**A close up of a logo

Description automatically generated**

**An IoT security project for granting controlled access to infrastructure using multi factor authentication.**

|  |  |  |
| --- | --- | --- |
| **DOCUMENT INFORMATION** | | |
| **Student Name:** |  | **Kieron Garvey** |
| **Student#:** |  | **96358157** |
| **eMail:** |  | **96358157@mail.wit.ie** |
| **Supervisor:** |  | **Frank Walsh** |
| **Module:** |  | **Placement & Project** |
| **Course:** |  | **Higher Diploma in Computer Science** |
| **Version:** |  | **2.0** |
| **Effective Date:** |  | **12th April 2025** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Declaration of Authenticity** | | | |
| I declare that the work which follows is my own, and that any quotations from any sources (e.g., books, journals, the internet) are clearly identified as such by the use of ‘single quotation marks’, for shorter excerpt and identified italics for longer quotations. All quotations and paraphrases are accompanied by (date, author) in the text and a fuller citation is the bibliography. I have not submitted the work represented in this report in any other course of study leading to an academic award. | | | |
| Student: | Kieron Garvey\_\_\_\_\_\_ | Date: | 12/04/2025\_\_\_\_\_\_ |

|  |
| --- |
| **Acknowledgements** |
| Maria, thank you to my wife who looked after everything in the home while I worked on this project.  Frank, to my Project Supervisor (Frank Walsh) for his guidance during the project. It was Frank who suggested to me to refactor the original AMS project.  John Stapleton (my boss), thank you for teaching me about electronics, building and wiring the electrical box with me.  Aaron Whelan, thank you my friend for helping combine everything together on the model. Attaching the electrical box, cutting out the door and attaching the maglock. It wouldn’t look as good without your help  Will Croke, thank you for all your advice on the card reader and the idea of linking in the actual mag lock into a model for the project.  Lecturers, thank you to all the lecturers on the Higher Diploma in Computer Science course. I used a lot of what I learnt throughout the last two years to build this application. |

|  |
| --- |
| **Preface** |
| The report should be read together with the following:  Project Website: <https://securegate-kg.vercel.app/>  On the project website you will find links to the Projects   * GitHub Repo * YouTube Walkthrough Video * Trello Board used for planning the project * Interim and this Final Report |

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| **Revision #** | Description | **Author** | **Effective Date** |
| 1.0 | Initial Draft Submission | Kieron Garvey | 23/01/2025 |
| 2.0 | Conversion of Interim Report into Final Report | Kieron Garvey | 02/04/2025 |

**Table of Contents**

[1 Introduction 7](#_Toc195277629)

[1.1 Objective 7](#_Toc195277630)

[1.2 Problem Statement 7](#_Toc195277631)

[1.3 Solution 8](#_Toc195277632)

[1.4 AMS: What is it? 8](#_Toc195277633)

[1.5 Acronyms and Abbreviations 8](#_Toc195277634)

[2 Research and Analysis 10](#_Toc195277635)

[2.1 Research 10](#_Toc195277636)

[2.2 Existing Solutions 11](#_Toc195277637)

[2.3 Tools 12](#_Toc195277638)

[2.3.1 Proxmox 12](#_Toc195277639)

[2.3.2 Portainer 12](#_Toc195277640)

[2.3.3 Docker 12](#_Toc195277641)

[2.3.4 Kubernetes 13](#_Toc195277642)

[2.3.5 TinyTuya 13](#_Toc195277643)

[2.3.6 Supabase 13](#_Toc195277644)

[2.3.7 MQTT (Message Queuing Telemetry Transport) 13](#_Toc195277645)

[2.3.8 OpenCV 14](#_Toc195277646)

[2.3.9 Face-Recognition 14](#_Toc195277647)

[2.3.10 Fast API 14](#_Toc195277648)

[2.3.11 Deep Face 14](#_Toc195277649)

[2.3.12 MediaPipe 14](#_Toc195277650)

[2.4 Technologies 15](#_Toc195277651)

[2.4.1 Python 15](#_Toc195277652)

[2.4.2 React 15](#_Toc195277653)

[2.4.3 React Native 15](#_Toc195277654)

[2.4.4 Flutter 15](#_Toc195277655)

[2.5 Equipment 16](#_Toc195277656)

[3 Design & Modeling 19](#_Toc195277657)

[3.1 High Level Overview 19](#_Toc195277658)

[3.2 User Persona 20](#_Toc195277659)

[3.2.1 Administrator 20](#_Toc195277660)

[3.2.2 Chemist 21](#_Toc195277661)

[3.2.3 Lab Technician (Senior) 22](#_Toc195277662)

[3.2.4 Lab Technician (Junior) 23](#_Toc195277663)

[3.2.5 Cleaner 24](#_Toc195277664)

[3.3 User Interface Design 25](#_Toc195277665)

[3.3.1 Front-End Application Interface Design 25](#_Toc195277666)

[3.3.2 Back-End Application Admin Dashboard Interface Design 31](#_Toc195277667)

[3.3.3 User Case Diagram 39](#_Toc195277668)

[3.4 Back-End Design 42](#_Toc195277669)

[3.4.1 Back-End API Design 42](#_Toc195277670)

[3.4.2 Database Schema Design 43](#_Toc195277671)

[3.5 Physical Model Design 49](#_Toc195277672)

[3.5.1 Electrical Drawing 49](#_Toc195277673)

[3.5.2 Front View Design 50](#_Toc195277674)

[3.5.3 Side View Design 50](#_Toc195277675)

[4 Project Methodology & Execution 51](#_Toc195277676)

[5 Review, Reflection & Further work 53](#_Toc195277677)

[5.1 Review 53](#_Toc195277678)

[5.2 Reflection 53](#_Toc195277679)

[5.2.1 What I have learned? 53](#_Toc195277680)

[5.2.2 What I would do differently? 53](#_Toc195277681)

[5.2.3 What went wrong? 54](#_Toc195277682)

[5.3 Further Work 54](#_Toc195277683)

[6 Citations/References 55](#_Toc195277684)

**Table of Figures**

[Figure 1 :- Raspberry Pi 4 & 5 16](#_Toc195277593)

[Figure 2 :- Lrtzcbi Touchscreen Portable Monitor 14 Inch 16](#_Toc195277594)

