



UNIVERSITI KUALA LUMPUR  
MALAYSIAN INSTITUTE OF INFORMATION  
TECHNOLOGY

IIB20804 IOT APPLICATION SECURITY  
OCTOBER 2025

BACHELOR OF INFORMATION TECHNOLOGY  
(HONS) (INTERNET OF THINGS)

**PROJECT CAPSTONE:**  
**Smart Surveillance and Monitoring System**

**PREPARED BY:**

NAME	BCSS/BIOT	STUDENT ID	HP NO.
SHAIFUL HAZIQ HIDZMI BIN SAIPOL AZMI	BIOT	52224123153	<b>0133487527</b>
SHARIFAH NUR QISTINA NABILA BINTI SYED ABD RANI	BIOT	52224123537	<b>0139941443</b>
NUR AISHAH KAMALIAH BINTI MEZLAN	BIOT	52224123034	<b>0192523240</b>
MUHAMMAD UWAIS BIN MOHD IMRAN	BIOT	52224123579	<b>0196042251</b>

**PREPARED FOR:**  
**PROF. DR. SHAHRULNIZA MUSA**

# Table of Contents

<b>1.0 Executive Summary .....</b>	4
<b>2.0 Project Aim and Objectives .....</b>	4
<b>3.0 System Requirements.....</b>	5
<b>3.1 Functional Requirements.....</b>	5
<b>3.2 Security Requirements.....</b>	6
<b>4.0 System Design .....</b>	7
<b>4.1 System Architecture.....</b>	7
<b>4.2 Schematic Diagram - Hardware schematics.....</b>	8
<b>4.3 System Flow .....</b>	10
<b>4.4 Project Components.....</b>	11
<b>4.4.1 Hardware Components .....</b>	12
<b>4.4.2 Software Components .....</b>	13
<b>4.5 Platforms and Services Used .....</b>	14
<b>5.0 Security Design.....</b>	15
<b>5.1 Authentication, Access Control, and Network Isolation .....</b>	15
<b>5.2 Threat Modelling (STRIDE Analysis) .....</b>	16
<b>1. Implementation .....</b>	17
<b>a. Configuration Setup .....</b>	17
<b>b. Source Code Overview.....</b>	17
<b>c. Hardware / Simulation Setup .....</b>	18
<b>d. Middleware setup .....</b>	18
<b>OneM2M (Mobius) setup.....</b>	18
<b>Node-RED setup.....</b>	20
<b>e. Provide link to GitHub .....</b>	21
<b>2. Testing and Evaluation .....</b>	21
<b>a. Test cases, methodologies, and tools used.....</b>	21
To ensure the system met both functional and security requirements, a structured testing approach was adopted, focusing on hardware functionality, middleware integration and security controls.....	21
<b>Test Environment:.....</b>	21
• Wi-Fi network with static IP addressing .....	21
• ESP32, ESP32-CAM, HC-SR04 ultrasonic sensor and LED .....	21
• Node-RED and Mobius oneM2M CSE running locally .....	22
• Postman for API validation.....	22
• Google Firebase for identity verification .....	22
<b>Test Cases:.....</b>	22
<b>b. Functionality, and security testing results.....</b>	22
<b>Functional Testing Results: .....</b>	22
<b>Discussion.....</b>	23

<b>c.</b>	<b>Analysis of results.....</b>	<b>23</b>
The system successfully demonstrated a working IoT based surveillance prototype with embedded security controls. Key findings include: .....		23
•	Edge Processing Efficiency: The ESP32 effectively performed threshold-based filtering, reducing false alarms before data transmission. ....	23
•	Low Latency Monitoring: Local network deployment ensured real-time sensor alerts and video streaming with minimal delay. ....	23
•	Security by Design: The layered security model, Firebase authentication, admin approval gating and network isolation that proved effective in preventing unauthorized access. ....	23
•	Standard Compliance: Using oneM2M (Mobius) enabled structured data storage and interoperability, though reliance on polling introduces minor latency. ....	23
<b>d.</b>	<b>Challenges faced and how they were addressed.....</b>	<b>23</b>
<b>Conclusion and Future Work .....</b>		<b>24</b>
<b>e.</b>	<b>Summary of achievements. ....</b>	<b>24</b>
<b>f.</b>	<b>Limitations and potential improvements. ....</b>	<b>24</b>
<b>g.</b>	<b>Suggestions for future improvement. ....</b>	<b>25</b>

# IoT Security Capstone Project: Smart Surveillance and Monitoring System.

## 1.0 Executive Summary

Technology is developing very rapidly in today's world. As human capabilities continue to grow, many new innovations and ideas are being introduced. Advanced tools and systems help make daily tasks easier and allow activities to be completed more efficiently. In addition, the use of smart technology can improve safety and security in everyday life.

Criminal cases such as home invasions and jewelry theft have become increasingly concerning every person around the world. As a result, security technologies such as CCTV systems and motion detection have been widely adopted to help protect homes and properties. These technologies allow users to monitor their surroundings and respond quickly to suspicious activities.

However, as technology becomes more advanced, criminals are also finding new ways to exploit it. Cybercrimes such as hacking and unauthorized system access are becoming more common. This shows that while technology brings many benefits, it also introduces new security challenges. Therefore, security systems must be continuously upgraded and properly maintained to ensure the safety, privacy, and well-being of users.

This is the main purpose of the Smart Surveillance and Monitoring System project. IoT-based surveillance systems are often exposed to serious security risks such as unauthorized camera access, data leakage, weak authentication, and insecure device firmware. Compromised surveillance systems may lead to privacy violations, system misuse, or even physical security threats. Therefore, security must be a core design consideration rather than an afterthought in IoT surveillance solutions. By utilizing Internet of Things (IoT) technology, this project aims to enhance the security of residential areas and private properties through continuous monitoring. Many people are busy during the day and feel tired at night, making it difficult to always be alert to their surroundings. Therefore, this system is designed to assist users by providing real-time monitoring and improving overall safety, ensuring that their living spaces are securely always guarded safely.

## 2.0 Project Aim and Objectives

The objectives of this project are as follows:

- To design and develop a smart surveillance system using IoT technology.
- To monitor residential or private areas in real time using CCTV and motion captured.
- To improve security by detecting suspicious activities and providing timely alerts to users.
- To reduce the need for constant manual monitoring and increase user convenience.

## **3.0 System Requirements**

### **3.1 Functional Requirements**

The system shall fulfill the following functional requirements:

- Live Video Monitoring

All linked camera modules will deliver real-time Closed-Circuit Television (CCTV) video streaming to an authorized user interface (such as a dashboard or mobile application).

- Motion Detection

The system will:

1. Use server-side or embedded motion detection algorithms to identify movement within a camera's field of view.
2. Trigger for an event such as starting to record or sending an alarm when motion is detected.

- Automated Alerts and Notifications

When the system detects suspicious motion or specified activity, it will instantly notify selected users via message, the main dashboard, or a mobile application.

