn points.



```python
"""Handle Drawing Polygon Area or Warning Area"""


def draw_polygon(frame, points):
    for point in points:
        frame = cv2.circle(frame, (point[0], point[1]), 5, CIRCLE_COLOR, -1)

    frame = cv2.polylines(frame, [np.int32(points)],
                          False, POLY_COLOR, thickness=2)
    return frame
```

**Figure 52.** Define 'draw_polygon' function in file main.py. (Source code screen capture)

Then, the 'rescaleFrame' function resizes the video frame based on the specified scale.



```python
"""Resize the frame"""


def rescaleFrame(frame, scale=2):
    width = int(frame.shape[1] * scale)
    height = int(frame.shape[0] * scale)
    dimension = (width, height)
    return cv2.resize(frame, dimension, interpolation=cv2.INTER_AREA)
```

**Figure 53.** Define 'rescaleFrame' function in main.py. (Source code screen capture)

Last but not least, the 'main()' function come up with several functionalities.

*'While True'*: A main loop processes video frames continuously.

*'frame'* and *'frame_resize'*: The frame is captured from the video stream and resized.

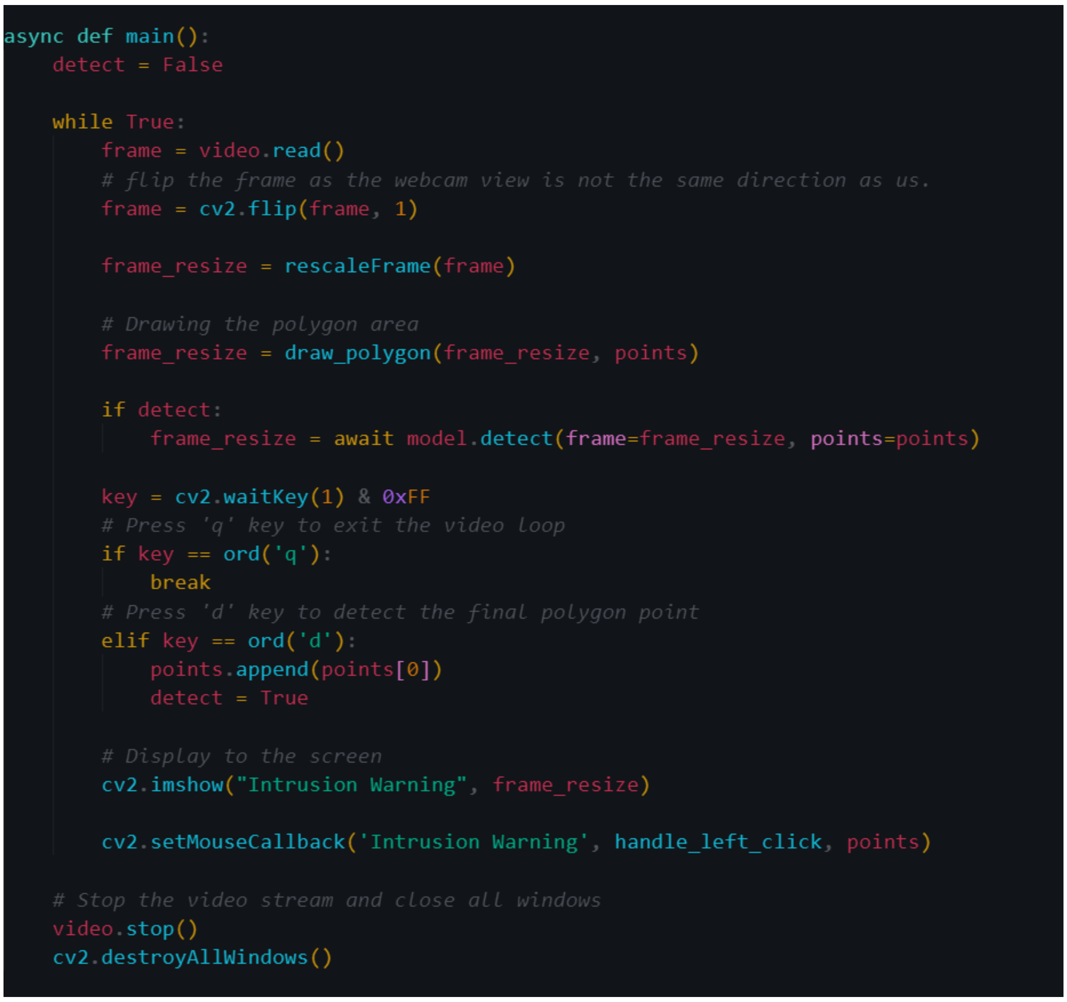
*'frame_resize = draw_polygon(frame_resize, points)'*: The drawn polygon area is superimposed on the frame.

*'if detect'*: If detect is True, the YoloDetect model is used to detect "person" objects within the polygon area.

*"key == ord('q') || key == ord('d')"*: User inputs ('q' for quit and 'd' for finalizing or drawing the polygon) are captured.

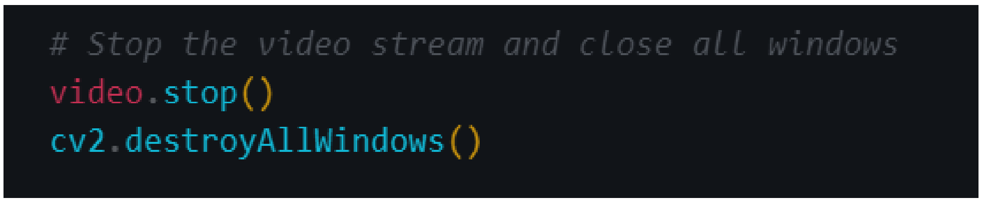
*'cv2.imshow'*: The frame is displayed with the polygon.

*'cv2.setMouseCallback'*: Mouse events are detected, and the points list is updated accordingly.



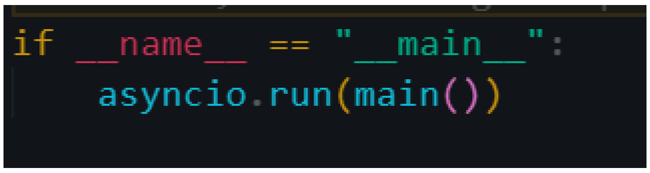
```
async def main():
    detect = False

    while True:
        frame = video.read()
        # flip the frame as the webcam view is not the same direction as us.
        frame = cv2.flip(frame, 1)

        frame_resize = rescaleFrame(frame)

        # Drawing the polygon area
        frame_resize = draw_polygon(frame_resize, points)

        if detect:
            frame_resize = await model.detect(frame=frame_resize, points=points)

        key = cv2.waitKey(1) & 0xFF
        # Press 'q' key to exit the video loop
        if key == ord('q'):
            break
        # Press 'd' key to detect the final polygon point
        elif key == ord('d'):
            points.append(points[0])
            detect = True

        # Display to the screen
        cv2.imshow("Intrusion Warning", frame_resize)

        cv2.setMouseCallback('Intrusion Warning', handle_left_click, points)

    # Stop the video stream and close all windows
    video.stop()
    cv2.destroyAllWindows()
```

**Figure 54.** Define main function in file main.py. (Source code screen capture)



```
# Stop the video stream and close all windows
video.stop()
cv2.destroyAllWindows()
```

**Figure 55.** Stop video stream feature in main.py. (Source code screen capture)



```
if __name__ == "__main__":
    asyncio.run(main())
```

**Figure 56.** main function execution in file main.py. (Source code screen capture)

Finally, the main function is executed within an asyncio event loop, enabling asynchronous functionality. The script waits for user inputs and displays the video stream.

**Start the application:**

After setting up all the code, go to the terminal at the path of root project, use the command as “python main.py” to execute the intrusion warning program.

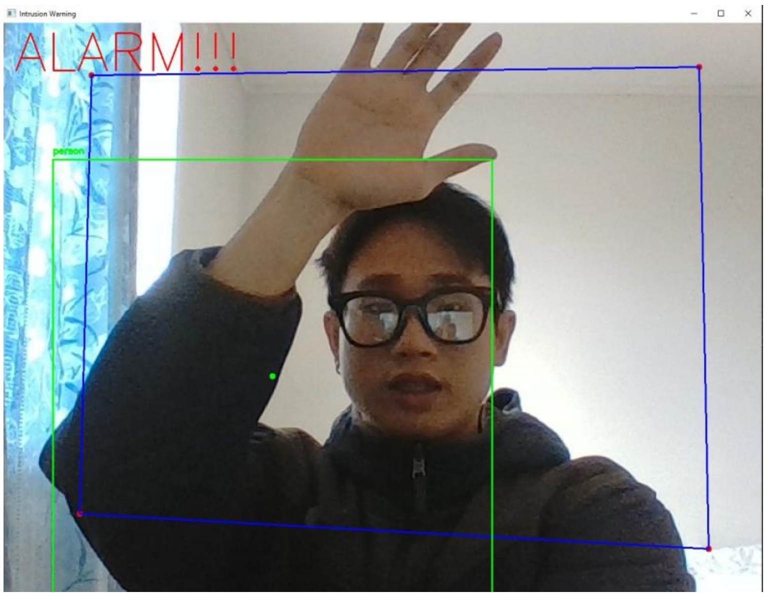
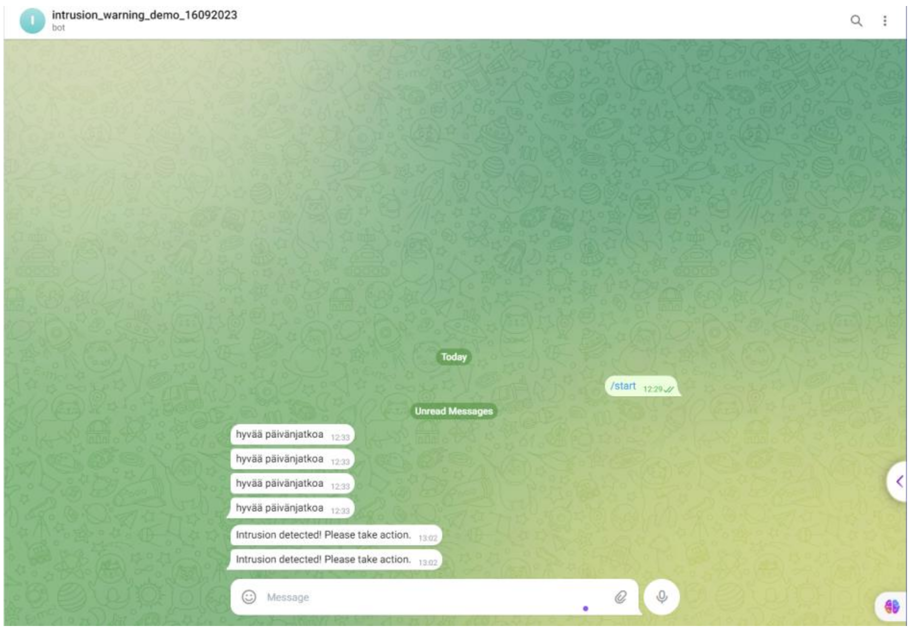