[Figure 3 :- pHAT Stack Kit for Raspberry Pi 16](#_Toc195277595)

[Figure 4 :- Microsoft Lifecam HD-3000 16](#_Toc195277596)

[Figure 5 :- ELEGOO Upgraded Electronics Fun Kit 17](#_Toc195277597)

[Figure 6 :- Mini Portable USB Night Light 17](#_Toc195277598)

[Figure 7 :- Woox R4785 smart plug 17](#_Toc195277599)

[Figure 8 :- Rekavin USB Plug 18](#_Toc195277600)

[Figure 9 :- High Level Raspberry Pi Setup 19](#_Toc195277601)

[Figure 10 :- Logo Screen Design 25](#_Toc195277602)

[Figure 11 :- Card Scanning Design 26](#_Toc195277603)

[Figure 12 :- Enter Pin Component Design 27](#_Toc195277604)

[Figure 13 :- Facial Recognition Component Design 28](#_Toc195277605)

[Figure 14 :- Select Devices Component Design 29](#_Toc195277606)

[Figure 15 :- Failed Recognition Component 30](#_Toc195277607)

[Figure 16 :- Dashboard Screen 31](#_Toc195277608)

[Figure 17 :- Devices Dashboard 32](#_Toc195277609)

[Figure 18 :- Roles Dashboard 33](#_Toc195277610)

[Figure 19 :- Users Management Dashboard 34](#_Toc195277611)

[Figure 20 :- User Details & Scanning the Card Modal 34](#_Toc195277612)

[Figure 21 :- User Photo Capture 35](#_Toc195277613)

[Figure 22 :- User Photo Cropping 35](#_Toc195277614)

[Figure 23 :- Edit User Modal 36](#_Toc195277615)

[Figure 24 :- Access Logs Dashboard 37](#_Toc195277616)

[Figure 25 :- Device Logs Dashboard 37](#_Toc195277617)

[Figure 26 :- Failed Logins Dashboard 38](#_Toc195277618)

[Figure 27 :- Admin User Flow 39](#_Toc195277619)

[Figure 28 :- Cleaner User Flow 40](#_Toc195277620)

[Figure 29 :- Chemist & Lab Technician User Flow 41](#_Toc195277621)

[Figure 30 :- Fast API Documents Screenshot 42](#_Toc195277622)

[Figure 31 :- Database UML Diagram 48](#_Toc195277623)

[Figure 32 :- Electrical Drawings 49](#_Toc195277624)

[Figure 33 :- Front View Design 50](#_Toc195277625)

[Figure 34 :- Side View Design 50](#_Toc195277626)

[Figure 35:- Trello Sprint Colors 51](#_Toc195277627)

[Figure 36 :- Sprint Execution 52](#_Toc195277628)

# Introduction

SecureGate a proof-of-concept project which is designed to create a secure, controlled environment inside a facility.

SecureGate uses multi-factor authentication using a combination of security card, Pin number and facial recognition to achieve secure identification.

SecureGate uses multi-level authorization with fine grained role-based access to devices on a user’s role classification. The devices the user is allowed access to are displayed within the application for the user to power on.

The system could be used by manufacturing and pharma businesses looking for a way to save on space and keep cost down by providing an environment which different teams have access to but ensuring only trained individuals have access to the different devices within the environment.

## Objective

My objective is to refactor Automated Management Systems (AMS) a project I completed in the Computer Systems Module of the Higher Diploma in Computer Science. I plan on removing several of its dependencies, rename it too SecureGate and making it more business friendly.

## Problem Statement

With the costs to a company to build an approved lab I decided to try come up with a way a company could use one lab which could be used by several different company departments/Employees. As training may be needed to use different devices there has to be a way to lock down the devices as well as access to the lab.

With data integrity and device history records needed for a most manufacturing and pharma businesses knowing that devices are only accessible to trained personnel is a must. Also keeping a record of who has access the lab and device is vital.

## Solution

SecureGate aims to address this by creating a system which controls both access to the lab and the devices within it using two step verification. A preapproved Company card and facial recognition software.

## AMS: What is it?

AMS was a IoT project which focused on security within a controlled environment. It addressed the granting of access to a Lab and the devices within the Lab.

AMS used Facial Recognition and cloud-based technologies to regulate access, ensuring only authorized personnel are granted access

The AMS system saved energy by powering off devices based on the users present in the room.

While multiple people may be present in the lab, AMS required facial recognition and/or a QR code to activate a device, ensuring that only approved users could turn on the device.

|  |  |
| --- | --- |
| **Original Project Documents** | |
| **DESCRIPTION** | **URL** |
| AMS Proposal | <https://github.com/ki321g/AMS/blob/main/Documentation/AssignmentProposal.pdf> |
| GitHUB Repository | <https://github.com/ki321g/AMS> |
| Video Walk Through | <https://youtu.be/SJLO5kULhWg> |

## Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Name** | **Description** |
| IoT | Internet Of Things |
| AMS | Automated Management Systems |
| VM | Virtual Machine |
| DC | Docker Container |
| API | Application Programming Interface |
| WiFi | Wireless Fidelity |
| LAN | Local Area Network |
| MQTT | Message Queuing Telemetry Transport |
| OpenCV | Open-Source Computer Vision Library |
| dlib's | dlib: A toolkit for machine learning and computer vision (not an acronym). |
| UI | User Interface |
| iOS | iPhone Operating System |
| DOM | Document Object Model |
| pHAT | Python Hardware Attachment on Top |
| USB | Universal Serial Bus |
| TPU | Tensor Processing Unit |
| NOIR | No Infrared |
| UML | Unified Modeling Language |
| UUID | Universally Unique Identifier |

# Research and Analysis

This section will describe the research and analysis phase of the project. It contains the Research I conducted, the existing solutions I found online, the tools and technologies I planned to use during the project.

## Research

My research really started back in October of 2024 when I decided to redevelop the AMS project and build a physical modal to demonstrate the project. I acquired a card reader in work and started looking up the documentation then developing a small python script to read the card.

When I developed AMS several times I updated and install different plugins and software on my Raspberry Pi which broke what I was trying to do so during my research for SecureGate I wanted to come up with a plan to try and help minimize the effects these issues had on the other components of the project so I decided to use Docker, using Portainer so I could easily manage my Docker containers.

