**SecureGate**

**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:** |  | **1.0** |
| **Effective Date:** |  | **23rd January 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: | 23/01/2025\_\_\_\_\_\_ |

|  |
| --- |
| **Acknowledgements** |
| **TBD** |

|  |
| --- |
| **Preface** |
| **TBD** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| **Revision #** | Description | **Author** | **Effective Date** |
| 1.0 | Initial Draft Submission | Kieron Garvey | 23/01/2025 |

**Table of Contents**

[1 Introduction 4](#_Toc189508452)

[1.1 Objective 4](#_Toc189508453)

[1.2 Purpose 4](#_Toc189508454)

[1.3 AMS (Original) Proposal 4](#_Toc189508455)

[1.4 Acronyms and Abbreviations 5](#_Toc189508456)

[2 Research and Analysis 5](#_Toc189508457)

[2.1 Context 5](#_Toc189508458)

[2.2 Existing Solutions 5](#_Toc189508459)

[2.3 Tools 6](#_Toc189508460)

[2.3.1 Proxmox 6](#_Toc189508461)

[2.3.2 Portainer 6](#_Toc189508462)

[2.3.3 Docker 6](#_Toc189508463)

[2.3.4 TinyTuya 6](#_Toc189508464)

[2.3.5 Supabase 6](#_Toc189508465)

[2.3.6 MQTT (Message Queuing Telemetry Transport) 7](#_Toc189508466)

[2.3.7 OpenCV 7](#_Toc189508467)

[2.3.8 Face-Recognition 7](#_Toc189508468)

[2.3.9 Fast API 7](#_Toc189508469)

[2.4 Technologies 8](#_Toc189508470)

[2.4.1 Python 8](#_Toc189508471)

[2.4.2 React 8](#_Toc189508472)

[2.4.3 React Native 8](#_Toc189508473)

[2.4.4 Flutter 8](#_Toc189508474)

[2.5 Equipment 8](#_Toc189508475)

[Rekavin USB Plug 11](#_Toc189508476)

[3 Design & Modeling 12](#_Toc189508477)

[3.1 User Persona 12](#_Toc189508478)

[3.1.1 Admin 12](#_Toc189508479)

[3.1.2 Chemist 13](#_Toc189508480)

[3.1.3 LAB TECHNICIAN (Senior) 14](#_Toc189508481)

[3.1.4 LAB TECHNICIAN (Junior) 15](#_Toc189508482)

[3.1.5 Cleaner 16](#_Toc189508483)

[3.2 User Interface Design 17](#_Toc189508484)

[3.2.1 User Case Diagram 17](#_Toc189508485)

[3.2.2 Front-End Interface Design 17](#_Toc189508486)

[3.2.3 Back-End Interface Design 17](#_Toc189508487)

[3.3 Database Schema 19](#_Toc189508488)

[3.4 Physical Model Design 24](#_Toc189508489)

[3.4.1 Electrical Drawing 24](#_Toc189508490)

[3.4.2 Front View Design 25](#_Toc189508491)

[3.4.3 Side View Design 25](#_Toc189508492)

[4 Conclusion + Further work 26](#_Toc189508493)

[4.1 Reflection 26](#_Toc189508494)

[4.1.1 What I have learned 26](#_Toc189508495)

[4.1.2 What I would do differently 26](#_Toc189508496)

[5 References 26](#_Toc189508497)

[6 Attachments 26](#_Toc189508498)

[6.1 Attachment#1:- Electrical Drawing 27](#_Toc189508499)

[6.2 Attachment#2:- 28](#_Toc189508500)

[6.3 Attachment#3:- 28](#_Toc189508501)

# Introduction

## Objective

Automated Management Systems (AMS) was a project I created for the Computer Systems & Networks Module of the Higher Diploma in Computer Science course.

My objective is to refactor this project removing several of its dependencies, rename it too SecureGate and making it more business friendly.