**Figure 57.** Detected person inside the restricted area successfully. (Screen capture)


intrusion_warning_demo_16092023
bot

Today

/start 12:29

Unread Messages

hyvää päivänjatkoa 12:33

hyvää päivänjatkoa 12:33

hyvää päivänjatkoa 12:33

hyvää päivänjatkoa 12:33

Intrusion detected! Please take action. 13:02

Intrusion detected! Please take action. 13:02

**Figure 58.** Our Intrusion Warning system sends telegram warning message successfully. (Telegram screen capture)

## 3.3 Guiding AI Growth: Personal Reflection on Understanding and Supervising the Development

Through the process of training an AI model and the simulation of Intrusion Warning Project. This thesis hopes to bring readers broader view in the industry of iVMS. However, for an AI product to gain a strong foothold in today's competitive market, continuous updating, optimization, and research are stories that deserve attention.

When a human being is born and grows up, from infancy to adulthood, he/she/they always needs mistakes and stumbles to become more and more mature. Besides, the logistics and support of parents and teachers around are also very important on this path.

And the same goes for a machine, humanity is the one who gradually creating them. In terms of social relationships, humans are now like the parents of artificial intelligence. Therefore, as a human we need to correct, teach, and supervise these children conscientiously, ready to point out the errors that this brainchild is making, so that the child can have better and better performance in its development lifecycle.

In the next chapter, this thesis will play the role as a supervisor and be ready to point out errors that the system is experiencing in an extremely serious way. An AI Testing Engineer is a metaphor for supervisor in the era of robotization.

# 4 TESTING STRATEGIES IN PRACTICE

After thoroughly examining the documents that describe the complexity of AI or APIs in any business or project (illustrated here with reference to the ITD Group), the AI testers should understand the processes that involve performing tests for the system, specifically focusing on the “intelligence” of the AI possessed by the organization. This requires a broader view of the testing processes related to the organization’s AI capabilities.

Furthermore, testers’ mindset needs to become more independent from the software construction and development team, which means that testers and the end users will be the one side and will not hesitate to point out defects that the general system or software is experiencing.

Test engineers will play a very important part in the stage of before publishing the final product, ensuring that the software will be carefully packaged to help increase competitiveness and customer satisfaction once the product is available on the marketplace.

In this chapter, the thesis will bring a preliminary look at testing a project for the purpose of “Support improving the recognition features” from using tools like Postman, several useful tips when using Postman and basic algorithms of writing Unit Tests. At the same time, this thesis will dig deeper into writing a script test for the “Object Tracing” feature to help organization having a better point of view to improve this module.

## 4.1 Manual Testing with Postman and Unit Testing with Pytest

To address confidentiality concerns with ITD, this section provides a general overview of applying manual testing and unit testing to AI APIs.

**Manual testing with Postman**

Manual testing with Postman is a straightforward approach for validating backend APIs developed by the IT team. The process involves ensuring that the system's responses align with the specifications documented in the ITD AI API Camera System Documentation (Nguyen et al. 2023d, 2-13).

Encoding image into Base64[2] format could be implemented by webapp services (Base64 image 2024).

To illustrate the manual testing process, it is necessary to follow an example in two simple steps.

*Step 1: Register a face to the AI recognition system*

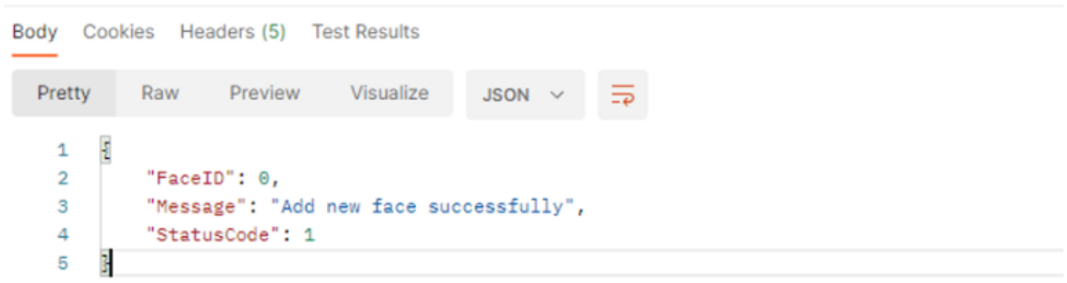
API URL of ITD: 8004/face_registration.

Image used. (ITD 2023a)

Request:

```
{
  "TimeSend": "2023-01-06T09:10:49,182",
  "Item": {
    "ImageData": "[Base64Image]",
    "FaceID": -1,
    "OpCode": 0
  }
}
```

[2] Base64 is a way to represent binary image data (like pictures) using only letters, numbers, and a few special characters. It's like turning a picture into text so that it can be easily included in documents, web pages, or data transmissions. This format is often used to embed images directly into HTML or CSS, making it a convenient way to handle images in various applications.



```
{
    "FaceID": 0,
    "Message": "Add new face successfully",
    "StatusCode": 1
}
```

**Figure 59.** Register a face to ITD's database successfully via API URL of ITD company 8004/face_registration. (Postman screen capture)

Response Data (ITD 2023b)

The response data confirms the successful registration of a “full-body” photo, aligning with our expectations.

*Step 2: Check if the software correctly verifies the registered person in the case of that person change his outfit.*

API URL of ITD: 8004/face_verification

Images Used (ITD 2023c)

Request:

```
{
    "TimeSend": "2023-01-06T09:10:49,182",
    "Item": {
        "ImageData": "A91h3jyjksi1",
        "FaceID": 0,
    }
}
```

Response Data 1 (ITD 2023d)

Response Data 2 (ITD 2023e)

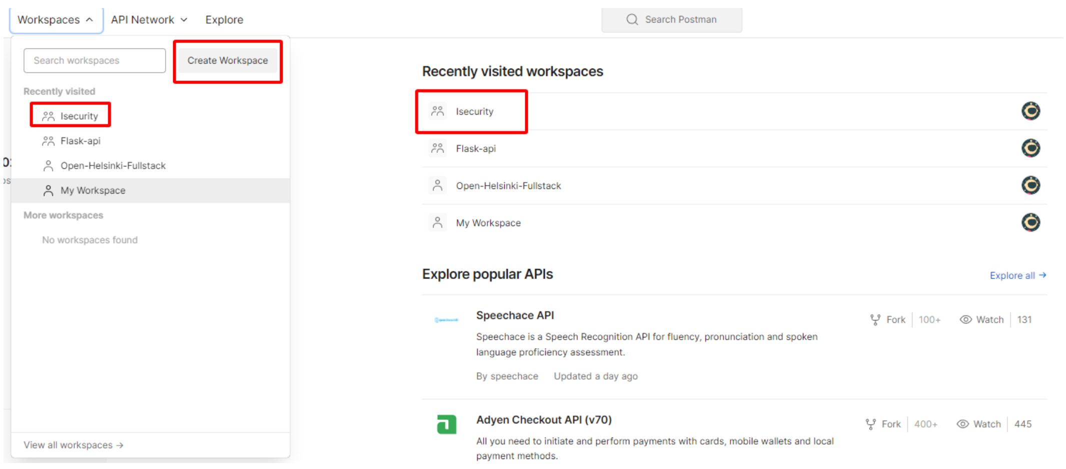
The results met expectations, with a successful confirmation of the same person, even with changes in outfit, achieving a verification rate of 74% and 69% for the respective images.

As a tester, exploring beyond defined successful cases is crucial. However, testing error cases, such as adjusting time formats or sending incorrect data, revealing system responses not caught by the backend… Reporting these findings aids in optimizing the software system.

**Postman environment configuration.**

Looking at some examples of the above information may seem quite vague to several newcomers. Therefore, in the following pages this thesis will focus on some setting up tips to make manual testing workflow become faster and more handy every time testers step into a similar project.

1. *Create a Workspace:*
   Maintain a clean and reusable workspace, like creating one named “Isecurity.”



Recently visited

- Isecurity
- Flask-api
- Open-Helsinki-Fullstack
- My Workspace

More workspaces

No workspaces found

View all workspaces →

Recently visited workspaces

- Isecurity
- Flask-api
- Open-Helsinki-Fullstack
- My Workspace

Explore popular APIs — Explore all →

**Speechace API** — Fork 100+ — Watch 131

Speechace is a Speech Recognition API for fluency, pronunciation and spoken language proficiency assessment.

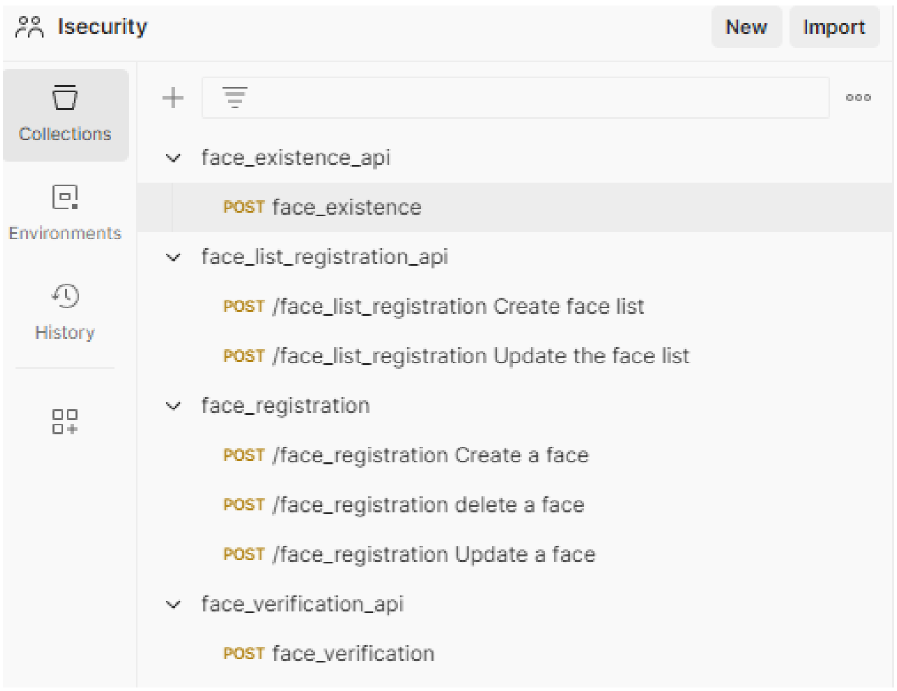
By speechace   Updated a day ago

**Adyen Checkout API (v70)** — Fork 400+ — Watch 445

All you need to initiate and perform payments with cards, mobile wallets and local payment methods.