While working on AMS I also had an issue with the Raspberry Pi Camera, Open CV (vdavidyang, 2024) and the then new Raspberry Pi 5. I wasted close to two weeks trying to get it working but when I plugged in a USB camera into the Raspberry Pi it started working straight away. This made it easy to decide to use a USB camera for SecureGate

I knew I wanted to use React and with a very simple google search for “React Webcam” I found the react-webcam (mozmorris, 2025) component. Using the demos provided I began developing a small application with react-webcam. Once I had an application working, I started looking at [deepface-react-ui](https://github.com/serengil/deepface-react-ui) (Serengil, 2024) to integrate facial detection and recognition.

I found that developing smaller applications as I went along researching the project really helped me decide on what I was going to do. It was looking at [deepface-react-ui](https://github.com/serengil/deepface-react-ui) (Serengil, 2024) example env file where I discovered Mediapipe (Google, 2025). After looking at the Mediapipe documentation and examples I added these to different components I developed within my small react applications improving as I went a long adding in Object Detection, Face Detection and Face landmark detection.

## Existing Solutions



[**SmartFace Access Control**](https://www.innovatrics.com/face-recognition-solutions/smartface-facial-access-control/)is a facial access control system that uses facial recognition to verify identities quickly and accurately. It’s features include

* **Registration:** used for visitor self-registration upon arrival. Allows for easy registration and verification using ID data supplied by the visitor.
* **Identification:** Proprietary facial identification algorithms suited for real-time access control.
* **Accuracy:** Highly accurate under varying lighting conditions and positioning
* **Security Measures:** Employs anti-spoofing, data encryption, and secure credential storage to protect biometric data.
* **Integration:** Can be easily accessed through its REST API making it possible to integrate into your custom applications



[**HIKVISION MinMoe Face Recognition Terminals**](https://www.hikvision.com/uk/products/Access-Control-Products/Face-Recognition-Terminals/) can achieve recognition speeds of 0.2 seconds while maintaining an accuracy of over 99%.

* **Registration:** Quick and easy registration via multiple access methods, and convenient device configuration via web, GUI or Hik-Connect App.
* **Accuracy:** Very fast facial recognition with accuracy over 99%
* **Security Measures:** Contains embedded anti-spoofing technology with all user data being encrypted for protection.
* **Integration:** Has five methods available for integration: Security API, Access Control Gateway API, Professional OpenAPI and Professional Database/CSV/TXT

A black background with white text

Description automatically generated

[Alcatraz AI Rock X](https://alcatraz.ai/rock-x) operates flawlessly in any lighting condition it is ensures reliable access control around the clock, irrespective of the ambient lighting.

It’s features include

* **Registration:** With Mobile enrolment both employees and visitors can enrol securely and quickly with their own devices.
* **Identification:** Provides Multi-Factor Authentication which includes PIN code and/or access card with facial biometrics.
* **Security Measures:** Implements end-to-end encryption, sophisticated anti-spoofing techniques, and secure edge processing to ensure biometric data integrity.

## Tools

I am planning on using some of the following tools this may change as I research the project a bit more.

### Proxmox

Proxmox delivers powerful, enterprise-grade solutions with full access to all functionality for everyone - highly reliable and secure. Proxmox Virtual Environment is a complete open-source platform for enterprise virtualization. With the built-in web interface, you can easily manage VMs and containers, software-defined storage and networking, high-availability clustering, and multiple out-of-the-box tools using a single solution.

(Proxmox.com, 2025)

### Portainer

Portainer is a platform for container management that works universally across different systems. With robust support for multiple clusters and devices, you can manage any environment. Whether it's Docker or Kubernetes, whether you're on a laptop, in a data center, in the cloud, or at the edge.

(Portainer.io, 2025)

### Docker

Docker is an open-source platform designed to simplify the development, deployment, and management of applications. It uses containerization, a technology that packages an application and its dependencies into a portable and isolated environment known as a container.

(Docker.com, 2021)

### Kubernetes

Kubernetes is an open-source platform designed to simplify the deployment, scaling, and management of containerized applications. It groups containers into logical units for easier management and discovery. Drawing on over 15 years of experience running production workloads at Google, Kubernetes combines proven techniques with innovative ideas from the broader community.

(Kubernetes.io, 2019)

### TinyTuya

This Python module allows you to control and monitor the status of Tuya compatible WiFi smart devices, such as plugs, switches and lights using either your local network or the TuyaCloud API.

While Tuya devices are built to communicate with the TuyaCloud, many also offer a local network API, enabling direct control without relying on the cloud. This module makes it easy to poll a device's status and send commands. TinyTuya can connect to the Tuya Cloud for status updates and device commands if needed.

(Cox, 2024)

### Supabase

Supabase is an open-source alternative to Firebase that lets you self-host projects featuring a Postgres database, user authentication, instant APIs, Edge Functions, real-time subscriptions, and storage. This platform empowers developers to rapidly build and scale applications by offering a complete set of integrated backend tools.

(Supabase.com, 2025)

### MQTT (Message Queuing Telemetry Transport)

MQTT (Message Queuing Telemetry Transport) is a messaging protocol designed for communication between devices in IoT and other systems. It provides a secure and reliable data transfer, making it an excellent choice for connected applications and remote device management.

(MQTT.org, 2022)

### OpenCV

OpenCV (Open Source Computer Vision Library) is a powerful open-source software library designed for real-time computer vision and image processing applications. It offers a comprehensive set of features and tools that make it a popular choice for applications ranging from object detection to facial recognition.

(OpenCV.org, 2019)

### Face-Recognition

Face Recognition is a simple face recognition library which can be used to

detect and manipulate faces either via Python or directly from the command line. It leverages dlib's advanced deep learning techniques to deliver cutting-edge face recognition accuracy, achieving 99.38% on the Labeled Faces in the Wild benchmark. This impressive level of performance makes it ideal for a wide range of real-world applications

(Geitgey, 2019)

### Fast API

FastAPI is a modern, high-performance web framework built for Python it excels at creating fast and efficient APIs. Its design prioritizes robustness, performance, and user-friendliness, ensuring that your APIs are both powerful and easy to work with.