## Purpose

## AMS (Original) Proposal

AMS is an innovative IoT project focused on enhancing security and privacy in a controlled company environment. AMS addressed the challenges of granting controlled access to a room and the devices within the room without compromising sensitive information.

AMS integrates facial recognition and remote access technologies to regulate entry, ensuring only authorized personnel enter the controlled environment. It employs sophisticated security monitoring to restrict access to specific devices, preventing unauthorized usage within the environment.

The system optimizes energy consumption by selectively powering on devices based on user permissions. When multiple individuals are present, AMS requires facial recognition or QR code authentication for device activation, ensuring access control.

Additionally, AMS implements secure login protocols for devices, requiring facial recognition and/or QR code authentication, thereby enhancing security measures within the controlled environment. By providing detailed access control and authentication mechanisms, AMS ensures heightened security and restricted access while optimizing resource utilization.

|  |  |
| --- | --- |
| **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** |
| 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

planning

## Context

planning

## Existing Solutions

planning

## 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. The software-defined and open platforms are easy to deploy and manage. 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.

**URL:** <https://www.proxmox.com/en/>

### *Portainer*

Portainer is a universal container management platform. It's multi-cluster and multi-device support which means you can manage environments of any type, anywhere (Docker and Kubernetes, running on dev laptops, in your DC, in the cloud, or at the edge), and we don't require you to run any specific Kubernetes distro.

**URL:** <https://www.portainer.io/>

### *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.

### *TinyTuya*

This python module controls and reads state of Tuya compatible WiFi Smart Devices (Plugs, Switches, Lights, Window Covers, etc.) using the local area network (LAN) or the cloud (TuyaCloud API).

**URL:** <https://github.com/jasonacox/tinytuya>

### *Supabase*

Supabase is an open-source Firebase alternative.

Start your project with a Postgres database, Authentication, instant APIs, Edge Functions, Realtime subscriptions, Storage, and Vector embeddings.

**URL:** <https://supabase.com/>

### *MQTT (Message Queuing Telemetry Transport)*

MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol designed for efficient communication between devices in IoT and other applications. It enables secure, reliable data transfer with minimal bandwidth usage, ideal for interconnected systems and remote device management.

**URL:** <https://mqtt.org/>

### *OpenCV*

OpenCV (Open Source Computer Vision Library) is a powerful open-source software library designed for real-time computer vision and image processing applications.

**URL:** <https://opencv.org/>

### *Face-Recognition*

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library.

Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.

**URL:** <https://github.com/ageitgey/face_recognition>

### *Fast API*

FastAPI is a modern, high-performance web framework for building APIs with Python. It is designed to deliver robust, efficient, and user-friendly APIs

**URL:** <https://fastapi.tiangolo.com/>

## Technologies

Again, I am planning on using the following technologies, but this may change as I research the project a bit more.

### *Python*

Python is a high-level, versatile programming language known for its simplicity and readability. It's used across various domains such as web development, data analysis, artificial intelligence, scientific computing, and more. Python's clean syntax and extensive libraries make it beginner-friendly while offering powerful capabilities for building applications, automating tasks, handling data, and implementing algorithms efficiently.

### *React*

React is a popular JavaScript library for building user interfaces, primarily for web applications. Developed and maintained by Facebook, React enables developers to create reusable UI components that efficiently update and render when data changes.

### *React Native*

React Native is a framework derived from React that focuses on building mobile applications for iOS and Android using JavaScript. Instead of rendering to the web's DOM, React Native components translate to native widgets, delivering a performance and feel akin to apps developed with native technologies.

### *Flutter*

Flutter is an open-source UI toolkit created by Google for crafting natively compiled applications for mobile, web, and desktop from a single codebase. Unlike React Native, Flutter uses Dart, a programming language also developed by Google. It boasts its own rendering engine, Skia, allowing for complete control over every pixel rendered on the screen. This ensures highly customizable, visually rich UIs that look and feel consistent across platforms.