**Figure 60.** Create a Workspace for testing project. (Postman screen capture)

2. *Allocate APIs into Collections:*



- face_existence_api
  - POST face_existence
- face_list_registration_api
  - POST /face_list_registration Create face list
  - POST /face_list_registration Update the face list
- face_registration
  - POST /face_registration Create a face
  - POST /face_registration delete a face
  - POST /face_registration Update a face
- face_verification_api
  - POST face_verification

**Figure 61.** Allocating each API into separate collections with Postman. (Postman screen capture)

3. *Setting up the URL Environment:*

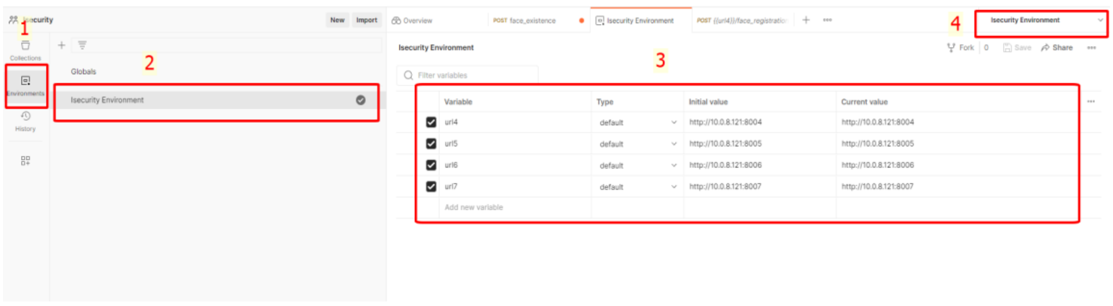
   Configuring the URL environment provides a shortcut for connecting with the organization’s API address. Follow these steps:

   (1) Access to Environments.

(2) Choose environment that we will use, if there is no environment yet, create one by clicking on the `+` button.

(3) Configure the working environment.

(4) Select the working environment.

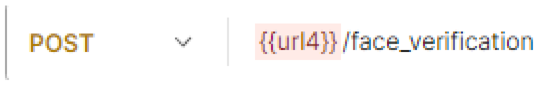


Isecurity Environment

| | Variable | Type | Initial value | Current value |
|---|---|---|---|---|
| ✓ | url4 | default | http://10.0.8.121:8004 | http://10.0.8.121:8004 |
| ✓ | url5 | default | http://10.0.8.121:8005 | http://10.0.8.121:8005 |
| ✓ | url6 | default | http://10.0.8.121:8006 | http://10.0.8.121:8006 |
| ✓ | url7 | default | http://10.0.8.121:8007 | http://10.0.8.121:8007 |

**Figure 62.** Setting up the url environment with Postman. (Postman screen capture)

After setting up the URL environment, use the shortcut `*{{name_of_url}}*` to quickly access the API address, for example `*{{url4}}*`.

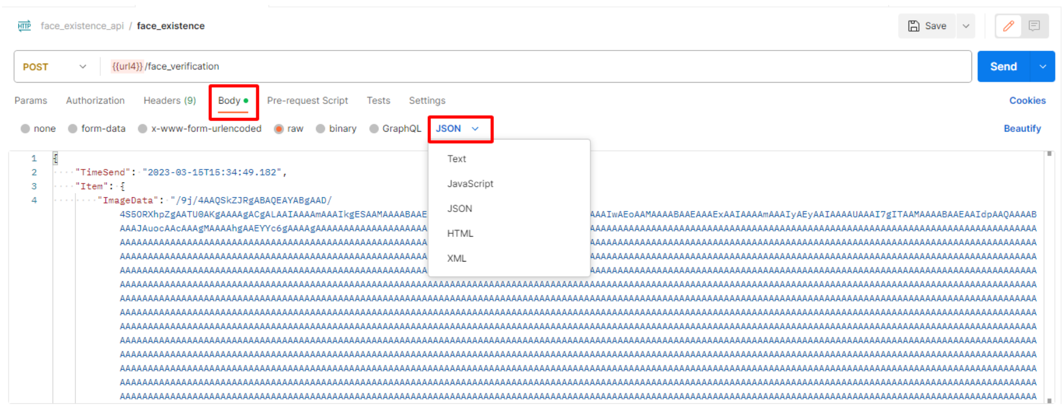


{{url4}} /face_verification

**Figure 63.** Shortcut for the URL address. (Postman screen capture)

4. *Configure request message:*

   Follow the JSON format specified in the AI description documentation (Nguyen et al.2023d, 2-13). Navigate to Body, then change the format to JSON.



face_existence_api / face_existence

POST {{url4}}/face_verification

Params | Authorization | Headers (9) | Body ● | Pre-request Script | Tests | Settings | Cookies

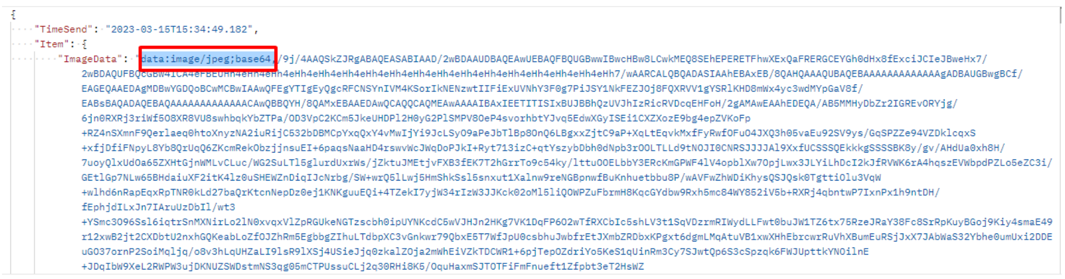
none | form-data | x-www-form-urlencoded | (●) raw | binary | GraphQL | JSON | Beautify

```
{
    "TimeSend": "2023-03-15T15:34:49.182",
    "Item": {
        "ImageData": "/9j/4AAQSkZJRgABAQEAYABgAAD/
```

4S5ORXhpZgAATU0AKgAAAAgACgALAAIAAAAmAAAIkgESAAMAAAABAAE … AAAIwAEoAAMAAAABAAEAAAExAAIAAAAmAAAIyAEyAAIAAAAUAAAI7gITAAMAAAABAAEAAIdpAAQAAAAB
AAAJAuocAAcAAAgMAAAAhgAAEYYc6gAAAAgAAAAAAAAAAAAAAAAAAAA … AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

**Figure 64.** Set request format as JSON in Postman. (Postman screen capture)

*5. Handling Base64 image Format:*



```
{
    "TimeSend": "2023-03-15T15:34:49.182",
    "Item": {
        "ImageData": "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEASABIAAD/2wBDAAUDBAQEAwUEBAQFBQUGBwwIBwcHBw8LCwkMEQ8SEhEPERETFhwXExQaFRERGCEYGh0dHx8fExciJCIeJBweHx7/
```

2wBDAQUFBQcGBw4ICA4eFBEUHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh7/wAARCALQBQADASIAAhEBAxEB/8QAHQAAAQUBAQEBAAAAAAAAAAAAAgADBAUGBwgBCf/
EAGEQAAEDAgMDBwYGDQoBCwMCBwIAAwQFEgYTIgEyQgcRFCNSYnIVM4KSorIkNENzwtIIFiExUVNhY3F0g7PiJSY1NkFEZJOj8FQXRVV1gYSRlKHD8mWx4yc3wdMYpGaV8f/
EABsBAQADAQEBAQAAAAAAAAAAAAACAwQBBQYH/8QAMxEBAAEDAwQCAQQCAQMEAwAAAAIBAxIEETITISIxBUJBBBhQzUVJhIzRicRVDcqEHFoH/2gAMAwEAAhEDEQA/AB5MMHyDbZr2IGREvORYjg/
6jn0RXRj3riWf5O8XR8VU8swhbqkYbZTPa/OD3VpC2KCm5JkeUHDPl2H0yG2PlSMPV8OeP4svorhbtYJvq5EdwXGyISEi1CXZXozE9bg4epZVKoFp
+RZ4nSXmnF9Qerlaeq0htoXnyzNA2iuRijC532bDBMCpYxqQxY4vMwIjYi9JcLSyO9aPeJbTlBp8OnQ6LBgxxZjtC9aP+XqLtEqvkMxfFyRwfOFuO4JXQ3h05vaEu92SV9ys/GqSPZZe94VZDklcqxS
+xfjDfiFNpyL8Yb8QrUqQ6ZKcmRekObzjjnsuEI+6paqsNaaHD4rswvWcJWqDoPJkI+Ryt713izC+qtYszybDbh0dNpb3rOOLTLLd9tNOJI0CNRSJJJJAl9XxfUCSSSQEkkkgSSSSBK8y/gv/AHdUa0xh8H/
7uoyQlxUdOa65ZXHtGjnWMLvCLuc/WG2SuLTl5glurdUxrWs/jZktuJMEtjvFXB3fEK7T2hGrrTo9c54ky/lttuOOELbbY3ERcKmGPWF4lV4opblXw7OpjLwx3JLYiLhDcI2kJfRVWK6rA4hqszEVWbpdPZLo5eZC3i/
GEtlGp7NLw65BHdaiuXF2itK4lz0uSHEWZnDiqIJcNrbg/SW+wrQ5lLwj5HmShkSsl5snxut1Xalnw9reNGBpnwfBuKnhuetbbu8P/wAVFwZhWDiKhysQSJQsk0Tgtti0lu3VqW
+wlhd6nRapEqxRpTNR0kLd27baQrKtcnNepDz0ej1KNKguuEQi+4TZekI7yjW34rIzW3JJKck02oMl5liQOWPZuFbrmH8KqcGYdbw9Rxh5mc84WY852i
V5b+RXRj4qbntwP7IxnPx1h9ntDH/
fEphjdILxJn7IAruUzDbIl/wt3
+YSmc3O96Ssl6iqtrSnMXNirLo2lN0xvqxVlZpRGUkeNGTzscbh0ipUYNKFcPF6O2wTfRXCbIc5shLV3t1SqVDzrmRIWydLLFwt0buJW1TZ6tx75RzeJRaY38Fc8SrRpKuyBGoj9Kiy4smaE49
r12xwB2jt2CXDbtU2nxhGQKeabLoZf0JZhRm5EgbbgZIhuLTdbpXC3vGnkwr79QbxE5T7WfJpU0csbhuJwbfrEtJXmbZRDbxKPgxt6dgmLMqAtuVB1xwXHhEbrcwrRuVhXBumEuRSjJxX7JAbWaS32Ybhe0umUxi2DDE
uG037ornP2SoiMqljq/o8v3hLqUHZaLI9lsR9lXSj4USieJjq0zkalZOja2mWhEiVZkTDCWR1+6pjTepOZdriYo5KeS1qUinRm3Cy7SJwtQp6S3cSpzqk6FWJUpttkYNOilnE
+JDqIbW9XeL2RWPW3ujDKNUZSWDstmNS3qg05mCTPUssuCLj2q3ORHi8K5/OquHaxmSJTOTFiFmFnueft1Zfpbt3eT2HsWZ