(FastAPI, 2023)

### Deep Face

DeepFace is a Python framework for face recognition and analysis of facial attributes. It can operate as a server with an API, making it easy for your application to perform tasks such as face identification and attribute examination

(Ilkin, 2020)

### MediaPipe

MediaPipe provides a suite of libraries and tools to quickly apply artificial intelligence(AI) and machine learning(ML) in your applications. You can plug in MediaPipe solutions immediately and customize them to your own needs

(Google, 2025)

## Technologies

The following technologies were used in this project.

### Python

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn’t specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today.

### React

ReactJS is a component-based JavaScript library used to build dynamic and interactive user interfaces. It simplifies the creation of single-page applications (SPAs) with a focus on performance and maintainability.

* It is developed and maintained by Facebook.
* Uses a virtual DOM for faster updates.
* Supports a declarative approach to designing UI components.
* Ensures better application control with one-way data binding.

### Bash Scripting

Bash scripts are typically used for system administration, deployment automation, and repetitive task execution in development workflows. I used it scripting to start my frontend and backend applications on the pi and helping build the Docker containers.

## Equipment

The system will incorporate the following devices

|  |  |
| --- | --- |
| **Name** | **Image** |
| Raspberry Pi 4 & 5 | Figure :- Raspberry Pi 4 & 5 |
| Lrtzcbi Touchscreen Portable Monitor 14 Inch | Figure :- Lrtzcbi Touchscreen Portable Monitor 14 Inch |
| pHAT Stack Kit for Raspberry Pi | PIMORONI PIM322  Figure :- pHAT Stack Kit for Raspberry Pi |
| Microsoft Lifecam HD-3000 | Figure :- Microsoft Lifecam HD-3000 |
| ELEGOO Upgraded Electronics Fun Kit | Figure :- ELEGOO Upgraded Electronics Fun Kit |
| Mini Portable USB Night Light | Figure :- Mini Portable USB Night Light |
| Woox R4785 smart plug | Figure :- Woox R4785 smart plug |
| Rekavin USB Plug | Figure :- Rekavin USB Plug |

# Design & Modeling

## High Level Overview

Figure 9 gives a high-level overview of the Raspberry Pi Setup.

It displays how each of the following communicate

* Docker Containers
  + Portainer running on port 9000 used to manage the docker container setup.
  + DeepFace running on port 5000 used for facial recognition by sending it two base64 images, one stored in the Supabase database with the one of the person trying to gain access.
  + TinyTuya running on port 8888 used to control the different devices.
  + Supbase running on port 8000 used for the Realtime Database.
* The Backend FastAPI script running on port 3002 used to communicate with the card reader, the maglock and the different docker containers installed.
* Frontend Application running on port 5175 used for user interaction with he system.

A screenshot of a computer

Description automatically generated

Figure :- High Level Raspberry Pi Setup

## User Persona

In this section you’ll find details of the different User persona’s and the users Flow/Journey.

### Administrator

#### Persona

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Name** | Sarah Johnson |
| **Age Range** | 35 |
| **Role** | ADMIN |
| **Description** | * Sarah is responsible for managing the SecureGate system. * She has full access to the Admin backend to add, delete, and update users, devices, and roles. * She doesn’t have access to the Lab. * She does not have access to any physical devices in the lab.Sarah ensures that the system is running smoothly and that all users have the appropriate permissions. | | |
| **Key Responsibilities** | * Manage user accounts (create, update, delete). * Manage devices and roles. * Monitor access logs and device logs for security and compliance. | | |
| **Access** | * No access to physical devices. * Full access to the backend system for user and device management. | | |

#### User Flow/Journey

Sarah approaches the control panel, and the application detects her presence using the camera. The screen automatically transitions from the Logo Screen to the Card Scan Screen, prompting her to scan her card. Once scanned, the system retrieves her stored details and profile image.

Next, the application attempts to detect her face. Once identified, it captures a live photo and compares it with the stored image using DeepFace. If the images match, Sarah gains access to the Admin Dashboard, where she can carry out her tasks. However, this does not grant her access to the lab itself.

### Chemist

#### Persona

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Name** | Dr. Michael Power |
| **Age Range** | 48 |
| **Role** | CHEMIST |
| **Description** | * Dr. Power is a senior chemist who oversees all lab operations. * He requires access to all devices in the lab to conduct experiments and monitor results. * He is highly skilled and trusted with sensitive equipment. | | |
| **Key Responsibilities** | * Conduct experiments using lab devices. * Monitor and analyze data from devices. * Ensure devices are used correctly and maintained properly. | | |
| **Access** | * Access to all devices in the lab. | | |

#### User Flow/Journey

Dr. Power approaches the control panel, and the application detects his presence using the camera. The screen automatically transitions from the Logo Screen to the Card Scan Screen, prompting him to scan his card. Once scanned, the system retrieves his stored details and profile image.

Next, the application attempts to detect his face. Once identified, it captures a live photo and compares it with the stored image using DeepFace. Once confirmed, the screen switches to the Device Screen. On the Device Screen, Dr. Power selects the lab equipment he needs for his experiments. Since he has unrestricted access, he can turn on any or all devices at once. After making his selections and confirming, the chosen devices power on, and the lab door unlocks.

### Lab Technician (Senior)

#### Persona

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Name** | Jenifer Grace |
| **Age Range** | 33 |
| **Role** | Senior Lab Technician |
| **Description** | * Emily is an experienced lab technician who assists the chemist and manages day-to-day lab operations. * She has access to most devices but not the most sensitive or high-risk equipment. * She is responsible for training junior technicians and ensuring lab protocols are followed. | | |
| **Key Responsibilities** | * Operate and maintain lab devices. * Train junior technicians. * Assist the chemist with experiments. | | |
| **Access** | * Access to most devices, except for high-risk or sensitive equipment. | | |

#### User Flow/Journey

As Jenifer approaches the control panel, the system detects her presence, transitioning from the Logo Screen to the Card Scan Screen.

She scans her access card, allowing the system to retrieve her stored details and image. The application then switches to the User Details Screen, where it verifies her identity by comparing her live image with the stored one.

Once confirmed, she proceeds to the Device Screen. On the Device Screen, Jenifer selects the equipment she needs to operate. After making her selections and confirming, the system powers on the approved devices.

With the necessary devices activated, the lab door unlocks, granting her access.

