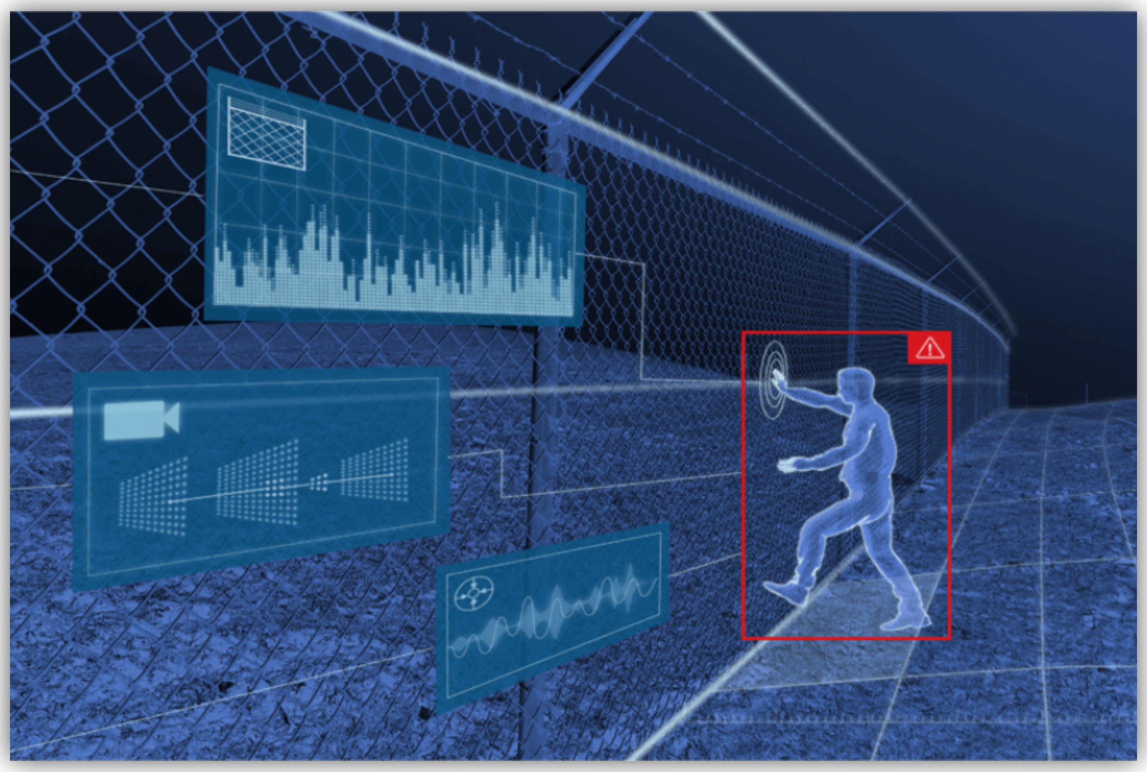
Software Engineering
1st Semester
2022/2023

Project Proposal Document

Human Detection in CCTV Systems



Under the supervision of:
Nuno Sá Couto
Rafael Direito

Table of Contents

1 Document History

Revision	Date	Updated By	Update Description
v0	27-09-2022	Rafael Direito	Initial Version
V0.1	26-10-2022	Rafael Direito	Added several **important** questions/answers to the FAQ Section. A new section "Useful Tutorials/Links" was also added to the document.

2 Motivation

This project will tackle the development of a software solution for a security company named SecCom. SecCom is a company that ensures critical buildings are not broken into, through the installation and operation of CCTV cameras on-premises.

Although, SecCom is still not foresting the technological advancements in the security monitoring field, having several people monitoring the cameras of the most critical buildings.

The goal of your team is to help SecCom with their transition to the digital world, creating an automatic system that can identify intruders without human-intervention and act accordingly.
With their digital transition, SecCom will also install several light and sound alarms on the spaces they are protecting and expect them to be automatically activated every time an intruder is detected. Besides this, SecCom also expects that your team develops a top-to-bottom solution they can use manage the cameras and alarms installed on-premises and to manage all their clients and buildings monitored.