**Figure 65.** Delete the base64 data tag before comma `,`. (Postman screen capture)

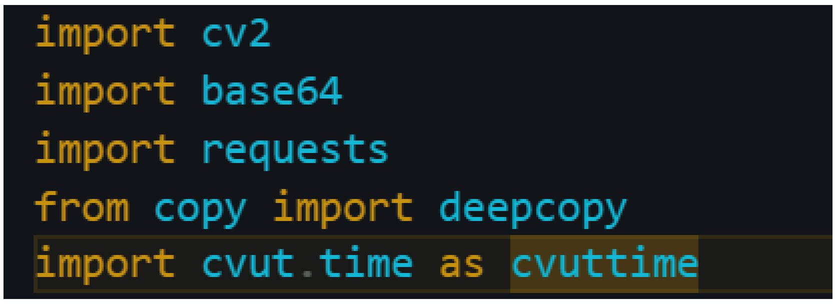
For a detailed report on API tests with Postman within the working scope, this thesis offer an AI supports improving recognition features report (Nguyen 2023a) that readers can read and refer for similar projects.

Overall, manual testing with Postman lays the foundation for understanding the testing process, setting the stage for implementing algorithms for Unit Tests. The subsequent subchapter will explore the implementation of unit testing using Pytest based on information gathered from the API description documentation and Postman manual testing.

## Writing Unit Testing with Pytest.

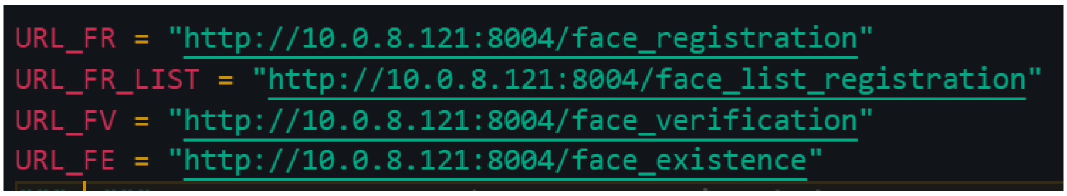
Algorithm Overview and Implementation in Pytest:

Due to confidentiality constraints, this section presents fundamental algorithms for unit testing and scalability testing without revealing ITD Group's private source code.



```python
import cv2
import base64
import requests
from copy import deepcopy
import cvut.time as cvuttime
```

**Figure 66.** Importing necessary libraries for Unit Testing with Pytest. (Source code screen capture)



```python
URL_FR = "http://10.0.8.121:8004/face_registration"
URL_FR_LIST = "http://10.0.8.121:8004/face_list_registration"
URL_FV = "http://10.0.8.121:8004/face_verification"
URL_FE = "http://10.0.8.121:8004/face_existence"
```

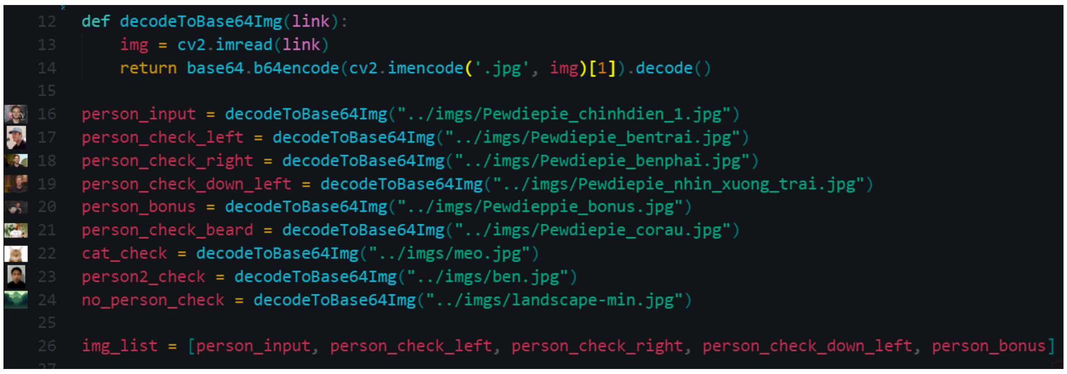
**Figure 67.** Declaring the constants for API URL. (Source code screen capture)

*'URL_FR'*: URL of the face_registration API for registering a face in the AI video database system at ITD company.

*'URL_FR_LIST'*: URL of the face_list_registration API for registering a face list in the AI video database system at ITD company.

*'URL_FV'*: URL of the face_verification API for verifying a face in the AI video database system at ITD company.

*'URL_FE'*: URL of the face_existence API for Checking if a face is existed in the AI video database system at ITD company.

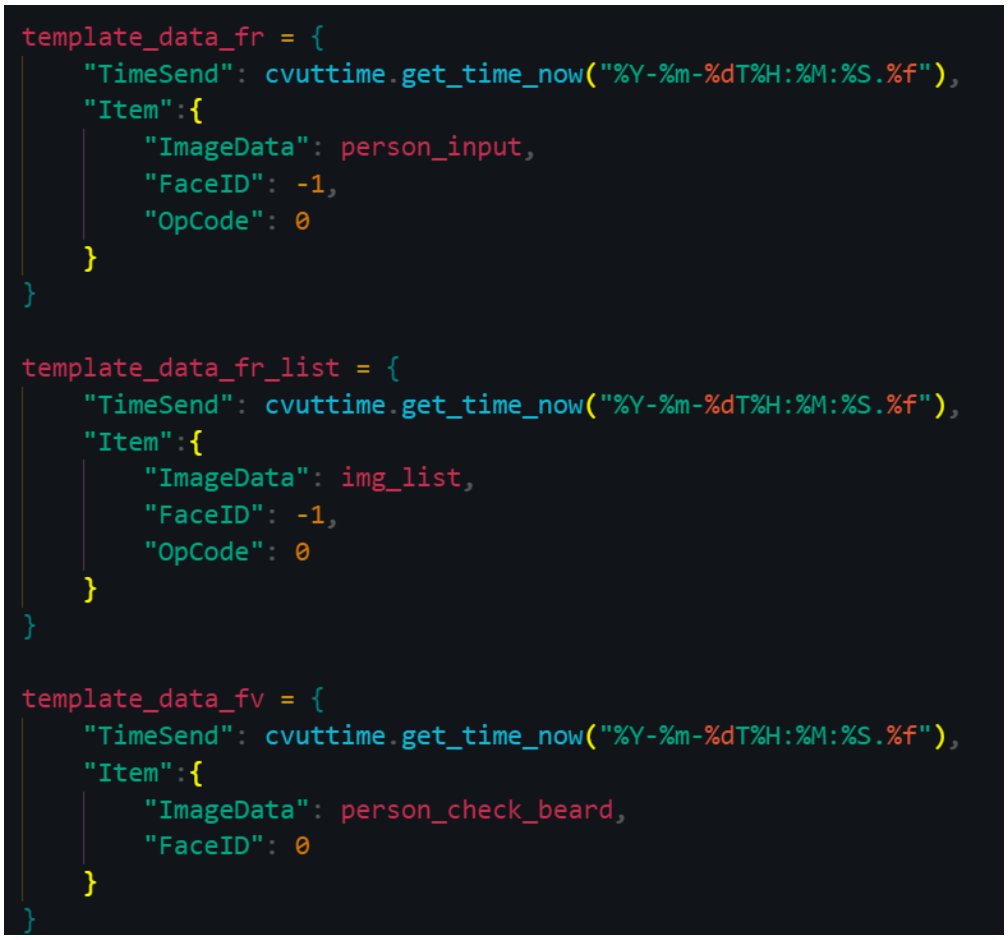


```python
def decodeToBase64Img(link):
    img = cv2.imread(link)
    return base64.b64encode(cv2.imencode('.jpg', img)[1]).decode()

person_input = decodeToBase64Img("../imgs/Pewdiepie_chinhdien_1.jpg")
person_check_left = decodeToBase64Img("../imgs/Pewdiepie_bentrai.jpg")
person_check_right = decodeToBase64Img("../imgs/Pewdiepie_benphai.jpg")
person_check_down_left = decodeToBase64Img("../imgs/Pewdiepie_nhin_xuong_trai.jpg")
person_bonus = decodeToBase64Img("../imgs/Pewdieppie_bonus.jpg")
person_check_beard = decodeToBase64Img("../imgs/Pewdiepie_corau.jpg")
cat_check = decodeToBase64Img("../imgs/meo.jpg")
person2_check = decodeToBase64Img("../imgs/ben.jpg")
no_person_check = decodeToBase64Img("../imgs/landscape-min.jpg")

img_list = [person_input, person_check_left, person_check_right, person_check_down_left, person_bonus]
```

