SafeOn

An Anomaly Detection System for Smart Home IoT Security

Jaemin Jeong
Dept. of Information Systems
Hanyang University
Seoul, Republic of Korea
woals5633@hanyang.ac.kr

Juseong Jeon
Dept. of Information Systems
Hanyang University
Seoul, Republic of Korea
hyu22ix@hanyang.ac.kr

Seungmin Son
Dept. of Information Systems
Hanyang University
Seoul, Republic of Korea
andyson0205@gmail.com

Wonyoung Shin
Dept. of Information Systems
Hanyang University
Seoul, Republic of Korea
pingu090@hanyang.ac.kr

***Abstract*—The Internet of Things (IoT) has rapidly expanded into home environments through devices such as IP cameras, smart locks, and household appliances. While these devices greatly enhance convenience, they simultaneously introduce broad attack surfaces that threaten user privacy and safety. Traditional security mechanisms often fail to detect subtle or previously unseen threats that manifest as abnormal network behavior rather than identifiable vulnerabilities. In this work, we present a smart home security framework adopting an Endpoint Detection and Response (EDR) approach tailored for IoT environments. The system integrates multimodal telemetry with rule-based detection and machine learning techniques—specifically Isolation Forest for real-time anomaly scoring and Random Forest for classification of suspicious flow patterns. This dual-stage detection pipeline enables fast, lightweight, and accurate identification of abnormal device activity suitable for resource-constrained edge environments. The system provides real-time alerts, captures detailed forensic logs, and preserves user privacy by performing analysis at the network edge without requiring access to device internals. Experimental evaluations demonstrate the framework's ability to identify both known and previously unseen anomalies in diverse IoT traffic patterns, maintaining high detection performance while minimizing false positives. Overall, this work offers a practical, scalable, and privacy-conscious security solution for protecting everyday smart home IoT devices against increasingly sophisticated network-based threats.**

***Index Terms*—Smart Home Security, IoT, Endpoint Detection and Response (EDR), IsolationForest, RandomForest, anomaly detection, privacy-preserving.**

Table I: Role Assignments

Roles	Name	Task Description
Project Manager, Security Developer	Juseong Jeon	Responsible for detecting, analyzing, and responding to anomalies in real time.approach Collaborates with AI/ML and Software Developers to embed security logic into the EDR pipeline and ensures secure communication between edge and cloud components. Supervises system integration, testing, and deployment, and maintains comprehensive project documentation.
User/Customer, Software Developer	Jaemin Jeong	Focuses on enhancing smart home IoT experience and interaction. Builds a modular security framework and a local monitoring agent integrating packet sniffer, device monitor, and rules engine. Works with AI and Security Developers to embed anomaly detection and response logic into the EDR pipeline, designing RESTful APIs for efficient edge–cloud communication and ensuring stable, scalable operation on lightweight devices.
User/Customer, AI/ML developer	Wonyoung Shin	Studies how AI can assist with corporate decision-making needs. As an AI developer, develops machine learning solutions to help companies make better data-driven decisions. An AI model related to LSTM and anomaly detection has been developed, and IoT security is improved through this. Focus on building AI models that analyze business data and provide clear insights for management teams.
Project Manager, AI/ML Engineer	Seungmin Son	It is important to assign tasks, manage schedules, and ensure all requirements are met. Oversees system integration and testing, and also maintains project documentation. Next, AI/ML Engineer is responsible to pre-process IoT data and train machine learning models(Isolation Forest, LSTM). And evaluating performance and integrating the models into the digital twin simulation environment for security analysis.

I. Introduction

I-A Background

The Internet of Things (IoT) has rapidly transformed modern homes into highly connected environments, integrating devices such as smart cameras, robot vacuums, sensors, and household appliances. While these devices bring automation and convenience, they also introduce substantial security and privacy risks by expanding the attack surface within residential networks. A single compromised device can enable lateral movement, unauthorized surveillance, or large-scale data exfiltration.

Consumer IoT devices are especially vulnerable because they are typically resource-constrained, rarely updated, and often deployed with weak authentication, outdated protocols, and limited visibility into their internal operations. Unlike traditional IT systems, these devices offer no standardized security interfaces, leaving users with no practical way to inspect what data their devices are transmitting or to whom.

The magnitude of these vulnerabilities is underscored by large-scale empirical research. In one of the most comprehensive global measurements to date, researchers scanned more than 16 million households and over 83 million IoT devices, uncovering widespread insecurity—including devices exposing open services, relying on weak default credentials, and communicating using outdated or insecure protocols. Complementary studies further reveal that a significant portion of commercially available smart home devices continue to use deprecated TLS versions or advertise weak cipher suites, making them susceptible to interception or manipulation.

Given these challenges, there is a critical need for an intelligent, lightweight, and privacy-preserving security framework operating directly at the network edge, where all device communications naturally converge. By analyzing network behavior rather than internal firmware, such a system can protect heterogeneous IoT devices without requiring modification of the devices themselves.

The objective of the SafeOn framework is to detect and respond to abnormal device behavior in real time while preserving user privacy and maintaining low computational overhead suitable for consumer home environments. The system leverages an edge-first Endpoint Detection and Response (EDR) architecture, flow-based behavioral modeling, and machine learning–driven anomaly detection to provide proactive protection for everyday IoT devices.