- Secure Access Control

Only authorized individuals will be able to access the surveillance dashboard and associated features.

- User Management

Implement role-based access control (BACC) providing distinct access levels, such as Administrator, Operator, and Viewer.

Permit the Administrator role to add, modify, or revoke user accounts and their associated access privileges.

- System Status Monitoring

Monitor every component's health (e.g., camera connectivity status—online/offline, storage availability, network connection quality).

Users will be informed if a vital device reports a fault or disconnects.

- Activity Logs

The system must keep thorough audit records of all noteworthy user actions, such as successful and unsuccessful login attempts, system access history, and configuration modifications.

## **3.2 Security Requirements**

- Authentication and Authorization

Implement an authentication measure in place. Enforce Role-Based Access Control (RBAC) to precisely limit and define user privileges based on their assigned role.

- Privacy Protection

Limit access to private videos according to user roles and specific permissions.

Ensure that all applicable data privacy regulations are followed, particularly the Personal Data Protection Act (PDPA) standards for operations in Malaysia.

- Intrusion Detection and Logging

Identify and record any abnormal network activity or attempts at illegal access.

Maintain secure, unchangeable audit logs for regulatory auditing and forensic analysis.

- Firmware and Software Integrity

Ensure secure over-the-air (OTA) firmware updates are supported by all connected devices, such as CCTV cameras, gateways, and processing units.

Verify the legitimacy before installing any firmware or software upgrades by using digital signatures.

- Physical Security

Include design elements that prevent illegal or physical tampering with camera hardware.

Require all vital storage and surveillance devices to be installed in secure areas with limited physical access.

## 4.0 System Design

The designed system incorporates a robust multi-tier Internet of Things (IoT) architecture that ensures the secure flow of data, real-time monitoring capabilities, and adequate management of the devices. The IoT multi-tier architecture is standards-compliant within oneM2M standards and consists of three main tiers that include the User Interface Layer, the Cloud Middleware Layer, and the Edge Device Layer. Through the incorporation of HTTPPs and MQTT protocols along with hardware-level security measures, the surveillance data is transmitted over the environment without the risk of intercepting it.

### 4.1 System Architecture

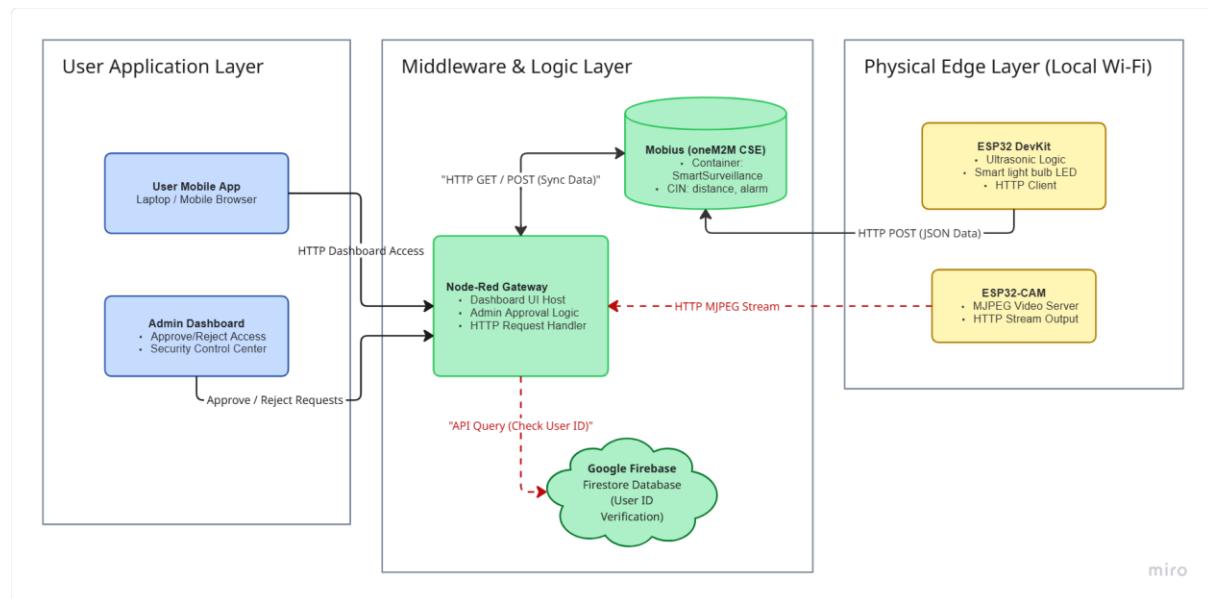


Figure 4.1: Architecture Diagram

Figure 4.1 above highlights the logical architectural design of the Smart Surveillance System proposed. Contrary to conventional designs, which focus on cloud-based systems, the proposed system is built using the Local IoT Gateway model, ensuring latency and networking isolation.

The architecture is divided into three primary layers:

#### 1. Physical Edge Layer (Local Wi-Fi):

- The ESP32 DevKit serves as the main sensor node, which performs edge computations for distance thresholding before transferring light JSON messages via the HTTP POST method to the middleware system.
- The ESP32-CAM acts as a standalone video server, streaming Motion-JPEG (MJPEG) video to the gateway through a dedicated HTTP connection in order to alleviate data saturation.

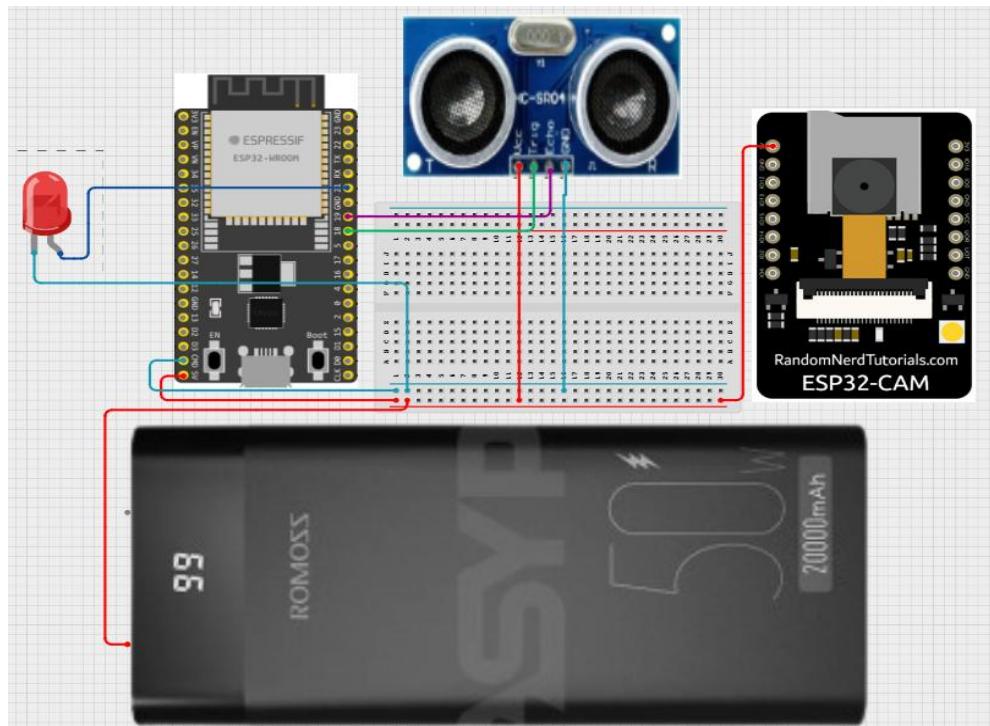
## 2. Middleware & Logic Layer:

- a. Node-RED serves as the central gateway and application logic controller. It aggregates data from the Mobius platform and manages the user dashboard interface.
  - b. Mobius (oneM2M CSE) acts as the standardized database, storing sensor logs in a hierarchical tree structure (AE/Container/ContentInstance).
  - c. Google Firebase is utilized specifically for Identity Verification. When a user requests access, Node-RED queries about this cloud database to validate the User ID before unlocking the local video stream.

### 3. User Application Layer:

- a. The system provides two distinct interfaces: a User Dashboard for monitoring and an Admin Panel for security control. Access to the video feed is logically restricted until the Administrator explicitly approves the request via the dashboard logic.

#### **4.2 Schematic Diagram - Hardware schematics.**



*Figure 4.2: Circuit Diagram*

Figure 4.2 shows the hardware connections of the edge monitoring node. The circuit is built around the microcontroller named ESP32 DevKit V1. It acts as the central unit for the acquisition of data from sensors and the management of alerts.

The connections be established as follows:

- a. Ultrasonic Sensor (HC-SR04): The power source for this module is connected to the VIN (5V) pin of the ESP32 microcontroller. Additionally, the Trigger Pin is connected to GPIO Pin 18 to generate distance pulses, while the Echo Pin is connected to GPIO Pin 19 to receive the echo signal duration.
- b. Visual Alarm (Red LED): For local feedback, a red LED is wired to GPIO pin 21. The red LED is a status indication LED that flashes in an "ambulance pattern" if the range limit has been exceeded.
- c. Power Supply: The whole system is powered through the micro-USB port of ESP32, which provides a controlled voltage supply for the microcontroller as well as the peripheral circuits.

Compared with the sensor node, the ESP32-CAM has the capability to act as a self-contained wireless module. Its hardware components are limited in order to provide the node with the flexibility of placement sites:

- a. Power Interface: This module is provided power through the 5V and GND pins. These are connected to the stable USB power.
- b. Data Interface: There are no direct interfaces for data TX/RX to the main controller. Instead, this module uses 2.4GHz Wi-Fi Antenna present in this module for connection to a network.
- c. Logical Connection: The video signal is connected logically to the IP Address provided by the local router, bypassing the necessity of wiring the signal of the camera to the dashboard.

## 4.3 System Flow

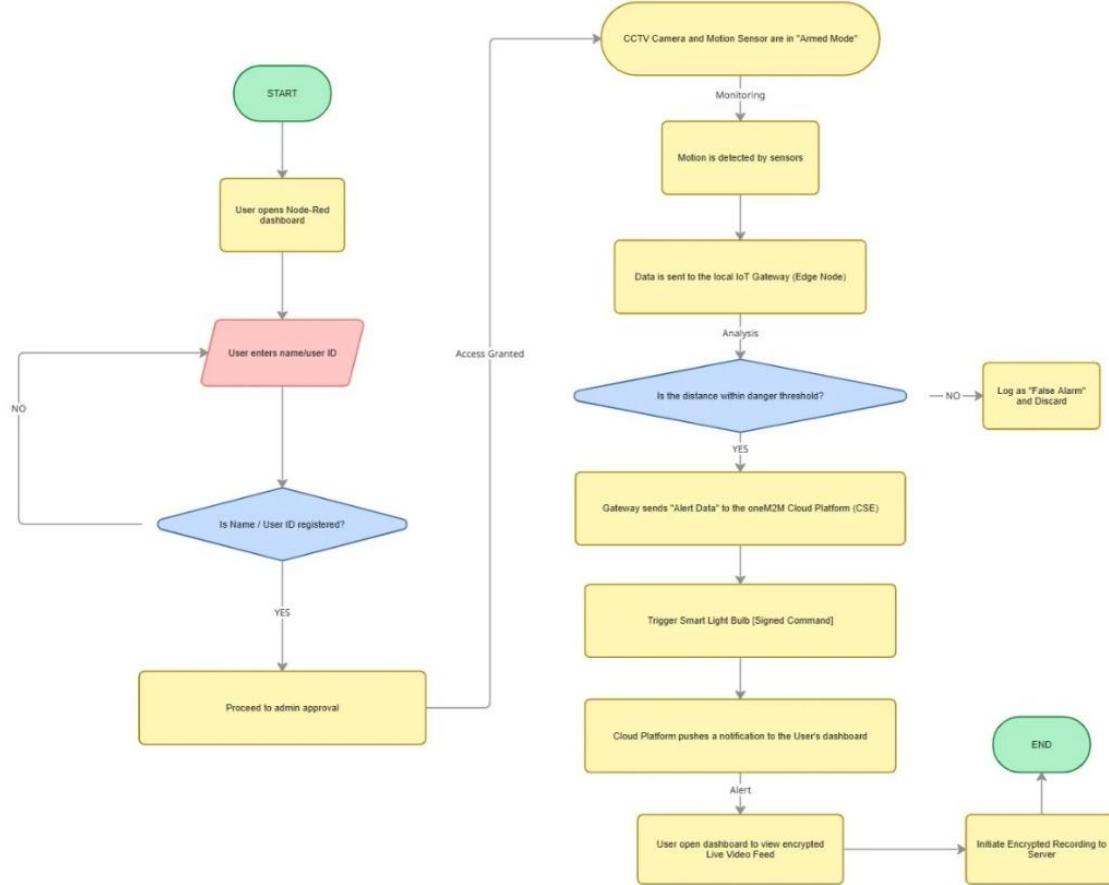


Figure 4.3: Flowchart Diagram

The system workflow, illustrated in Figure 3.4, details the operational logic from user authentication to intrusion detection and automated response. The process is divided into two primary stages: Access Control and Active Monitoring.

### Phase 1: User Authentication & Access Control

The operation begins with the user accessing the app (in this project, Node-RED Dashboard is used for demonstration). To ensure system security, a verification process is strictly enforced:

- a. Input: User enters their unique Name.
- b. Verification (Decision Block): The system cross-references the input with the registered database.
  - If Invalid: Access is denied, and the user is looped back to re-enter credentials.
  - If Valid: The request is forwarded for Admin Approval.
- c. Access Granted: Once approved, the system transitions into the "Armed Mode," enabling the physical security hardware.