**Figure 68.** Define a function ´decodeToBase64Img´ for decoding images into Base64. Simultaneously, declare base64 image variables. (Source code screen capture)

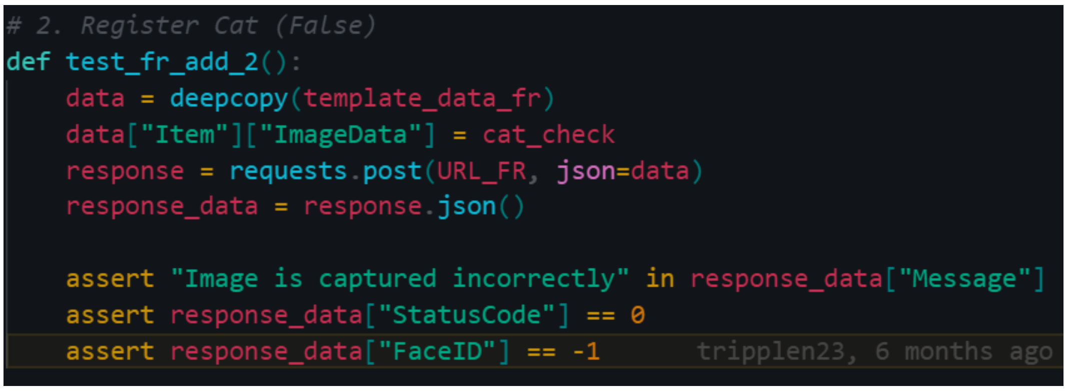
*‘decodeToBase64Img(link)’*: In this function, we got a parameter *‘link’* and decode the path of the image in .jpg format into base64 using `*b64encode*` method.

For enhancing code structure to be reusable, declare template variables containing the model request data that will be sent to the system. Consequently, whenever a message is requested, inherit these templates with the desired data (e.g., Person image)



```python
template_data_fr = {
    "TimeSend": cvuttime.get_time_now("%Y-%m-%dT%H:%M:%S.%f"),
    "Item":{
        "ImageData": person_input,
        "FaceID": -1,
        "OpCode": 0
    }
}

template_data_fr_list = {
    "TimeSend": cvuttime.get_time_now("%Y-%m-%dT%H:%M:%S.%f"),
    "Item":{
        "ImageData": img_list,
        "FaceID": -1,
        "OpCode": 0
    }
}

template_data_fv = {
    "TimeSend": cvuttime.get_time_now("%Y-%m-%dT%H:%M:%S.%f"),
    "Item":{
        "ImageData": person_check_beard,
        "FaceID": 0
    }
}
```

**Figure 69.** Declare template data variables. (Source code screen capture)



```python
# 2. Register Cat (False)
def test_fr_add_2():
    data = deepcopy(template_data_fr)
    data["Item"]["ImageData"] = cat_check
    response = requests.post(URL_FR, json=data)
    response_data = response.json()

    assert "Image is captured incorrectly" in response_data["Message"]
    assert response_data["StatusCode"] == 0
    assert response_data["FaceID"] == -1        tripplen23, 6 months ago
```

**Figure 70.** Test case registering a cat into the face recognition system. (Source code screen capture)

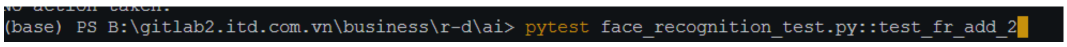
Use the `*deepcopy*` module from imported `*copy*` library to copy the template that is declared earlier. (`*template_data_fr*` for face registration data).

Modify the image data requested test. Remember to change both the request and the response data into JSON format.

Utilize the `*assert*` module of Pytest to evaluate the test result, determining whether it passed or not.

Execute the test case with the command `*pytest [file_path]::[test_case]* `.

For example: `*pytest face_recognition_test.py::test_fr_add_2*`.



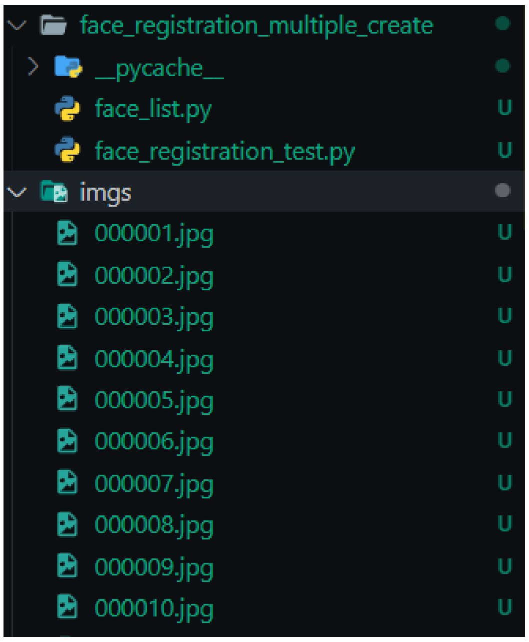
```
(base) PS B:\gitlab2.itd.com.vn\business\r-d\ai> pytest face_recognition_test.py::test_fr_add_2
```

**Figure 71.** CLI command for test our test case. (Source code screen capture)

This unit testing approach ensures the functionality of various components within the AI system, verifying that each unit performs as expected. The subsequent section will delve into an illustrative example of scalability testing, shedding light on the system's performance under varying workloads. Moreover, writing unit tests help testers or developers understand deeply how everything works behind the scenes of a typical software project.

### Writing scalability test with Pytest

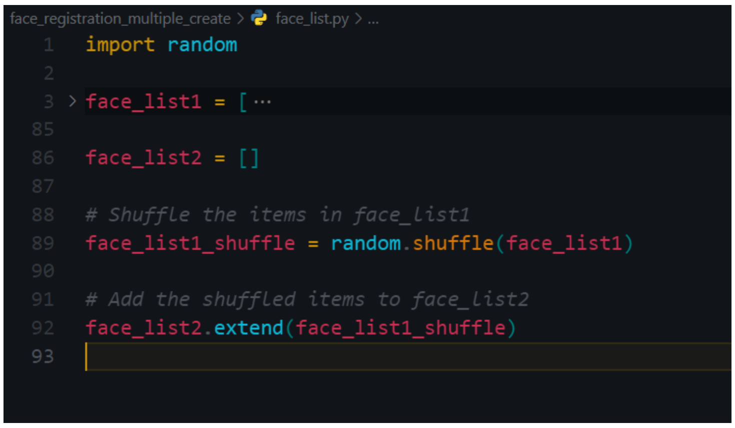
By applying these basic testing algorithms workflow, Unit Testing can be extended to cover various cases. For scalability testing, where manual testing of hundreds or thousands of images become more time consuming, Pytest offers an efficient solution. The algorithm idea of scalability test in API of face registration can be found at the Scalability test with Pytest for ITD group (Nguyen 2023b).



```
face_registration_multiple_create
  __pycache__
  face_list.py                          U
  face_registration_test.py             U
imgs
  000001.jpg                            U
  000002.jpg                            U
  000003.jpg                            U
  000004.jpg                            U
  000005.jpg                            U
  000006.jpg                            U
  000007.jpg                            U
  000008.jpg                            U
  000009.jpg                            U
  000010.jpg                            U
```

**Figure 72.** Folder structure for scalability test (Visual studio code screen capture)

the first folder named *`face_registration_multiple_create`* with 2 files inside as *`face_list.py`* and *`face_registration_test.py`*. The second folder named *`imgs`*, contains the person face images data:

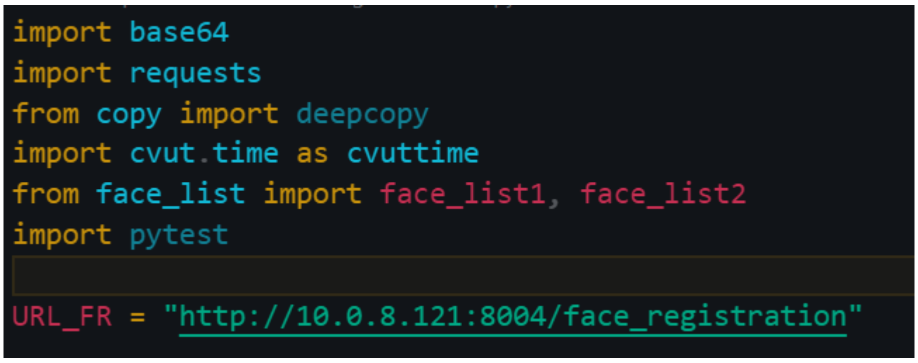


```
face_registration_multiple_create > face_list.py > ...
import random

face_list1 = [...

face_list2 = []

# Shuffle the items in face_list1
face_list1_shuffle = random.shuffle(face_list1)

# Add the shuffled items to face_list2
face_list2.extend(face_list1_shuffle)
```

**Figure 73.** Declare different face list for scalability test in file ‘face_list.py’. (Source code screen capture)

In the *´face_list.py´* file, two different face lists are declared. *´face_list2´* is a randomly shuffled version of *´face_list1´* to save time and resources while ensuring the test process is implemented as expected.