3 Base Use Cases and Considerations

3.1 Mandatory

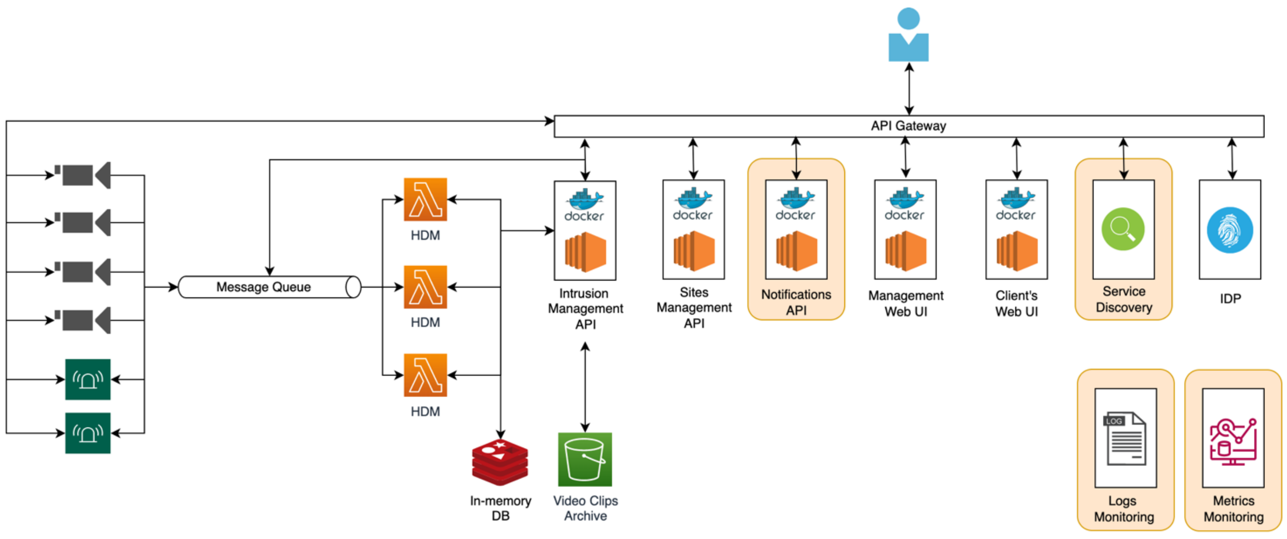
- The CCTV cameras will be continuously recording video and sending frames to the “Human Detection Module” in order to evaluate if there is a human on-premises or not;
- If the “Human Detection Module” detects a human in 3 consecutive frames, we assume there is an intruder on-site;
- In case an intruder is detected, your software solution should persist the camera recording. To do so, your solution will have to query the camera and request a portion of the recording, starting 3 minutes before the intrusion and ending 3 minutes after the intrusion is detected. This video clip should be stored in a file archival solution;
- If an intrusion is detected, the light and sound alarms should be automatically triggered;
- Your solution should provide a portal to manage all buildings being monitored, register new cameras and alarms and manage them;
- Your solution should also provide a client-targeted portal to obtain data from the cameras and alarms, manage how the client wishes to receive intrusion notifications, and check all events triggered in the client’s building;
- The access to these portals should be mediated through an Identity Provider mechanism, for user authentication and authorization. Please adopt RBAC and a widely known protocol for authentication and authorization;
- The previous web-portals and the IDP should be accessible through an API Gateway;

3.2 Extra

- In case of an intrusion, the system you will develop should notify the building administrators and the police;

- Your solution should also monitor the health of the cameras and alarms installed on-premises and, in case any of them is not working properly, it should raise an alarm addressing this situation;
- All site's cameras and alarms should be registered through a Service Discovery mechanism;
- Your system should store all notifications sent and alarms raised;
- You should also monitor all system's components (metrics + logs).

4 Architecture



4.1 Components and Micro-Services

Component	Description
Human Detection Module (HDM)	This module analyzes the frames sent by the security cameras, to detect if there is a human on-site or not
In-memory Database	This database should be used by the HDM to evaluate if there are N consecutive frames where a human was detected
Video Clips Archive	The Video Clips Archive should store the video recordings of the intrusions. The Intrusion Management API should request the video clips from the cameras and store them in AWS S3
Intrusion Management API	This API will be used to act whenever an intrusion is detected. It will get the intrusion video clips from the cameras, activate the alarms, and trigger a new notification in the Notifications API
Sites Management API	This API should be used to track all properties being monitored, along with all the logic regarding the "owners" of each property. It should make available endpoints for the creation/update/deletion of new properties, creation/update/deletion of new property owners, etc…
Notifications API	This API is responsible for informing the property owners and the police whenever an intrusion is detected
Management Web UI	Via this graphical interface, the platform's admins should be able to see all properties being monitored, the intrusions that took place, a list of each

	property cameras and sensors (also their health), and all data regarding the platform's clients. Basically, this UI is used to manage the entire platform
Client's UI	This UI is solely offered to the owners of the properties. Through it, the property owners should be able to see a listing of all cameras, sensors, intrusion events, etc. Besides this, the property owners should be able to update their information through this UI
Service Discovery	Every time a new camera or sensor is added to a property, the camera/sensor should register itself in the Service Registry, listing how it can be accessed
Identity Provider (IDP)	The IDP will provide authentication and authorization mechanisms for all the aforementioned APIs and UIs
Logs Monitor	All system logs should be centralized in this entity
Metrics Monitor	All system metrics should be centralized in this entity