### *Phase 2: Monitoring & Threat Analysis*

In the Armed Mode, the ESP32-CAM and ultrasonic sensors continuously monitor the environment. The data processing flow follows these steps:

- a. Event Detection: When motion is detected by the sensors, the raw data is immediately transmitted to the local IoT Gateway (Edge Node).
- b. Threshold Analysis (Edge Computing): The gateway performs a logic check to determine if the detected distance falls within the pre-set "Danger Threshold" (an intruder is too close).
  - False Alarm: If the distance is outside the threshold, the event is logged as a "False Alarm" and discarded to prevent spamming the user.
  - Confirmed Threat: If the distance is within the danger zone, the system initiates the alert protocol.

### *Phase 3: Automated Response & Alerting*

Upon confirming a threat, the system executes three parallel actions:

- a. Cloud Sync: The gateway transmits the "Alert Data" to the oneM2M Cloud Platform (CSE) for permanent record keeping.
- b. Physical Deterrence: A Signed Command is sent to trigger the Smart Light Bulb, providing immediate visual deterrence.
- c. User Notification: The cloud platform pushes a real-time alert to the user's dashboard.

### *Phase 4: User Verification*

The workflow concludes with the user's response to the alert:

- a. The user opens the app dashboard to view the Live Video Feed to visually verify the situation.

## **4.4 Project Components**

In this section, the specific technical requirements of the hardware and software elements used in facilitating the design of the IoT security solution are described in depth. These elements are organized based on two major categories: Hardware Components, which involve the physical implementation of the sensing elements utilizing microcontrollers that record the data, and Software Tools, which refer to the development software, cloud software, as well as the dashboard software. These software elements were considered based on their compatibility with the oneM2M standards and the ease of integration in the Node-RED platform.

#### 4.4.1 Hardware Components

##### a. ESP32 Devkit



A high-performance, dual-core microcontroller. It acts as the primary edge device, processing sensor data locally to detect threats and triggering the physical alarm system.

##### b. ESP32-CAM



A Wi-Fi enabled microcontroller with an integrated camera module. It functions as the visual monitoring node, capturing real-time video and streaming it securely to the dashboard via HTTP.

##### c. HC-SR04 Ultrasonic Sensor



A non-contact distance measurement module. It continuously monitors the environment and detects motion when an object breaches the safety threshold distance.

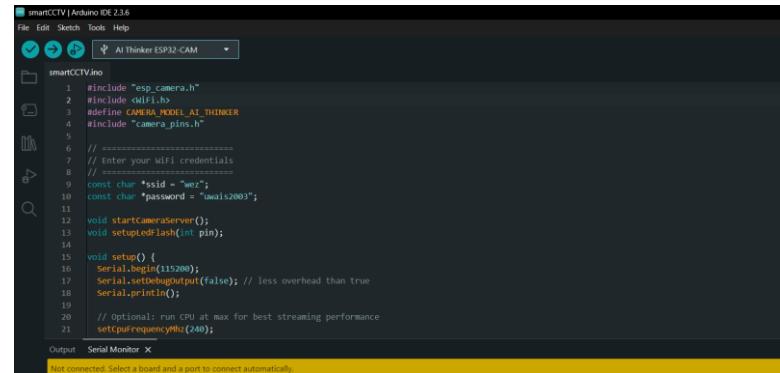
##### d. Red LED



Physical actuators used for immediate local feedback. It provides visual warnings to deter intruders the moment motion is detected.

## 4.4.2 Software Components

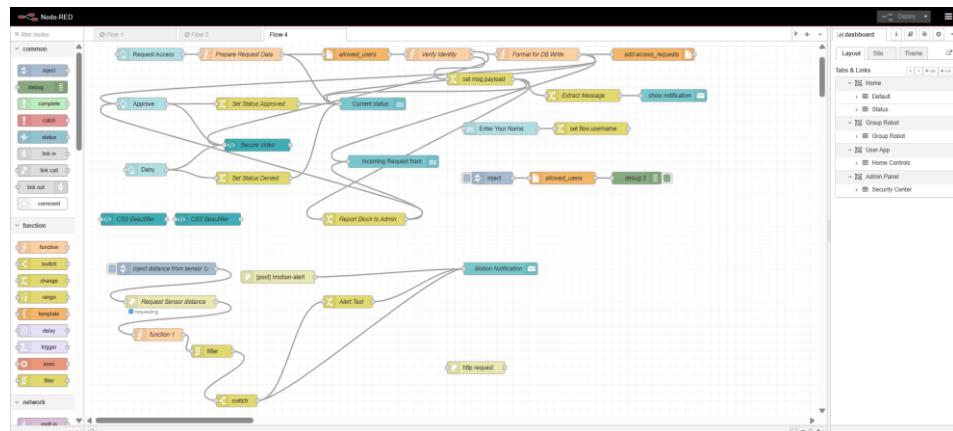
### a. Arduino IDE



```
smartCCTV | Arduino IDE 2.3.6
File Edit Sketch Tools Help
AI Thinker ESP32-CAM
smartCCTV.ino
1 #include <esp_camera.h>
2 #include <WiFi.h>
3 #define CAMERA_MODEL_AI_THINKER
4 #include "camera_pins.h"
5
6 // =====
7 // Enter your WiFi credentials
8 // =====
9 const char *ssid = "wz";
10 const char *password = "wais2003";
11
12 void startcameraserver();
13 void setupOLEDflash(int pin);
14
15 void setup() {
16   Serial.begin(115200);
17   Serial.setDebugOutput(false); // less overhead than true
18   Serial.println();
19
20   // Optional: run CPU at max for best streaming performance
21   setCoreFrequencyMHz(240);
}
Output Serial Monitor
Not connected. Select a board and a port to connect automatically.
```

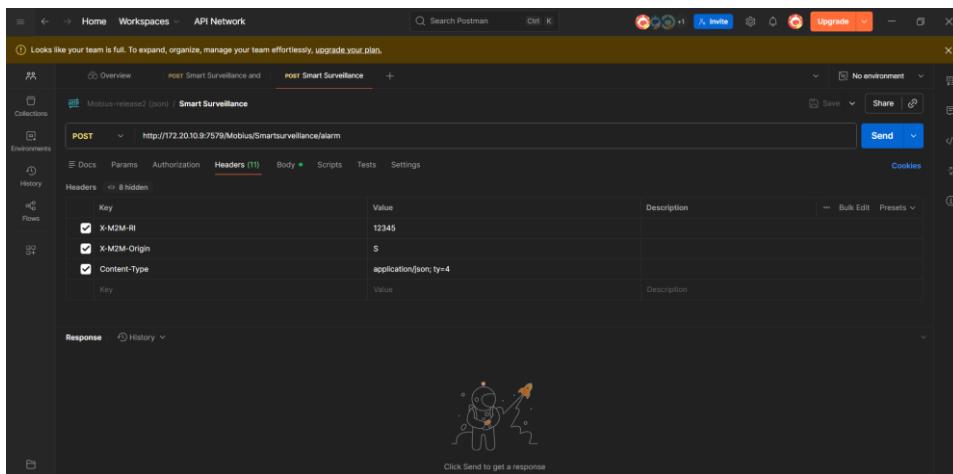
The primary integrated development environment used to write, compile, and upload firmware to the ESP32 microcontrollers. It manages the necessary libraries for the camera module and sensor integration.