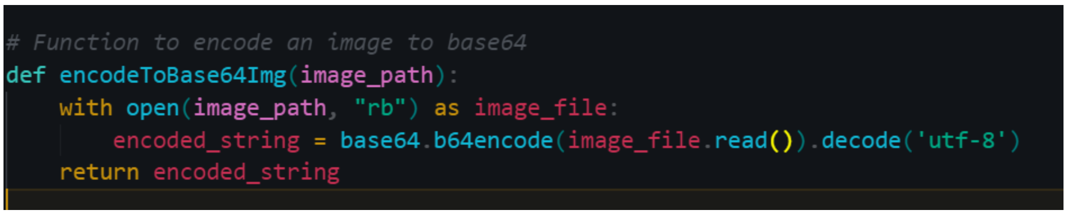
Let’s implement the Scalability Testing Workflow in ´face_registration_test.py´. Firstly, Import Libraries included base64, requests, cvut.time, face_list, and pytest, then define *‘URL_FR’* variable for fetching the Face Registration API URL of ITD:



```
import base64
import requests
from copy import deepcopy
import cvut.time as cvuttime
from face_list import face_list1, face_list2
import pytest

URL_FR = "http://10.0.8.121:8004/face_registration"
```

**Figure 74.** Import libraries and define API endpoint. (Source code screen capture)

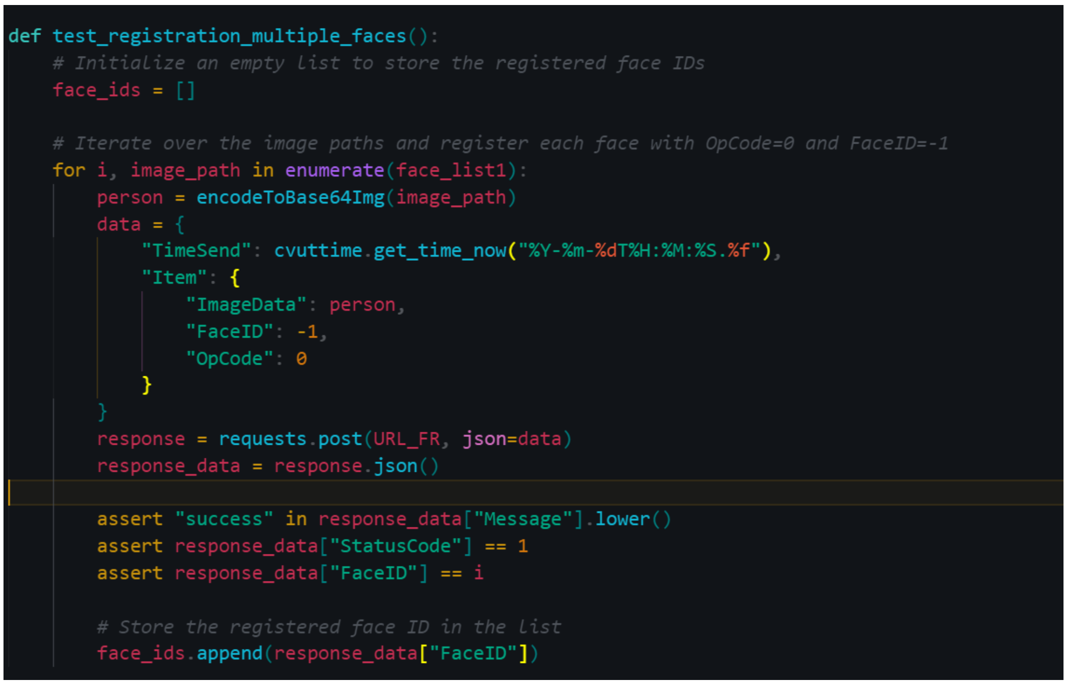
Next, define function that takes an image file path as input, reads the image, encodes it to base64 format, and returns the encoded string. It prepares images for registration in the tests.



```
# Function to encode an image to base64
def encodeToBase64Img(image_path):
    with open(image_path, "rb") as image_file:
        encoded_string = base64.b64encode(image_file.read()).decode('utf-8')
    return encoded_string
```

**Figure 75**. Defining ‘encodeToBase64Img’ function for scalability testing. (Source code screen capture)

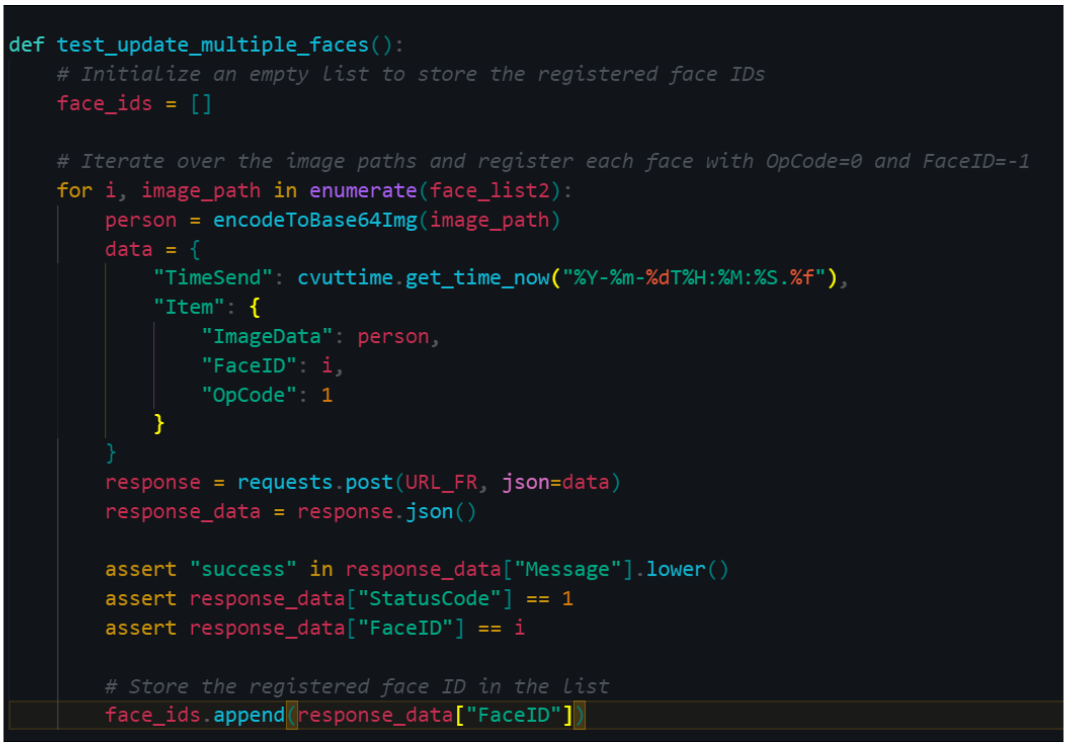
Later, define `test_registration_multiple_faces` function to initialize an empty list *`face_ids`* to store the registered face IDs. It then encodes each image into base64 format, constructs a JSON payload with the image data, FaceID, and OpCode. After that, it sends a POST request to the registration API. The function asserts that the response indicates success, has a StatusCode of 1, and that the returned FaceID matches the expected value (based on the iteration).



```python
def test_registration_multiple_faces():
    # Initialize an empty list to store the registered face IDs
    face_ids = []

    # Iterate over the image paths and register each face with OpCode=0 and FaceID=-1
    for i, image_path in enumerate(face_list1):
        person = encodeToBase64Img(image_path)
        data = {
            "TimeSend": cvuttime.get_time_now("%Y-%m-%dT%H:%M:%S.%f"),
            "Item": {
                "ImageData": person,
                "FaceID": -1,
                "OpCode": 0
            }
        }
        response = requests.post(URL_FR, json=data)
        response_data = response.json()

        assert "success" in response_data["Message"].lower()
        assert response_data["StatusCode"] == 1
        assert response_data["FaceID"] == i

        # Store the registered face ID in the list
        face_ids.append(response_data["FaceID"])
```

**Figure 76.** Defining `test_registration_multiple_faces` function for scalability testing. (Source code screen capture)

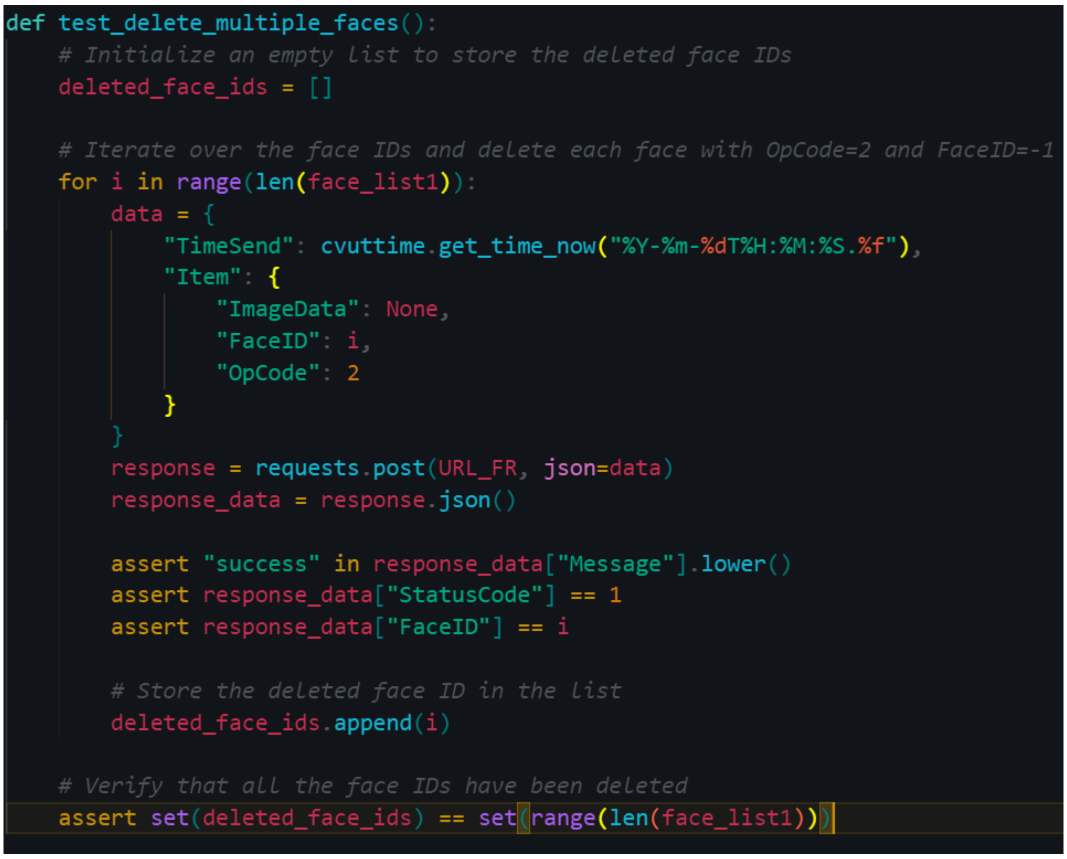
After that, defining `test_update_multiple_faces`, this function is similar to the registration test but focuses on updating existing faces. It iterates over a list of image paths (´*face_list2*´) and updates each face with an OpCode of 1. The function then verifies the response.



```python
def test_update_multiple_faces():
    # Initialize an empty list to store the registered face IDs
    face_ids = []

    # Iterate over the image paths and register each face with OpCode=0 and FaceID=-1
    for i, image_path in enumerate(face_list2):
        person = encodeToBase64Img(image_path)
        data = {
            "TimeSend": cvuttime.get_time_now("%Y-%m-%dT%H:%M:%S.%f"),
            "Item": {
                "ImageData": person,
                "FaceID": i,
                "OpCode": 1
            }
        }
        response = requests.post(URL_FR, json=data)
        response_data = response.json()

        assert "success" in response_data["Message"].lower()
        assert response_data["StatusCode"] == 1
        assert response_data["FaceID"] == i

        # Store the registered face ID in the list
        face_ids.append(response_data["FaceID"])
```