4.2 Suggested Technologies (but NOT MANDATORY)

Component	**Suggested Technologies**
Human Detection Module (HDM)	**This module will have to be scaled and descaled on demand**. We suggest using serverless computing to implement it, for instance AWS Lambda. Regarding the Human Detection algorithms, students can use several technologies: OpenCV, Pytorch, Tensorflow, Keras, etc.
In-memory Database	Suggested Technologies: REDIS, Amazon ElastiCache, Memcached, etc
Video Clips Archive	Suggested Technologies: AWS S3 (**highly recommended)**, FTP Server, etc
Intrusion Management API	Suggested Technologies: FastAPI, Django, SpringBoot, Ruby on Rails, Express.js, etc
Sites Management API	Suggested Technologies: FastAPI, Django, SpringBoot, Ruby on Rails, Express.js, etc
Notifications API	Suggested Technologies: FastAPI, Django, SpringBoot, Ruby on Rails, Express.js, etc
Management Web UI	Suggested Technologies: React, Angular, Vue.js, Plain HTML + JS + CSS, etc
Client's UI	Suggested Technologies: React, Angular, Vue.js, Plain HTML + JS + CSS, etc
Service Discovery	Suggested Technologies: AWS Cloud Map, Consul, Etcd, Zookeeper, Eureka, DNS Service Discovery, etc
Identity Provider (IDP)	Suggested Technologies: Identity Server, Keycloak, Ory, AWS IAM
Logs Monitor	Suggested Technologies: ELK Stack, Fluentd, Graylog
Metrics Monitor	Suggested Technologies: TICK Stack, Prometheus + Grafana, Zabbix, Nagios, etc

5 General Project Guidelines

In this project, you should:

- To deliver working software applying an **Agile/Scrum** software development methodology
- Apply a **microservice architecture** and best practices
- Use as much as possible the free tier AWS services
- Document all APIs using the **OpenAPI** specification
- Having a **CI/CD pipeline** with Automated Unit Testing, and **integrated with the SCM**, is considered as plus

You must use:

- **Docker**: To package most of the system's components
- **Git**: Source Code Management
- **JIRA**: To manage and communicate project sprints, user stores and other relevant issues
- **Docusauros**: To create a documentation page / wiki. In this page you should list all the deliverables (see next section)

6 Deliverables

1. Application **detailed architecture** diagram (APIs, databases, etc) + **workflows**
2. **User Stories detail** (for the ones implemented) by sprint – **to be presented in weekly sprint Reviews**
3. **Application code** (the course lecturers should have access to your Git repository)
4. **Demo** of the system (video that should be included in the final presentation) - to be discussed later
5. Final presentation

7 Useful Tutorials/Links

- Build & Push Docker Image to AWS ECR using GitHub Actions
- GitHub: Deploying to Amazon Elastic Container Service
- AWS ECS docker logs to CloudWatch
- AWS: Monitoring your container instances
- AWS: Passing environment variables to a container

8 FAQ

[Question]
Which are the camera/alarms-level metrics that the Client UI should make available?

[Answer]
You have total control of the metrics you should provide via the Client UI. Although, there are a few questions that must be answered:

1. Is the camera alive?
2. Is the camera streaming video?
3. How much bandwidth is the camera is using?
4. Are the alarms alive?

[Question]
Which property-related information should my Sites Management API store?

[Answer]
This information depends on what functionalities you will provide to the end-user. As you know, there are several extra-features you can implement. These will heavily influence the information you will have to store regarding the properties.
To start, it is suggested you store, at least, the following property-related data:

- Property Id
- Address
- Information on the Cameras that are associated with the Property
- Information on the Sensors placed at the Property
- Owner's Personal Information
- Etc…

[Question]
Will you provide an extended CCTV recording for us to test our Human Detection Module?

[Answer]
We can provide you more videos to test your solution. Once we have selected some videos, we will post them on e-learning.
Although, you can create this extended recording by yourself. Just concatenate N copies of the dummy video we provided into an extended video. Let's say a 10 min. video.
You can also check the VIRAT Video Dataset. There you'll find a lot of CCTV videos.

[Question]
Let's say the Human Detection Module detects an intruder at the timestamp (IntrusionTS) 2022-10-26 18:22:26. Given the project's proposal document, the Intrusion Management API should obtain a video clip that starts 3 minutes before the intrusion and ends 3 minutes after the intrusion. Although, the camera is still recording, and it can't provide the 3 min video clip after the intrusion. How can I deal with this? Should I create a cronjob to gather the video? Besides, if the video has 10 minutes and an intruder is detected at minute 9, how can I deal with this?