### b. Node-RED



A flow-based visual programming tool. It serves as the system's logic gateway, integrating hardware signals, managing database checks, and hosting the user dashboard.

### c. Postman



A screenshot of the Postman application. The main window shows a 'POST' request to 'http://172.20.10.9:7578/Mobius/Smartsurveillance/alarm'. The 'Headers' tab is selected, showing the following headers:

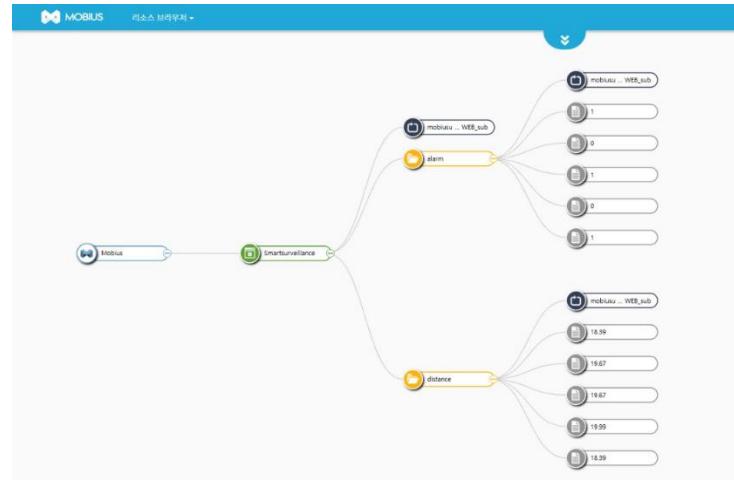
Key	Value	Description
X-M2M-RI	12345	
X-M2M-Origin	S	
Content-Type	application/json; ty=4	

The 'Body' tab is also visible. Below the request, there's a response placeholder with a cartoon character and the text 'Click Send to get a response'.

A comprehensive API testing tool used to verify the communication with the Mobius server. It allows for the manual sending of HTTP GET/POST requests to ensure the oneM2M data structure (Containers and ContentInstances) is functioning correctly before integrating with the hardware.

## 4.5 Platforms and Services Used

### a. Mobius (oneM2M CSE)



An open-source IoT server platform. It functions as the central data repository, organizing all sensor logs into a standardized hierarchy (Containers and ContentInstances) for retrieval.

### b. Firebase Authentication

The screenshot shows the Firebase Database interface for a project named "SmartSurveillance". The left sidebar shows various project settings like Project Overview, Realtime Database, and Firestore Database. The main area displays the "access\_requests" collection under the "Database" tab. It shows a list of documents with their IDs and some metadata. One document is expanded to show its fields: "status: "Waiting Approval From Admin\"", "time: "Tue Jan 13 2026 13:30:05 GMT+0800 (Singapore Standard Time)", and "user: "ali"". There is also a "allowed\_users" collection listed below it.

A backend identity management service. It secures the system by verifying user credentials (username) before granting access to the live video feed or control panel.

## 5.0 Security Design

### 5.1 Authentication, Access Control, and Network Isolation

Given the system's operation takes place in a resource-limited local area networking environment, the design of the system's security infrastructure moves its emphasis away from Transport Encryption (like TLS) and toward solid Application-Level Logic and Network Isolation techniques instead. The design and implementation of its overall infrastructure have been carried out using the "Defense-in-Depth" methodology and incorporating three specific layers:

1. Identity Verification (Cloud-Based Authentication): Although a lightweight HTTP is used for communication within the device, User identity is authenticated with a secure third-party authority.
  - a. Implementation: The system uses Google Firebase as its Identity Provider (IdP). When a user tries to login and access the Node-RED dashboard, their entered User ID is not validated or processed by the system but is transmitted as a query parameter to the Firebase Firestore database.
  - b. Security Logic: The system strictly follows an "Allowlist" policy. If the User ID does not appear in the cloud database, the "Request Access" logic in Node-RED will be completely disabled and will not interact with system logic whatsoever.
2. Human-in-the-Loop Access Control (Logical Gating): To mitigate the risk of compromised credentials, the system implements a Two-Step Authorization workflow.
  - a. Implementation: A valid User ID does not grant immediate access to the video feed. Instead, it transitions the user session to a "Pending" state.
  - b. Access Control Policy: The video stream is logically blocked (hidden behind a UI overlay) until a secondary actor (the Administrator) explicitly approves the request via a separate Admin Panel. This ensures that no video data is rendered to the client without real-time human verification.
3. Network Perimeter Isolation: The primary defense against external remote attacks is the deployment environment itself.
  - a. Implementation: The entire IoT ecosystem (ESP32 Nodes, Camera, and Broker) operates on a Local Area Network (LAN) with static IP addressing.
  - b. Defense: By not exposing port forwarding to the public internet, the system is inherently protected from external botnets and remote exploits. Access is physically restricted to users within the Wi-Fi coverage area.

## 5.2 Threat Modelling (STRIDE Analysis)

To evaluate the system's resilience, a STRIDE analysis was conducted. This method categorizes potential threats and identifies the specific logical mitigations implemented in this project.

*Table 5.1: Stride Analysis*

Threat Category	Potential Risk	Implemented Mitigation
S - Spoofing	An attacker impersonates a valid user to view the surveillance feed.	Cloud Validation: The system queries Firebase to verify if the ID exists. Unknown IDs are rejected instantly.
T - Tampering	An attacker injects false distance data (for example, "Safe User") to bypass the alarm.	Input Validation: Node-RED flows are configured to parse specific JSON structures. Malformed packets or direct injections that don't match the oneM2M schema are discarded.
R - Repudiation	A user denies accessing the system or triggering an alert.	Audit Logging: Every access request and sensor trigger is stored as a "ContentInstance" (CIN) in the Mobius oneM2M server, creating a permanent, timestamped history of events that cannot be locally deleted by the user.
I - Information Disclosure State-Based	Unauthorized viewing of the private home video stream.	UI Locking: The video element in the dashboard is programmatically hidden by default. The image source URL is only injected into the DOM after the specific logic flow receives the Admin_Approved signal.
D - Denial of Service	Flooding the dashboard with requests to crash the system.	Network Bandwidth Limits: Operating on a standard Local Wi-Fi network limits the traffic volume. Since the system is not public, massive Distributed Denial of Service (DDoS) attacks are not possible from external sources.
E - Elevation of Privilege	A standard user trying to access the Admin Panel controls.	Role Separation: The "Admin Panel" and "User App" are deployed on separate dashboard URLs (or UI Groups). The Admin controls are not rendered on the User's interface, preventing them from approving their own requests.