**Figure 77.** Defining `test_update_multiple_faces` function for scalability testing. (Source code screen capture)

Then, defining `test_delete_multiple_faces` function, the function Iterates over the FaceIDs of previously registered faces and sends a request to delete each face with an OpCode of 2. It then verifies that the deletions were successful and ensures the company's database is returned empty after the test to avoid resource overflow.



```python
def test_delete_multiple_faces():
    # Initialize an empty list to store the deleted face IDs
    deleted_face_ids = []

    # Iterate over the face IDs and delete each face with OpCode=2 and FaceID=-1
    for i in range(len(face_list1)):
        data = {
            "TimeSend": cvuttime.get_time_now("%Y-%m-%dT%H:%M:%S.%f"),
            "Item": {
                "ImageData": None,
                "FaceID": i,
                "OpCode": 2
            }
        }
        response = requests.post(URL_FR, json=data)
        response_data = response.json()

        assert "success" in response_data["Message"].lower()
        assert response_data["StatusCode"] == 1
        assert response_data["FaceID"] == i

        # Store the deleted face ID in the list
        deleted_face_ids.append(i)

    # Verify that all the face IDs have been deleted
    assert set(deleted_face_ids) == set(range(len(face_list1)))
```

**Figure 78.** Defining `test_delete_multiple_faces` function for scalability testing. (Source code screen capture)

The purpose of scalability testing is to evaluate how a system performs as it scales to handle an increasing load or number of users, ensuring the system remains responsive and stable as demand is increasing.

## 4.2 Writing script test for object tracing feature

According to Akash7 "Test Scripts are line-by-line description that contains information about system functions that must be performed to verify an application or system under test." (Akash7 2023)

**Writing ITD Script test for tracing feature**

In the context of security-focused AI applications, particularly at ITD, the scripting test for the AI API's tracing feature is crucial. This script aligns with ITD Software department's requirements and serves as an educational example for every AI Testing Engineers.

Objective: Verify the functionality of the AI API's tracing feature for the camera system at ITD building.

Prerequisites:

1. Ensure proper installation and configuration of the smart camera system.
2. Confirm installation and configuration of the AI API for the smart camera system.
3. Enable the AI API tracing feature.

Testing step:

1. Start the smart camera system and make sure it works normally.
2. Access the AI API interface and navigate to the tracing feature.
3. Configure trace settings, specifying duration and objects to be traced.
4. Start the tracking and verification feature of the specified object by clicking on the specific ID or image displayed on the tracing interface (further, we can click on the object that needs to be traced in streaming cam).
5. Interrupt the tracing feature and confirm that it stops capturing objects.
6. Save the traced objects to a file and verify that the file is saved correctly.
7. Open the traced objects file and confirm the traced objects are represented correctly.
8. Repeat steps 3-7 with different tracking settings and objects to be traced.
9. Disable tracing and verify that it is no longer capturing objects.

Expected result:

1. *AI API Tracking Feature:*

   Accurately capture the specified objects. Once the object is clicked, the entire camera system will identify the object in real time.

   In streaming camera: Switches camera views continuously, displaying the moving location of the identified object within the building.

In camera recording: Allow access to image and video data files related to the identified object, considering similarities in appearance over time.

2. *Tracking Interruption:*

   Tracking stops capturing objects when interrupted.

3. *Traced Object File:*

   Property saves and accurately represents the traced objects.

4. *Tracing Disablement:*

   Disabling tracing halts object capturing.

Investigation and Fix: If any of the above steps or expected results fail, the tracking feature of the AI API for smart camera systems will be further investigated and fixed.

**Features that can be integrated into the tracking/tracing object feature**

*Real-time alerts:* Configurable alerts to security personnel or building managers for detected traced object.

*Object recognition:* Tracking of various objects, such as vehicles, packages, or devices.

*Analyze historical data:* Identification of patterns or anomalies in movements for enhanced security.

*Integration with other systems:* Integration with access control or building automation systems for a comprehensive view of building operations and security.

**Purpose of writing script test**

Script tests for object tracing in an AI Video Management System (VMS) offer multifaceted benefits include.

Quality Assurance:  Ensures quality and reliability of the object tracing feature.

Early Issue Detection: Detects and addresses tracking issues early in development.

Reliability: Ensures consistent and reliable object tracing across different scenarios.

Documentation: Serve as documentation for expected tracing behavior, making it easier for AI and IT teams to understand requirements.

Regression Testing: Detects regressions caused by code changes, ensuring tracing remains functional.

Scalability: Scales with the VMS to handle more cameras and scenarios, ensuring tracing accuracy at scale.

Continuous Integration: Automates testing in CI/CD pipelines[3], validating tracing functionality with each code change.

Cost Saving: Identifies and resolves issues early, reducing the need for extensive manual testing and troubleshooting.

Confidence: Builds confidence in the reliability and correctness of object tracing, enhancing user satisfaction.

Compliance: Demonstrates compliance with industry standards and certification requirements.

Advanced Features: Allows integration of advanced features for a more comprehensive security solution.

[3] CI/CD stands for Continuous Integration and Continuous Delivery (or Continuous Deployment). It is a software development method that involves automating testing and pushing code changes into production in a streamlined and automated manner.

In summary, script tests for object tracing improve the AI Video Management System's quality, reliability, and efficiency, ensuring it satisfies user expectations and industry requirements while enabling additional security features.

## 4.3 Summary

In conclusion, Chapter 4 of this thesis stands as a guiding light for the next generation of AI testing engineers, poised to navigate the intricate landscape of AI and APIs. Drawing inspiration from the real-world scenario at ITD Group, this chapter imparts invaluable wisdom and strategies that will empower future engineers to excel in their roles.

At its core, this chapter underscores the paramount importance of holistic understanding. AI testing engineers must transcend traditional roles, immersing themselves in AI workflows, evolving into champions of software quality and user satisfaction.

Independent thinking is emphasized. AI testing engineers must stand with the end-users, fearlessly pointing out defects, ensuring AI systems meet the highest standards - an essential trait of a great AI tester.

Moreover, practical testing comes alive with insights into manual testing via Postman and automated unit testing using Pytest. These are not just tools; they are powerful instruments. Additionally, understanding API documentation becomes a superpower, enabling meticulous AI system validation and effective issue reporting.

Efficiency thrives here. Valuable tips optimize manual testing with Postman, fostering a clean workspace, organized collections, and smart URL environments. These tips save time and boost efficiency.

Furthermore, unit testing with Pytest is demystified step by step, paving the way for automating testing, ensuring robust software quality—a skill essential for any AI tester.

Finally, the piece of resistance is the exploration of script tests for object tracing in AI Video Management Systems. These tests offer a treasure trove of benefits, from enhancing quality to ensuring scalability and compliance. They empower AI testing engineers to instill confidence in AI systems, making them impeccable, not just functional.

In essence, Chapter 4 is a rite of passage for future AI testing engineers, equipping them with knowledge, mindset, and tools. It prepares them to usher AI systems into a future where excellence is a necessity. It ignites a passion for AI testing that will drive innovation, elevating AI engineering standards for generations to come.

# 5 CONCLUSIONS

Generally, this thesis delved into the complexities of AI testing and its important role in ensuring the reliability and performance of AI systems. The journey so far explored many topics, from practical testing methods to scalability testing or even handing on a script test.

As the thesis come to the end of its research, it is necessary to highlight some key lessons and extended directions for future work in similar industry.

Firstly, throughout the research, the thesis has brought readers more knowledge about smart camera system software, the needed features in any iVMS and the basic applications of AI or computer vision to make monitoring security to be easier. Besides, there are also detailed reports as well as basic instructions for an engineer to have a clearer view when embarking on a similar project. The thesis has also provided a typical style of writing reports in ITD company.

However, while this thesis delved into software testing concepts and gained insights into major AI-integrated Camera systems, it is apparent that the role of an AI testing engineer is vast and continually evolving. Our understanding and expertise in this field go beyond the scope of a regular thesis. To excel in this role, a software engineer must commit to continuous learning and expanding knowledge every day.

Moreover, due to compliance with ITD's privacy policy, this thesis has constraints, and direct access to the source code of the original project software is unavailable. Access to such resources would greatly enhance testers’ testing capabilities.

Furthermore, the thesis also would like to contribute in several aspects to the industry of iVMS. Obviously, developing an intelligent camera system relies on rich data sources. The rising demand for data-related professions underscores the need for a robust data strategy tailored to business needs. This includes customer camera installations, security management, and observation systems.

After implementing a data strategy, the focus shifts to system automation, reducing the need for extensive human intervention. To avoid legal issues, companies must review customer data protection policies. Ethical considerations, such as privacy invasion through deepfake tools, underscore software engineers' duty to anticipate and prevent adverse consequences.

Prioritizing security in AI camera systems is crucial for business benefits and user privacy. Looking ahead, incorporating decentralization principles is essential for optimal security, protecting individual rights, and ensuring software integrity. Key considerations include data privacy, resilience against unauthorized access, ethical AI practices, interoperability, user-friendly interfaces, decentralized data storage, user education, and regular security testing. These aspects collectively contribute to advancing AI surveillance responsibly and innovatively.

However, this thesis has not delved deeply into data analysis. For a testing engineer, a deeper understanding of the output data from training models is invaluable. Therefore, further exploration in this area would be beneficial and not redundant.

To summaries, this journey has been an investigation of the fundamental principles and mindsets required of an AI testing engineer. The thesis has not only explored the complexities of working flow in this industry, but also provided ideas into building AI camera systems and core software testing abilities.

This work is valuable because it provides readers with a larger perspective on research, learning, and software development, bridging the theoretical and practical divides. Furthermore, the final chapter introduces critical aspects to further expand understanding in the sector, moving technology towards a future with even greater promise for humanity.