### Lab Technician (Junior)

#### Persona

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Name** | James O’Connor |
| **Age Range** | 24 |
| **Role** | Junior Lab Technician |
| **Description** | * James is a recent graduate who is still learning the ropes of lab operations. * He has limited access to devices and is supervised by senior technicians and the chemist. * He is eager to learn and follows strict protocols. | | |
| **Key Responsibilities** | * Assist senior technicians with device operation. * Follow lab protocols and safety guidelines. * Perform routine tasks under supervision. | | |
| **Access** | * Access to basic and low-risk devices. * Cannot access high-risk or sensitive equipment. | | |

#### User Flow/Journey

James approaches the control panel, the system detects his presence, transitioning from the Logo Screen to the Card Scan Screen.

He scans his access card, prompting the system to retrieve his stored details and image. The application then switches to the User Details Screen, where his identity is verified through facial recognition. Once confirmed, he moves on to the Device Screen. On the Device Screen, James chooses the on the device.

With the device turn on, the lab door unlocks, granting him access to the lab.

### Cleaner

#### Persona

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Name** | Maria Walsh |
| **Age Range** | 51 |
| **Role** | CLEANER |
| **Description** | * Maria is responsible for cleaning and maintaining the lab environment. * She does not have access to any lab devices, as her role does not require it. * She works during off-hours to ensure the lab is clean and safe for the next day. | | |
| **Key Responsibilities** | * Clean lab equipment and workspaces. * Dispose of waste materials safely. * Ensure the lab is tidy and organized. | | |
| **Access** | * No access to any lab devices. * Can only access the lab during designated cleaning hours. | | |

#### User Flow/Journey

As Maria approaches the control panel during her designated cleaning hours, the system detects her presence, transitioning from the Logo Screen to the Card Scan Screen.

She scans her access card, prompting the system to retrieve her stored details and image. The application then switches to the User Details Screen, where her identity is verified through facial recognition. Once confirmed, the system unlocks the lab door, granting her access to perform her cleaning duties.

he does not have access to any lab devices, as her role does not require it.

## User Interface Design

### Front-End Application Interface Design

The Front-End Interface designs are the designs of application that everyone would use to unlock and select the device they want to turn on.

#### Logo Screen

The Logo Screen, figure 10 below, contains a hidden WebCam Component which will uses Mediapipe’s object detection to monitor for a person. If a person is detected the screen moves onto the Card Scanning Screen

A screen shot of a computer

Description automatically generated

Figure :- Logo Screen Design

#### Card Scanning Screen

The Scan Card Component, figure 11, stays on the screen until a card is scanned. Once a card is scanned the application will make an API call to the SupaBase database and pull back the relevant user details.

A screen shot of a computer

Description automatically generated

Figure :- Card Scanning Design

#### Enter Pin Number Screen

The Enter Pin Component, figure 12, stays on the screen until a valid Pin is Entered. Once a Pin is entered it is compared with the users Pin and if it is valid the application moves onto the facial recognition screen.

A screenshot of a computer

Description automatically generated

Figure :- Enter Pin Component Design

#### Facial Recognition Screen

Once the User details are populated the Facial Detection begins on the Facial Recognition Component, figure 13. When the application detects a face, it will trigger taking a screen capture and saving as a base64 image in a variable. The application will then send both images to the DeepFace server container to check if they are the same person. While this is happening the application will draw the facial landmark mesh on the persons face and the READY button will change to ANALIZING. Once the application gets a reply it will display SUCCESS or FAIL. If a fail occurs there will be a second attempt made to confirm the user.

A screen shot of a computer

Description automatically generated

Figure :- Facial Recognition Component Design

#### Selecting Device

If the user is successfully authenticated the list of devices, they can start will be displayed on the Select Devices Component, figure 14, for them to select then press the START button. Once the START button is pressed the Door will open and the Device will start.

A screenshot of a computer

Description automatically generated

Figure :- Select Devices Component Design

#### Failed Recognition Screen

If Facial Detection fails, then the Failed Recognition Component, figure 15, is displayed. The user gets three chances and then their account gets locked out. If Facial Detection passes the count is reset on the device selection component.

A screenshot of a computer

Description automatically generated

Figure :- Failed Recognition Component

### Back-End Application Admin Dashboard Interface Design

The Back-End Interface designs are the designs of application that the admin will use to ADD, DELETE, UPDATE the users, devices and roles.

#### Dashboard

On the admin dashboard, Figure 16, you are given an overview of all Users, Roles and Devices. Followed by Labe Access & Device Analytics.

A screenshot of a computer

Description automatically generated

Figure :- Dashboard Screen

#### Devices Dashboard

The Devices dashboard, Figure 17 is where you control the devices on the system. The Status column will let the Admin know if the device is ON or OFF, if ON the Admin is prevented from editing or deleting the device.

**Functionality:**

* **Add/Edit/Delete** a device
* **Assign** the Roles to the device

A screenshot of a computer

Description automatically generated

Figure :- Devices Dashboard

#### Roles Dashboard

The Roles dashboard, Figure 18, is where you control the Roles

**Functionality:**

* **Add/Edit/Delete** a role
* **Assign** the devices to different Role

A screenshot of a computer

Description automatically generated

Figure :- Roles Dashboard

#### Users Dashboard

The Users Management dashboard, Figure 19, is where you control the Users

A screenshot of a computer

Description automatically generated

Figure :- Users Management Dashboard

**Functionality:**

* **Add** a user

When adding a user there is a twostep process

1. Adding the User Details & Scanning the Card, Figure 20

A screenshot of a computer

Description automatically generated

Figure :- User Details & Scanning the Card Modal

1. Adding the User Photo which itself is another two steps.

Capturing the image from the WebCam, figure 21

A screenshot of a computer

Description automatically generated

Figure :- User Photo Capture

Zooming in on the Users face and cropping the image, figure 22.

A screenshot of a computer

Description automatically generated

Figure :- User Photo Cropping

* **Edit** a user

When editing a user, figure 23, you can change all the user details.

**A screenshot of a computer

Description automatically generated**

Figure :- Edit User Modal

* **Delete** a user

The Delete user is a simple modal which Disables the user rather than deletes them. I did it this way because of data integrity and needing to keep the log files linked to the user.

#### Access Logs Dashboard

The Access Logs dashboard, figure 24, displays a table of access logs showing user entry attempts with details including user photos, names, access status (granted/denied), timestamps, and comments.