## 1. Implementation

### a. Configuration Setup

To facilitate communication between IoT devices, middleware, and the application dashboard, this project was implemented in a local network environment.

The development environment consists of the following components:

- **Hardware environment:** LED, ESP32-CAM, ESP32 microcontroller, and an ultrasonic sensor.
- **Middleware environment:** Node-RED and oneM2M platform (MOBIUS).
- **Testing tools:** Postman for REST API testing.
- **Network Setup:** To guarantee dependable communication between devices and servers, Local Area Network (LAN) with static IP addressing was used.

The Arduino IDE was used to program the ESP32 devices, and the necessary libraries for Wi-Fi connectivity and sensor handling were added. On a local PC, Node-RED was set up to use HTTP requests to reach the MOBIUS platform. For oneM2M was set up as the Common Service Entity (CSE) and distributed locally.

### b. Source Code Overview

IoT device code, middleware logic, and application dashboard logic are the three primary parts of the system source code.

#### ESP32 Code

The ESP32 source code handles:

- Ultrasonic measuring of distance.
- Threshold-driven decision-making.
- LED alert activation (ambulance style blinking).
- Sensor and alarm data transmission to the oneM2M platform.

The ESP32-CAM code handles:

- Setting up the camera module.
- Putting on a live video broadcast.
- Limiting access until the application later grants permission.

#### Node-RED flow logic

- Handle sensor data that comes in from oneM2M.
- Control user access requests.
- Manage the reasoning for admin approval and refusal.
- Regulate the state of CCTV access.

- Show the dashboard's warnings and system status.

(NANTI AKU LETAK LINK REPO GITHUB)

### c. Hardware / Simulation Setup

An ESP32 microcontroller connected to LED, and an ultrasonic sensor makes up the physical hardware configuration. The distance between the gadget and surrounding objects is continuously measured by the ultrasonic sensor. The ESP32 activates the LED and sends an alarm event to the oneM2M platform when the measured distance drops below a predetermined danger threshold.

A separate ESP32-CAM module is used as a CCTV system. Once permission is given, it offers a live video stream that may be viewed via the application dashboard. To lower storage costs and privacy concerns, the camera offers real-time monitoring instead of storing video data.

To provide minimal latency and dependable connection, every hardware component runs on the same local network.

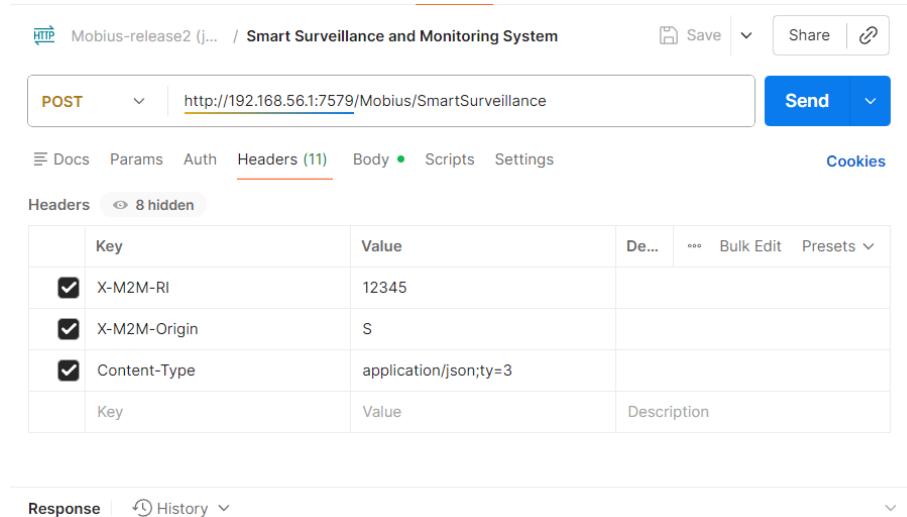
### d. Middleware setup

#### OneM2M (Mobius) setup

The oneM2M Common Services Entity (CSE) for centralized data management is MÖBIUS. The oneM2M resource structure is used by the platform to store IoT data:

- Application Entity (AE): SmartSurveillance.
- Containers: alert, distance
- ContentInstances: Alert values and specific sensor data.

RESTful HTTP POST requests are used to provide sensor data and alert status to MÖBIUS. The /la (latest) resource is used to retrieve the most recent data, providing effective access for real-time monitoring.

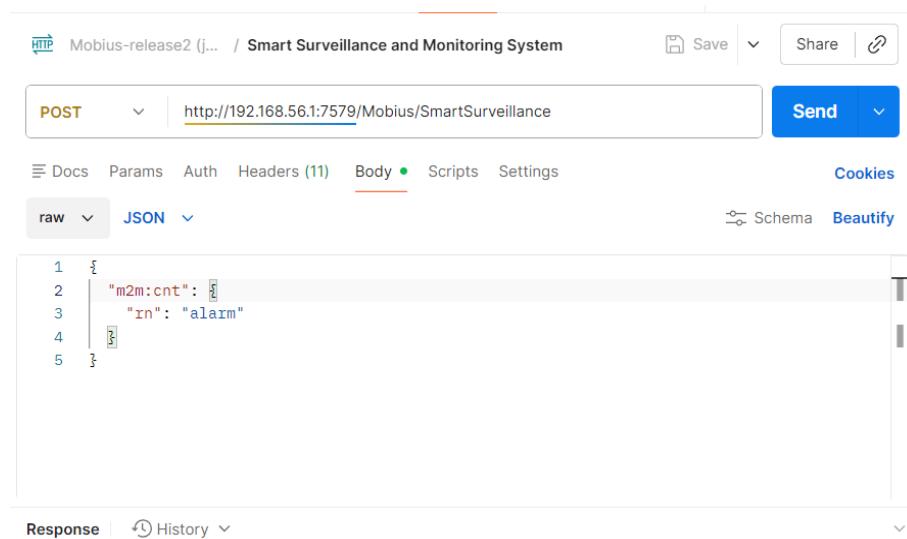


The screenshot shows the Postman application interface. At the top, there's a header bar with 'HTTP' and 'Mobius-release2 (j... / Smart Surveillance and Monitoring System'. To the right are 'Save', 'Share', and a copy icon. Below the header, a search bar shows 'POST http://192.168.56.1:7579/Mobius/SmartSurveillance' with a 'Send' button. Underneath the search bar, tabs include 'Docs', 'Params', 'Auth', 'Headers (11)', 'Body', 'Scripts', and 'Settings'. The 'Headers (11)' tab is selected. A table below lists the headers:

Key	Value	Description
X-M2M-RI	12345	
X-M2M-Origin	S	
Content-Type	application/json;ty=3	
Key	Value	Description

At the bottom, there are 'Response' and 'History' buttons.