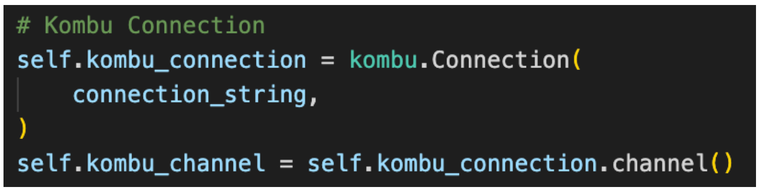
[Answer]
We know the project is complex, so we will make things easier for you…
Assume you always have the entire recording! In fact, since we are emulating the cameras, you already have it locally, on your camera.
Regarding the edge case you mentioned, if you detect an intruder at the 9th minute, your camera should provide a video clip starting at minute 6 (9 – 3) and ending at minute 10 (the end of the recording). The same is valid when you can't obtain the 3 minutes before the intrusion (beginning of the recording).

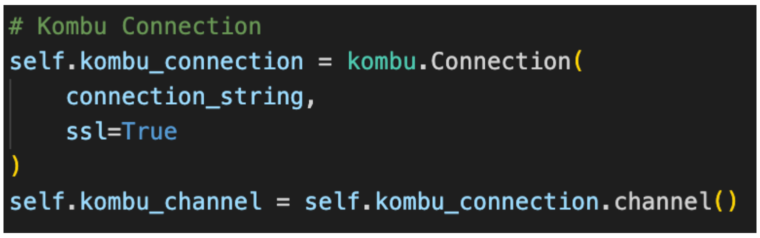
[Question]
What is the best way to gather a video from a camera? Frame-by-frame or getting the entire video at once?

[Answer]
Obtain the entire video at once. To do so, you can make available a REST API in the cameras. It will be through this API that you Intrusion Management API will request the intrusion video clips.

[Question]
I created an AWS MQ using RabbitMQ. When I tried to connect the cameras provided by Professor Rafael to RabbitMQ I faced and error stating that the AWS MQ was not accepting the connection. I've tried everything but nothing worked… Is there any problem at the camera-level that makes it impossible to establish a connection with an AWS MQ?

[Answer]
First of all, the camera code that Professor Rafael made available is just initial code. It doesn't address all the problems you may face during this project. You'll have to adapt it to your needs.
Regarding the problem you experienced, it should be related to SSL. AWS's Message Queues make available an SSL 'protected' endpoint, so you'll have to update the cameras code to establish a connection using SSL. This minor update should solve it…


```
# Kombu Connection
self.kombu_connection = kombu.Connection(
    connection_string,
)
self.kombu_channel = self.kombu_connection.channel()
```



```
# Kombu Connection
self.kombu_connection = kombu.Connection(
    connection_string,
    ssl=True
)
self.kombu_channel = self.kombu_connection.channel()
```

[Question]
What you be the best way to starting migrating the Cameras and the Human Detection Module to AWS? How can I start doing this?

[Answer]

We can give you a collection of steps you should follow regarding this process. We know it may be complex for you to make available your solution in AWS, so you should follow the next steps attentively. Please, don't progress to the next step before validating the results of the current step. Otherwise, you'll make things much more difficult for you.

<u>Step 1</u>
Start by deploying an AWS MQ (RabbitMQ) and having it publicly available.
Then, update the RabbitMQ credentials in your camera and HDM and try to start them locally. If everything is correctly configured, your HDM should receive frames from the camera.

<u>Step 2</u>
Develop a Dockerfile you can use to create the Camera and HDM docker images. The RabbitMQ credentials should be passed as environment variables.
Then you can run your docker images and check if the HDM is receiving frames from the cameras. Use the AmazonMQ broker for the communication between the two different entities.

<u>Step3</u>
Create an Amazon Elastic Container Register (ECR) and onboard your docker images to this registry. Also create an Amazon Elastic Container Service (ECS) Cluster.
Then, develop a Task Definition, which you will use to deploy the docker image you have just uploaded to your ECR. To pass the environment variables to your docker container you may use AWS Secrets Manager or an env file, stored in AWS S3.
Try to run the task you defined, and check if a docker container was started. You may want to configure the Logging process of your docker containers. To do so, you may use the *awslogs* log driver, which will collect all logs from your containers and make them available in AWS Cloud Watch.

<u>Step4</u>
Create an ECS Service to manage your docker containers. You can set the minimum/desired/maximum number of tasks (containers) running. In the future you will also have to configure the scaling policies for your service.

<u>Step5</u>

After you successfully concluded all previous steps, you may start automating everything you did. If you’re using GitHub, you may want to read this. It will be quite helpful.

To help you achieve all this, you may read the tutorials/websites listed in the Useful Tutorials/Links Section.