A screenshot of a computer

Description automatically generated

Figure :- Access Logs Dashboard

#### Device Logs Dashboard

The Device Logs dashboard, figure 25, displays a table of device-related activities, displaying information about device operations, associated users, actions performed, status results, and timestamps.

A screenshot of a computer

Description automatically generated

Figure :- Device Logs Dashboard

#### Failed Logins Dashboard

The Failed Logins dashboard, figure 26, provides a security management interface that displays users with failed login attempts, allowing administrators to view user information and recent access logs for each affected user. It offers functionality to reset locked user accounts, helping administrators quickly respond to locked account calls.

A screenshot of a computer

Description automatically generated

Figure :- Failed Logins Dashboard

### User Case Diagram

#### Admin

This diagram shows the Admins flow, figure 27, within the Application.

**A screenshot of a computer

Description automatically generated**

Figure :- Admin User Flow

#### Cleaner

This diagram shows the Cleaners flow, figure 28, within the Application.

**A screenshot of a computer

Description automatically generated**

Figure :- Cleaner User Flow

#### Chemist & Lab Technician’s

This diagram shows the Chemist & Lab Technician’s user flow, figure 29, within the Application.

*A screenshot of a computer

Description automatically generated*

Figure :- Chemist & Lab Technician User Flow

## Back-End Design

### Back-End API Design

Figure 30 below display my custom designed Fast API Documents page.

A screenshot of a computer

Description automatically generated

Figure :- Fast API Documents Screenshot

### Database Schema Design

Below is a detailed breakdown of the database schema, including relationships between tables and a UML diagram to visualize the database model.

**Database Schema Breakdown**

**Schema Name:** public

1. **ROLES Table**

**Purpose:** Stores the roles that users can have (e.g., Admin, Chemist, Lab Technician, Cleaner).

**Table Fields:**

* + **role\_id** (UUID, Primary Key): Unique identifier for the role.
  + **role\_name** (TEXT, Unique): Name of the role.
  + **description** (TEXT): Optional description of the role.
  + **created\_at** (TIMESTAMP WITH TIME ZONE): Timestamp when the role was created.

1. **DEVICES Table**

**Purpose:** Stores information about devices in the lab.

**Table Fields:**

* + **device\_id** (UUID, Primary Key): Unique identifier for the device.
  + **device\_name** (TEXT): Name of the device.
  + **description** (TEXT): Optional description of the device.
  + **created\_at** (TIMESTAMP WITH TIME ZONE): Timestamp when the device was added.
  + **deviceid** (TEXT): Tiny Tuya Device ID.
  + **ip\_address** (INET): Tiny Tuya Device IP Address.
  + **local\_key** (TEXT): Tiny Tuya Device Local Key.
  + **version** (TEXT): Tiny Tuya Device version.

1. **ROLE\_TO\_DEVICE Table**

**Purpose:** Stores which roles have access to which devices.

**Table Fields:**

* + **device\_id**: (UUID, Foreign Key): References devices(device\_id) to assign a device to the role.
  + **Role\_id**: (UUID, Foreign Key): References roles(role\_id) to assign a role to the device.
  + **PRIMARY KEY** (role\_id, device\_id) Composite primary key

1. **USERS Table**

**Purpose:** Stores user information, including their role and access details.

**Table Fields:**

* + **uid** (UUID, Primary Key): Unique identifier for the user.
  + **first\_name** (TEXT): User's first name.
  + **last\_name** (TEXT): User's last name.
  + **email** (TEXT, Unique): User's email address (validated with a regex check).
  + **password** (TEXT): Hashed password for authentication.
  + **phone\_number** (TEXT): User's phone number.
  + **role\_id**(UUID, Foreign Key): References roles(role\_id) to assign a role to the user.
  + **card\_id** (TEXT, Unique): Unique identifier for the user's access card.
  + **user\_picture** (TEXT): Base64-encoded image for facial recognition.
  + **last\_seen\_at** (TIMESTAMP WITH TIME ZONE): Timestamp of the user's last activity.
  + **created\_at** (TIMESTAMP WITH TIME ZONE): Timestamp when the user was created.
  + **status** (TEXT): Users Status, Active, InActive or Disabled

1. **ACCESS\_LOGS Table**

**Purpose:** Logs access attempts to the lab.

**Table Fields:**

* + **log\_id** (UUID, Primary Key): Unique identifier for the log entry.
  + **user\_id** (UUID, Foreign Key): References users(uid) to identify the user.
  + **success** (BOOLEAN): Indicates whether the access attempt was successful.
  + **notes** (TEXT): Reason for access failure (if applicable).
  + **user\_picture** (TEXT): Base64-encoded image captured during the access attempt.
  + **created\_at** (TIMESTAMP WITH TIME ZONE): Timestamp of the access attempt.

1. **DEVICE\_LOGS Table**

**Purpose:** Logs actions performed on devices (e.g., turning on/off).

**Table Fields:**

* + **log\_id** (UUID, Primary Key): Unique identifier for the log entry.
  + **device\_id** (UUID, Foreign Key): References devices(device\_id) to identify the device.
  + **user\_id** (UUID, Foreign Key): References users(uid) to identify the user performing the action.
  + **action** (TEXT): Description of the action performed (e.g., "Turn On", "Turn Off").
  + **status** (BOOLEAN): Status of the device after the action (e.g., true for "On", false for "Off").
  + **notes** (TEXT): Additional notes about the action.
  + **created\_at** (TIMESTAMP WITH TIME ZONE): Timestamp of the action.

1. **FAILED\_ATTEMPTS Table**

**Purpose:** Keeps a record of the users failed access attempts. Entries are deleted on a successfully attempt.

**Table Fields:**

* + **user\_id** (UUID, Primary Key, Foreign Key): References users(uid) to identify the user.
  + **failed** (int4): Holds the number of failed attempts for a user.

1. **GET\_DEVICES\_BY\_USER\_UID Function**

**Purpose:** The primary purpose of the function is to fetch details about devices that are linked to a user through their role in the system

**Output:** The function returns a set of records containing the following information for each device associated with the user:

**Returned Fields from devices table:**

* + **device\_id** (UUID): Unique UID to identify the device.
  + **deviceid** (TEXT): Tiny Tuya Device ID
  + **device\_name** (TEXT): Name of the device.
  + **description**: (TEXT): Description of the device.