While this thesis may be extensive, it comes with its limitations. However, there is a reminder that learning and fostering curiosity are ongoing challenges in every phase of careers in software industry. To excel as a software engineer, it is essential to combine technical talent with ethical integrity or virtue.

With a vision of continued progress and innovation, this thesis aspires to bring more value not only to the iVMS domain but also to all related software development fields.

**Final words**

Finally, the author extends heartfelt gratitude to all the readers for the patience and understanding throughout this thesis journey. Special thanks are reserved for Professor Kenneth Norrgård, whose guidance contributed significantly to refining this work. The author also expresses sincere appreciation to the ITD Software department and Vaasa University of Applied Sciences for providing the conducive environment necessary for the successful completion of this thesis.

## REFERENCES

Akash7, May 24th, 2023. Software Testing Test Script. Accessed 30.06.2023. https://www.geeksforgeeks.org/software-testing-test-script/.

Axis Communications, 2023. Axis Learning Documentation. Accessed 01.07.2023. https://www.axis.com/learning.

Axis Communications, 2024. Secure insights. Accessed 01.07.2023. https://www.axis.com/blog/secure-insights/audio-analytics-ip-cameras/.

Axis Developer Community, 2024, Access control integration. Accessed 25.02.2024. https://www.axis.com/developer-community/access-control-integration.

Axis Products, 2024a. Axis Video Motion Detection. Accessed 25.02.2024. https://www.axis.com/products/axis-video-motion-detection.

Axis Products, 2024b. Axis Object Analysis. Accessed 25.02.2024. https://www.axis.com/products/axis-object-analytics.

Axis Products, 2024c. Analytics. Accessed 25.02.2024. https://www.axis.com/products/analytics.

Axis Public, 2023. Third-party devices user manual. Accessed 25.02.2024. https://www.axis.com/dam/public/10/f5/23/axis-camera-station-and-third-party-device-support-en-US-388160.pdf.

Axis Solutions, 2024a, License Plate Recognition. Accessed 25.02.2024. https://www.axis.com/solutions/license-plate-recognition.

Axis Solutions, 2024b, Privacy in surveillance. Accessed 25.02.2024. https://www.axis.com/solutions/privacy-in-surveillance.

Axis Support, 2024a, Axis Camera Station Mobile App. Accessed 25.02.2024. https://help.axis.com/en-us/axis-camera-station-mobile-app.

Axis Support, 2024b, Device-to-cloud integration for Azure Cognitive Service for Vision. Accessed 25.02.2024. https://help.axis.com/en-us/device-to-cloud-integration-for-azure-cognitive-service-for-vision.

Axis Vapix Library, 2024, Vapix Library. Accessed 25.02.2024. https://www.axis.com/vapix-library/.

Base64 image, 2024. Base64 encoder web application. Accessed 25.02.2024. https://www.base64-image.de/.

Barton, D., Court, D., October 2012. Making Advanced Analytics Work for You. Accessed 20.10.2023. https://hbr.org/2012/10/making-advanced-analytics-work-for-you.

Cvat, 2024. Cvat platform for labelling image dataset. Accessed 25.02.2024. https://app.cvat.ai/.

Fowler, M., March 25th, 2014. Microservices. Accessed 04.12.2023. https://martinfowler.com/articles/microservices.html.

ITD, 2023a. Image for manual testing step 1. Accessed 25.02.2024. https://drive.google.com/file/d/10Qv3TGcrAxSs-QTzXAASYQZSbr3j9z6V5/view?usp=drive_link.

ITD, 2023b. Image of response data in manual test step 1. Accessed 25.02.2024. https://www.notion.so/image/https%3A%2F%2Fs3-us-west-2.amazonaws.com%2Fsecure.notion-static.com%2F50754711-74d5-4b4f-8308-bb35bfc5619d%2F2.png?table=block&id=19e679fb-0368-4a7d-81ff-db21145f190d&spaceId=2ee80eb6-7237-44d9-b9ed-be4959866b3b&width=2000&userId=8111ef9f-03d9-46db-a6bb-c48ce8127521&cache=v2.

ITD, 2023c. Images for manual testing step 2. Accessed 25.02.2024. https://drive.google.com/drive/folders/1kbzqbSqdqTj7_WXi_xtavGRMU20VOaTy?usp=sharing.

ITD, 2023d, Image 1 of response data in manual test step 2. Accessed 25.02.2024. https://www.notion.so/AI-h-tr-n-ng-cao-t-nh-n-ng-nh-n-di-n-227d0d99077143a79ef59b59748e5308?pvs=4#60314d44dac24f8eb7b34272b4277960.

ITD, 2023e. Image 2 of response data in manual test step 2. Accessed 25.02.2024. https://www.notion.so/AI-h-tr-n-ng-cao-t-nh-n-ng-nh-n-di-n-227d0d99077143a79ef59b59748e5308?pvs=4#9fa40cbc21f74aca8bf97ea0b1119a5f.

Keineahnung2345, 2023. Darknet pre-trained YOLOv4, v3 and v2 model Open Source. Accessed 21.08.2023. https://github.com/AlexeyAB/darknet.

Milestone Learning, 2023a. Training for Milestone products. Accessed 15.05.2023. https://learn.milestonesys.com/index.htm.

Milestone Learning, 2023b. User manual. Accessed 15.05.2023. https://doc.milestonesys.com/sc/pdf/2023r3/en-US/MilestoneXProtectSmartClient_UserManual_en-US.pdf.

Milestone System, 2024a. Product information. Accessed 25.02.2024. https://www.milestonesys.com/products/software/xprotect/.

Milestone System, 2024b. XProctect Clients – Stay in control wherever you are. Accessed 25.02.2024. https://www.milestonesys.com/products/software/video-management-software/XProtect-Clients/.

Milestone System, 2023a. Configure user profile on server (administrator). Accessed 25.02.2024. https://doc.milestonesys.com/latest/en-US/add-ons/addon_access/mobc_configureuserprofil.htm.

Milestone System, 2023b. Ai Smart City Counter. Accessed 25.02.2024. https://www.milestonesys.com/technology-partner-finder/imotion-analytics/ai-smart-city-counter/.

Milestone System, 2022a. AI-based Abnormal Sound and Voice Analytics System. Accessed 25.02.2024. https://www.milestonesys.com/it/technology-partner-finder/ivs-technology-co.-ltd/ai-based-abnormal-sound-and-voice-analytics-system/.

Milestone System, 2022b. Vezha Heatmap. Accessed 25.02.2024. https://www.milestonesys.com/technology-partner-finder/incoresoft-llc/vezha-heatmap/.

Milestone System, 2022c. AI-Smart Surveillance. Accessed 25.02.2024. https://www.milestonesys.com/fr/technology-partner-finder/a.i.tech-srl/ai-smart-surveillance/.

Milestone System, 2022d. AI-Smart Retail. Accessed 25.02.2024. https://www.milestonesys.com/technology-partner-finder/a.i.tech-srl/ai-smart-retail/.

Milestone System, 2022e. VTrack – Video Analysis for intelligent VS. Accessed 25.02.2024. https://www.milestonesys.com/technology-partner-finder/technoaware-srl/vtrack---video-analysis-for-intelligent-vs/.

Naidu, S., February 21st, 2023. Chaos Testing – A Guide to Chaos Engineering. Accessed 25.11.2023. https://www.qatouch.com/blog/chaos-testing/.

Ng, A., 2023. Building an AI project, AI and society. Accessed 01.08.2023. https://www.coursera.org/learn/ai-for-everyone.

Nguyen, B., May 20th, 2023a. Sample AI API testing report for ITD group. Accessed 15.09.2023. https://www.notion.so/AI-supports-improving-recognition-features-227d0d99077143a79ef59b59748e5308?pvs=4.

Nguyen, B., May 21st, 2023b. Scalability test with Pytest for ITD group. Accessed 07.09.2023. https://github.com/tripplen23/thesis/tree/main/scalability_test_example.

Nguyen, B., August 2023c. Instruction for using Telegram Bot. Accessed 03.09.2023. https://github.com/tripplen23/thesis/blob/main/Intrusion%20warning/telegram_bot_instruction.md.

Nguyen, B., Le, K., July 2023d. ITD AI API Camera System Documentation English Version. Accessed 01.08.2023. https://livepuv-my.sharepoint.com/:w:/g/personal/e2001352_edu_vamk_fi/Eax4uC-bhZZLrEjvmCRANUcB0Lbi3oeN-Sdn7jfqfClUnQ.

Nguyen, B., August 2023e. Intrusion Warning project. Accessed 15.09.2023. https://github.com/tripplen23/thesis/tree/main/Intrusion%20warning.

Nguyen, B., August 2023f. Pre-trained YOLOv3 model for human detection. Accessed 22.09.2023. https://drive.google.com/drive/folders/19Dx4WInLRKh5TnUkdTucFQ8gi0yB4D4a?usp=sharing.

Nguyen, B., 2023g. Training YOLO v8 model with Google Collaboratory. Accessed 22.09.2023. https://drive.google.com/file/d/1RUBh7fbZ0yPJ-JiMibDm7cvVnwKquvnhq/view?usp=sharing.

Nguyen, B., 2023h. Folder structure for training AI model. Accessed 22.09.2023. https://drive.google.com/drive/folders/1aUHAatJnt-Giw8ntURGBxJGkLB8wkV2ut?usp=sharing.

Nguyen, B., 2023i. Computer vision Google Drive folder. Accessed 22.09.2023. https://drive.google.com/drive/folders/1XakorSccF6LC8ve8AG-psKdDVwQCe_X0A?usp=sharing.

Ngo, H., 2022, Intrusion Warning Project. Accessed 03.07.2023. https://github.com/ngominhhaibk/Intrusion_Warning-.

Nist, 2023, Comprehensive testing. Accessed 30.11.2023. https://csrc.nist.gov/glossary/term/comprehensive_testing.

Refaces, 2020. What is facial recognition used for. Accessed 04.12.2023. https://recfaces.com/articles/what-is-facial-recognition-used-for.

Samsara, July 29th, 2021. What is an Intelligent Video Management System? Accessed 01.09.2023. https://www.samsara.com/guides/ivms/.

Schillerstrom, M., October 11th, 2022. The top chaos engineering tools. Accessed 05.09.2023. https://www.harness.io/blog/chaos-engineering-tools.