Figure 1: Postman headers



The screenshot shows the Postman application interface. At the top, there's a header bar with 'HTTP' and 'Mobius-release2 (j... / Smart Surveillance and Monitoring System'. To the right are 'Save', 'Share', and a copy icon. Below the header, a search bar shows 'POST http://192.168.56.1:7579/Mobius/SmartSurveillance' with a 'Send' button. Underneath the search bar, tabs include 'Docs', 'Params', 'Auth', 'Headers (11)', 'Body', 'Scripts', and 'Settings'. The 'Body' tab is selected. A dropdown menu shows 'raw' and 'JSON' with 'JSON' being selected. Below the dropdown, there's a code editor area containing JSON code:

```
1 {  
2   "m2m:cnt": {  
3     "in": "alarm"  
4   }  
5 }
```

At the bottom, there are 'Response' and 'History' buttons.

Figure 2: Postman Body (JSON)

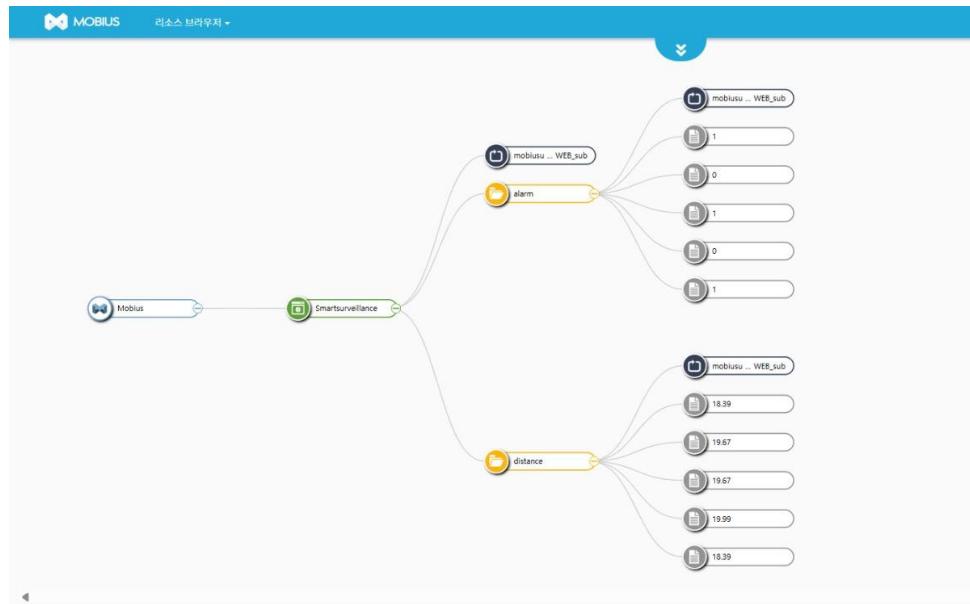


Figure 3: MOBIUS data management.

## Node-RED setup

Node-RED serves as both the dashboard interface and the application layer. It handles access requests, enforces security regulations, and retrieves sensor and alert data from MÖBIUS.

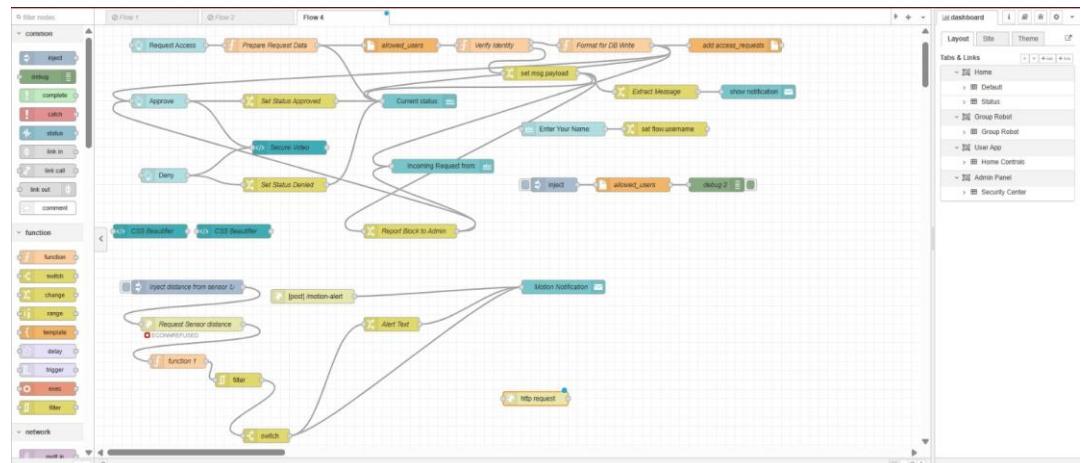


Figure 4: Node-RED Flow for Access Control, Admin Approval, and Alert Handling

There were two dashboards put in place:

Admin panel: Approve or reject access requests using the admin panel.

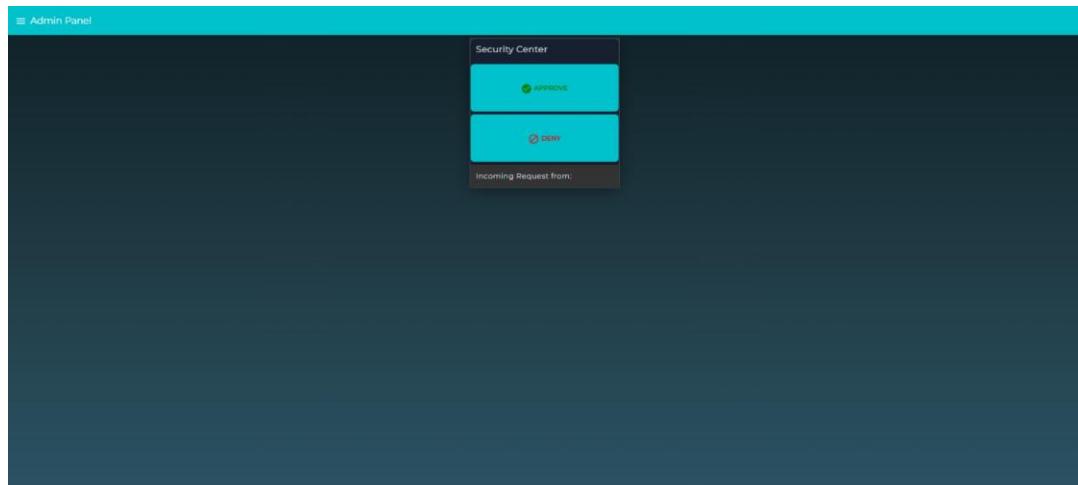


Figure 5: Admin Panel

User dashboard: Examine system status and request access through the User Dashboard.