## Equipment

The system will incorporate the following devices

|  |  |
| --- | --- |
| **Name** | **Image** |
| Raspberry Pi 4 & 5 |  |
| Lrtzcbi Touchscreen Portable Monitor 14 Inch |  |
| pHAT Stack Kit for Raspberry Pi | PIMORONI PIM322 |
| Raspberry Pi Camera Module 3 Wide NOIR | RASPBERRY-PI SC0875 |
| USB Accelerator Edge TPU Coporocessor | CORAL G950-01456-01 |
| ELEGOO Upgraded Electronics Fun Kit |  |
| Mini Portable USB Night Light |  |
| Woox R4785 smart plug |  |
| Rekavin USB Plug |  |

# Design & Modeling

## User Persona

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

### *Admin*

#### 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

### *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

### *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

### *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

### *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

## 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 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

#### Card Scanning Screen

The Scan Card Component 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

#### Facial Recognition Screen

Once the User details are populated the Facial Detection begins. 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

#### Selecting Device

If the user is successfully authenticated the list of devices they can start will be displayed on the screen 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 screen shot of a computer

Description automatically generated

### *Back-End 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.

#### Users

TBD

#### Devices

TBD

#### Roles

TBD

## 

### *User Case Diagram*

#### Admin

This diagram shows the Admins flow within the Application.

**A screenshot of a computer

Description automatically generated**

#### Cleaner

This diagram shows the Cleaners flow within the Application.

A screenshot of a computer

Description automatically generated

#### Lab Technician’s

This diagram shows the user flow within the Application.

*A screenshot of a computer

Description automatically generated*

## Database Schema

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:** securegate

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.

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\_uid** (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.

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).

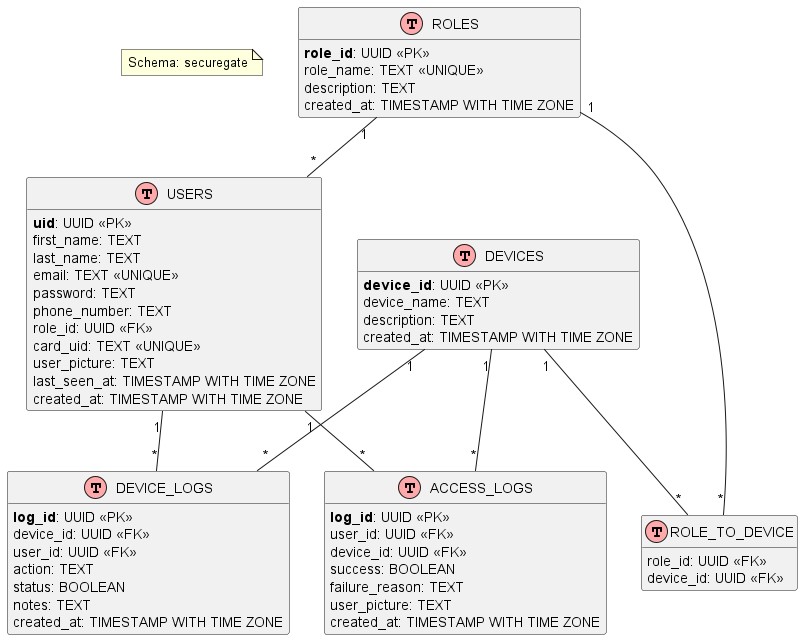
**Table Fields:**

* + **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.

**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

**UML Diagram**



## Physical Model Design

Below you will find the drawings for the physical design for the project. These are subject to change.

### *Electrical Drawing*

A diagram of a machine

Description automatically generated

### *Front View Design*

A close-up of a door

Description automatically generated

### *Side View Design*

A diagram of a door structure

Description automatically generated

# Conclusion + Further work

## Reflection

### What I have learned

TBD

### What I would do differently

TBD

# References

TBD

# Attachments

## Attachment#1:- Electrical Drawing

A diagram of a machine

Description automatically generated

## Attachment#2:-

## Attachment#3:-