**Relationships Between Tables**

1. **users and roles:**
   * **Relationship**: Many-to-One.
   * **Description**: Many users can have the same role.
   * **Foreign Key**: users.role\_id references roles.role\_id.
2. **access\_logs and users:**
   * **Relationship**: One-to-Many.
   * **Description**: One user can have multiple access log entries.
   * **Foreign Key**: access logs.user\_id references users.uid.
3. **access\_logs and devices:**
   * **Relationship**: One-to-Many (optional).
   * **Description**: One device can have multiple access log entries.
   * **Foreign Key**: access\_logs.device\_id references devices.device\_id.
4. **device\_logs and users:**
   * **Relationship**: One-to-Many.
   * **Description**: One user can perform multiple actions on devices.
   * **Foreign Key**: device\_logs.user\_id references users.uid.
5. **device\_logs and devices:**
   * **Relationship**: One-to-Many.
   * **Description**: One device can have multiple action logs.
   * **Foreign Key**: device\_logs.device\_id references devices.device\_id.
6. **roles and role\_to\_device:**
   * **Relationship**: Many-to-Many.
   * **Description**: One role can have access to many devices, and each device can be accessed by multiple roles.
   * **Foreign** **Key**: role\_id references roles.role\_id.
7. **devices and role\_to\_device:**
   * **Relationship**: Many-to-Many.
   * **Description**: One device can be accessed by many roles, and each role can have access to multiple devices.
   * **Foreign Key**: device\_id references devices.device\_id
8. **failed\_attempts and users:**
   * **Relationship**: One-to-One.
   * **Description**: Each entry in the failed\_attempts table corresponds to a user in the users table, identified by the user\_id. This allows the system to maintain a count of failed login attempts for each user, which is used for locking accounts.
   * **Foreign Key**: failed\_attempts.user\_id references users.uid.

**Database UML Diagram**

Database UML Diagram is displayed below in figure 31

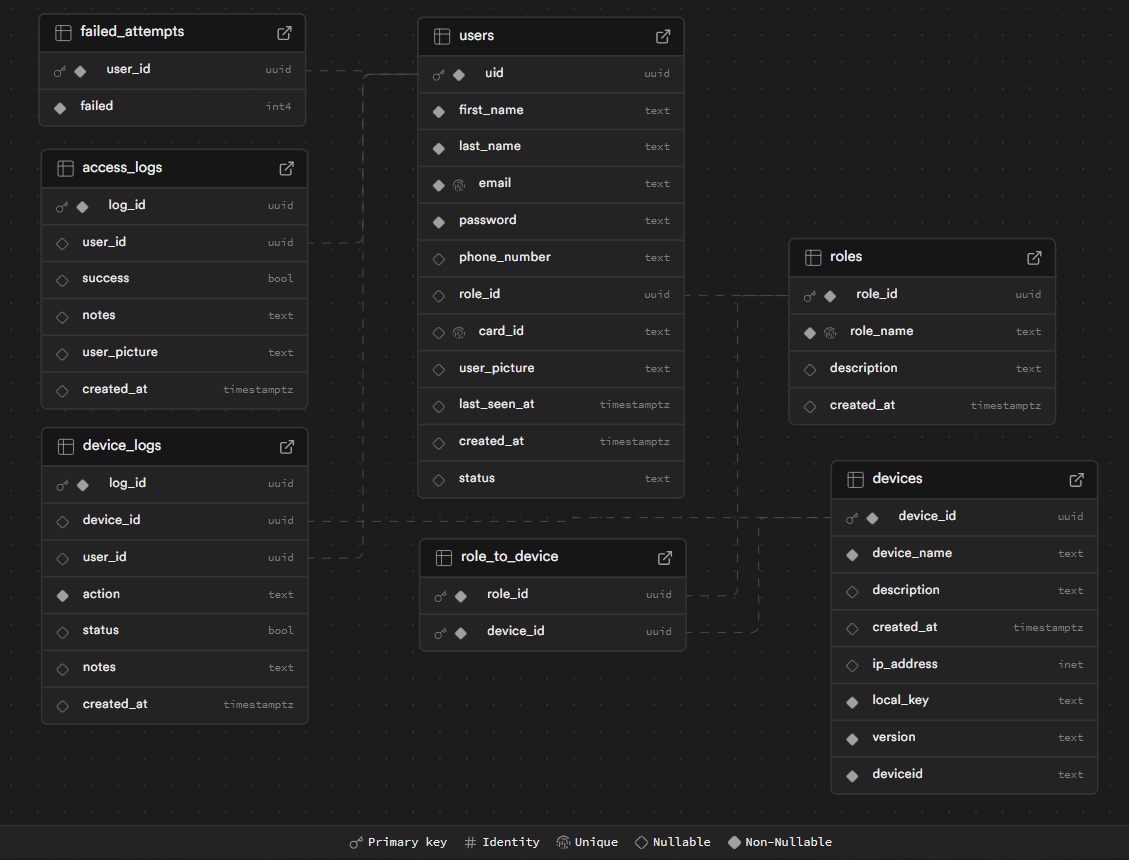


Figure :- Database UML Diagram

## Physical Model Design

Below you will find the drawings for the physical design for the project.

### Electrical Drawing

Figure 32 show the electrical drawings for the project

A computer screen shot of a computer

Description automatically generated

Figure :- Electrical Drawings

### Front View Design

Figure 33 shows the Front View Design of the model

A close-up of a door

Description automatically generated

Figure :- Front View Design

### Side View Design

Figure 34 shows the Side View Design of the model

A diagram of a door structure

Description automatically generated

Figure :- Side View Design

# Project Methodology & Execution

For the project, I chose the Agile software development lifecycle (SDLC). This approach is all about making progress in short, manageable sprints. I planned the project in six separate two week sprints, each focusing on specific deliverables and improvements. On my Trello board I left the sixth sprint empty to add things to as I went about the project. This process kept the project flexible and responsive to changes, while also maintaining a steady pace of development.

I used Trello (Trello Board for project <https://trello.com/b/EV3ZwiZu>) for project management, it was incredibly beneficial for tracking all the different tasks that needed completing in the SecureGate project. Color coordinating the six sprints keeping all the tasks related to each sprint the same color. This helped me manage the state of each sprint as I progressed through each sprint. Figure 35 below shows the colors I chose for each sprint

A screenshot of a cell phone

AI-generated content may be incorrect.