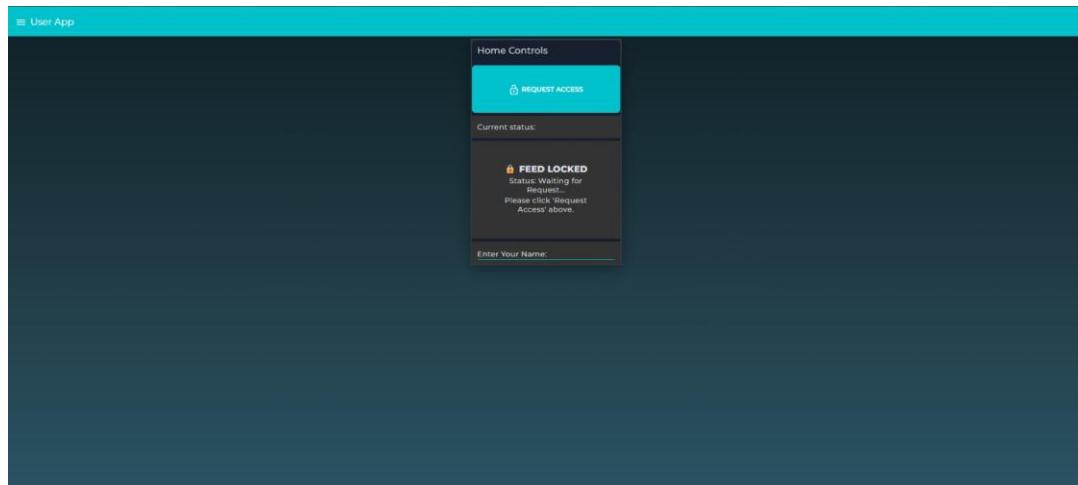


Figure 6: User dashboard

Node-RED offers an extra layer of application-level access security by guaranteeing that the CCTV stream stays locked until admin consent is given.

#### e. Provide link to GitHub

## 2. Testing and Evaluation

### a. Test cases, methodologies, and tools used.

To ensure the system met both functional and security requirements, a structured testing approach was adopted, focusing on hardware functionality, middleware integration and security controls.

Test Environment:

- Wi-Fi network with static IP addressing
- ESP32, ESP32-CAM, HC-SR04 ultrasonic sensor and LED

- Node-RED and Mobius oneM2M CSE running locally
- Postman for API validation
- Google Firebase for identity verification

**Test Cases:**

Test ID	Test Description	Method	Expected Outcomes
TC-01	Ultrasonic sensor distance detection	Place object within defined threshold	LED flashes and alert sent to Mobius
TC-02	ESP32-CAM video streaming	Access dashboard after admin approval	Live video feed displayed without latency
TC-03	User authentication via Firebase	Enter valid or invalid User ID	Valid ID – pending state, Invalid ID – access denied
TC-04	Admin approval workflow	Admin approves/rejects access request	Approved – video unlocked Rejected – access blocked
TC-05	Data persistence in Mobius	Send sensor data via HTTP POST	Data stored as ContentInstance under /SmartSurveillance/container
TC-06	Unauthorized access attempt	Try to access video without approval	Video element remains hidden
TC-07	Network isolation validation	Attempt external HTTP request to local IP	Request fails (no port forwarding)

**Tools Used:**

- Postman: For validating oneM2M REST API endpoints
- 
- 
- 

**b. Functionality, and security testing results.**

**Functional Testing Results:**

Component	Result	Status
Ultrasonic Sensor	Accurately detected objects within 50cm threshold	Pass
ESP32-CAM Stream	Stable MJPEG stream at 640x480 resolution	Pass
Node-RED Dashboard	Real-time updates of sensor data and alerts	Pass

Admin Approval Flow	Access granted only after admin intervention	Pass
Mobius Data Storage	All sensor events stored and retrievable via resource	Pass

### Security Testing Results:

Security Aspect	Test	Result
Authentication Bypass	Attempt to inject false User ID	Blocked by Firebase validation
Unauthorized Video Access	Direct URL access to video stream before approval	UI overlay blocked video rendering
Data Tampering	Send malformed JSON to Mobius	Rejected by oneM2M schema validation
Denial-of-Service Resilience	Simulate repeated access requests	Local network bandwidth limited impact, no system crash
Audit Logging	Verify Mobius logs after sensor triggers and user actions	All events timestamped and stored immutable

## Discussion

### c. Analysis of results.

The system successfully demonstrated a working IoT based surveillance prototype with embedded security controls. Key findings include:

- Edge Processing Efficiency: The ESP32 effectively performed threshold-based filtering, reducing false alarms before data transmission.
- Low Latency Monitoring: Local network deployment ensured real-time sensor alerts and video streaming with minimal delay.
- Security by Design: The layered security model, Firebase authentication, admin approval gating and network isolation that proved effective in preventing unauthorized access.
- Standard Compliance: Using oneM2M (Mobius) enabled structured data storage and interoperability, though reliance on polling introduces minor latency.

### d. Challenges faced and how they were addressed.

Challenge	Solution Implemented
ESP32-CAM video streaming latency	Used MJPEG over HTTP with local Wi-Fi to avoid cloud streaming
User authentication without heavy crypto	Leverage Firebase as lightweight cloud IDP, kept logic off the device
Ensuring video privacy	Implemented UI-level video hiding until admin approval
Data consistency in Mobius	Use resource for latest data retrieval, structured containers for event logging
Integrating multiple protocols	Unified under HTTP REST for sensor data and video streaming
Admin approval workflow complexity	Designed separate Node-RED dashboards for user and admin with clear UI controls

## Conclusion and Future Work

### e. Summary of achievements.

This project effectively used IoT technologies and the oneM2M standard to create and execute a secure smart surveillance and monitoring system. The system incorporates ESP32-based devices, such as an ESP32-CAM for real-time video monitoring, an LED alert system, and an ultrasonic sensor.

Sensor readings and alert events were stored using standard oneM2M resources on the oneM2M platform (MÖBIUS), which served as a centralized data management layer. To provide a dashboard interface, handle user access requests, enforce admin approval, and provide system status in real time, Node-RED was created as the application layer.

Several levels of security were taken into account, such as centralized data management, threshold-based event detection at the edge device, and access restriction through admin clearance. The study shows how IoT security concepts can be used in a smart surveillance setting.

### f. Limitations and potential improvements.

The current system was found to have a number of shortcomings despite its effective adoption. As the system grows, its reliance on polling to obtain the most recent data from the oneM2M platform may cause delays and lower efficiency.

For bigger installations with numerous users, the name-based access control technique that currently requires human admin approval might

not be appropriate. Furthermore, the ESP32-CAM does not provide secure video storage or sophisticated video analytics; it simply offers live video streaming.

Due to time and resource limitations, network security capabilities like complete HTTPS/TLS configuration and certificate management were not fully incorporated in the design.

**g. Suggestions for future improvement.**

Future work can consider a number of improvements. Polling can be replaced with oneM2M subscription and notification systems to enhance the system and enable real-time event updates with reduced overhead.

To improve security, stronger authentication methods like token-based authentication or role-based access control can be incorporated. Secure video recording, encryption, and fundamental video analytics like motion categorization can all be added to the CCTV component.

The system's functionality might be increased by integrating more sensors and actuators, and scalability and availability could be enhanced by cloud deployment. The system's performance, security, and usability would all be strengthened by these enhancements.