Figure :- Trello Sprint Colors

In Figure 36 I show the titles of each sprint and the order I completed them in. As you can see things didn’t go to plan. I needed help both wiring the electrical box and building the physical model which pushed up everything I had planned in Sprint#4 to the front as my help had some free time early on in my project.

Screens screenshot of a computer screen

AI-generated content may be incorrect.

Figure :- Sprint Execution

In the end I only used five of the sprints planned, I did move on from one sprint to the next even when I didn’t complete the sprint but keep revisiting tasks of other sprints until I had completed them. I also found I discovered tasks which I had not thought of as I worked on other sprints and just kept adding them to the boards.

At the end now I can see that putting a bit more effort into the planning could have made the project go a bit smoother but I adapted to what I had done and am proud of what I achieved

# Review, Reflection & Further work

This section contains a project review, reflection, and further work. It will outline what I learned, what I would do differently, problems encountered and how they were handled.

## Review

I enjoyed this project immensely and I think that was down to the fact I built a working model to demonstrate the application working from the scanning of the user’s card to facial recognition to the opening of the door and turning on the selected device(s).

I believe I achieved what I set out to do which was create a working proof of concept for an IoT security project for granting controlled access to infrastructure using multi factor authentication.

## Reflection

### What I have learned?

Working on this project, I learnt a lot about different technologies on top of what I learnt during the course.

Docker, Supabase, Python, FastAPI, React and Material UI are just some of the technologies I learnt about.

Failing to plan is planning to fail, I honestly did not plan the project correctly. I didn’t stick to my sprints and did different bits of the project at different times. I completed Sprint four before sprint three, but this was because I needed help with different parts of the project and had to do it when others where free.

### What I would do differently?

Use a mini-PC rather than a Raspberry PI. I found I was unplugging the Pi multiple times a night as I was programming. This was down to using SSH into the PI while editing the code, running the application and the backend API.

Would a PI Camera work better than the USB one? I would try test the PI camera instead of the USB it my not use up as many resources.

### What went wrong?

I encountered many problems along the way. Problems with schema design, problems with the Raspberry Pi, problems with React, problems with nearly every little thing. All these problems lead to watching many different YouTube videos on the topic or asking AI for a little help.

Time was wasted trying to get both my backend API and my frontend working in a Docker container on the raspberry pi. I couldn’t get either my card reader or the Web Camera to work properly between the two different containers. In the end what I did was install the API on my raspberry pi in a virtual environment and my frontend react app on the pi also.

Further on in development there was times that my react application would crash and freeze the raspberry pi after running through the facial recognition only two or three times. I redeveloped the component several times and found that once I removed the hidden web camera component which I had on the logo screen which automatically moved the user onto the next component things got better.

The Pi still freezes but not as often. I wanted to fix this, so I tested the application recording the errors in the console as I went in a separate document. I then used the Cody Visual Studio extension to help with all five errors I recorded. Most of them were related and Cody helped me redevelop my Web Camera Component, so the issues didn’t occur as much. It can still happen, but I believe this is because of the raspberry Pi not the application itself.

## Further Work

I missed one big feature in this project which would be the exiting of the room. Therefore, further work would involve the addition of another card reader which would be used by the employee to exit the room and turn on the devices they had turned on when entering the room.

This would involve adding a new table to keep track of the devices currently on and what user has them turned on, then once the user scans the card reader it opens the door and the devices the user had turned on turn off. Also the entries in this new table related to the devices the user had turned on get deleted.

Also during my research I thought it would have been nice to add in AI to the application and have the application talk to the user asking them to Scan there car, enter there pin number and look at the camera, I had planned to have AI decide if the user needed to come closer turn there head slightly to ensure a better picture. This didn’t happen as the other features took longer to develop and it would be nice to come back to this in the future and add something like it.

# Citations/References

Cox, J., 2024. *jasonacox/tinytuya.* [Online]   
Available at: https://github.com/jasonacox/tinytuya  
[Accessed 7 January 2025].

Docker.com, 2021. *Docker Desktop overview.* [Online]   
Available at: https://docs.docker.com/desktop/  
[Accessed 6 January 2025].

FastAPI, 2023. *FastAPI.* [Online]   
Available at: https://fastapi.tiangolo.com/  
[Accessed 7 January 2025].

Geitgey, A., 2019. *ageitgey/face\_recognition.* [Online]   
Available at: https://github.com/ageitgey/face\_recognition  
[Accessed 7 January 2025].

Google, 2025. *MediaPipe Solutions guide | Edge.* [Online]   
Available at: https://ai.google.dev/edge/mediapipe/solutions/guide  
[Accessed 8 January 2025].

Ilkin, S., 2020. *serengil/deepface.* [Online]   
Available at: https://github.com/serengil/deepface  
[Accessed 8 January 2025].

Kubernetes.io, 2019. *Production-Grade Container Orchestration.* [Online]   
Available at: https://kubernetes.io/  
[Accessed 6 January 2025].

mozmorris, 2025. *React WebCam.* [Online]   
Available at: https://www.npmjs.com/package/react-webcam  
[Accessed 20 01 2025].

MQTT.org, 2022. *MQTT - The Standard for IoT Messaging.* [Online]   
Available at: https://mqtt.org/  
[Accessed 7 January 2025].

OpenCV.org, 2019. *OpenCV library.* [Online]   
Available at: https://opencv.org/  
[Accessed 7 January 2025].

Portainer.io, 2025. *Open Source Container Management GUI for Kubernetes, Docker, Swarm.* [Online]   
Available at: https://www.portainer.io/  
[Accessed 6 January 2025].

Proxmox.com, 2025. *Powerful open-source server solutions.* [Online]   
Available at: https://www.proxmox.com/en/  
[Accessed 6 January 2025].

Serengil, S. a. O. A., 2024. A Benchmark of Facial Recognition Pipelines and Co-Usability Performances of Modules. *Journal of Information Technologies,* Volume 17, pp. 95-107.

Supabase.com, 2025. *The Open Source Firebase Alternative.* [Online]   
Available at: https://supabase.com/  
[Accessed 7 January 2025].

vdavidyang, 2024. *Raspberry Pi 5+Module 3 camera can not read frame with openv.* [Online]   
Available at: https://github.com/opencv/opencv/issues/25072  
[Accessed 1 02 20025].