I-B Problem Statement

Traditional security approaches such as signature-based firewalls, antivirus engines, or cloud-centric monitoring are fundamentally ill-suited for today's IoT environments. Smart home devices rarely support full endpoint protection, lack computational capacity for on-device analysis, and operate with diverse protocols and proprietary firmware, making unified security enforcement nearly impossible.

Cloud-based detection introduces additional challenges:

- Latency hampers real-time response,
- Privacy concerns arise from uploading raw device data,
- Dependency on external infrastructure reduces reliability during network disruptions.

Furthermore, modern IoT attacks increasingly leverage stealth and mimicry. Adversaries gradually alter communication patterns, spoof legitimate services, or exfiltrate data using low-volume continuous traffic—making such attacks invisible to static rules or simple threshold-based systems.

To overcome these limitations, SafeOn introduces a privacy-preserving, edge-oriented EDR system designed specifically for smart home networks. Operating on an OpenWrt-based gateway, SafeOn captures every packet from IoT devices, aggregates them into short flow windows, and performs real-time anomaly detection without exporting sensitive payload data.

The proposed system combines:

- **Isolation Forest** — for unsupervised, real-time anomaly scoring on flow-level features,
- **Random Forest** — for supervised classification of suspicious versus benign flow patterns,

This hybrid detection pipeline enables the system to identify both known and previously unseen behaviors without requiring any intrusive access to device internals. By placing the intelligence directly at the network edge, the framework minimizes privacy risks, reduces latency, and delivers actionable protection for real-world smart home environments.

I-C Research on Related work

1 SunBlock

SunBlock is a cloudless IoT security system that runs entirely on a home router. It combines rule-based filtering (Snort3) with AI-based anomaly detection to identify and block suspicious traffic locally, without sending data to the cloud. This design preserves privacy, reduces latency, and effectively detects common IoT attacks such as unauthorized access and DoS.

2 LG Shield

LG Shield is LG Electronics' IoT security platform designed to protect connected home appliances and devices. It ensures secure communication between devices and the cloud by providing features such as device authentication, secure firmware updates, and data transmission encryption. LG Shield is integrated into the LG ThinQ ecosystem to enhance privacy and device reliability.

3 Samsung Knox

Samsung Knox is a multi-layered security platform that protects mobile and IoT devices from hardware to software levels. Built on a hardware root of trust and ARM TrustZone technology, Knox provides features such as secure boot, data encryption, application isolation, and enterprise mobile device management (MDM). It is widely used in both consumer and enterprise environments.

II. Threat Model

II-A System Context

SafeOn is an on-premise IoT defense framework designed to secure smart-home environments through an edge-oriented Endpoint Detection and Response (EDR) architecture. Deployed on an OpenWrt-based gateway such as a Raspberry Pi, SafeOn intercepts all traffic originating from household IoT devices—including cameras, sensors, robot vacuums, and smart appliances—before it reaches the wider internet. Rather than relying on intrusive firmware instrumentation or cloud services, the system passively collects network-level telemetry and transforms raw packets into short flow windows suitable for lightweight machine learning analysis. SafeOn employs a hybrid anomaly detection pipeline consisting of:

- Isolation Forest for real-time, unsupervised anomaly scoring,
- Random Forest for supervised classification of suspicious flow behaviors,

This design allows SafeOn to detect both instantaneous anomalies and persistent behavioral changes using only local inference. Because all detection logic runs on-premise at the network edge, SafeOn ensures low latency, strong privacy preservation, and does not expose raw device data to external cloud servers.This architecture provides a practical and deployable defense model suitable for real-world consumer IoT environments.

II-B Assets

SafeOn protects several critical digital and operational assets within the smart-home network. These assets fall into the following categories:

- **Data Assets**: Captured packet-flow windows, device discovery records, anomaly scores, classification outputs, and stored telemetry used for forensic analysis.
- **Functional Assets**: The ML inference pipeline—including feature extraction, Isolation Forest scoring, and Random Forest classification—as well as backend ingestion and dashboard interfaces.
- **Control Assets**: Local configuration states, device-blocking policies, model parameters, MQTT topic bindings, and gateway-level network rules executed through the SafeOn edge node.
- **Privacy Assets**: Sensitive user-relevant information implicitly revealed through device traffic patterns, usage timelines, and network behavior signatures.

Maintaining the confidentiality, integrity, and availability of these assets is essential to ensuring reliable real-time detection and response.

II-C Adversarial Model

SafeOn considers a range of adversarial threats commonly observed in residential IoT environments. The threat model includes the following attacker classes:

- **External Adversaries (WAN-side)**: Remote attackers attempting to compromise IoT devices through exposed services, weak authentication,

outdated protocols, or brute-force entry points. Their goals may include unauthorized data access, device hijacking, or covert command execution.

- **Internal Adversaries (LAN-side)**: Compromised IoT devices, malware-infected clients, or malicious insiders within the home network. These adversaries can generate traffic that mimics legitimate device behavior, perform low-rate data exfiltration, or manipulate flow patterns to evade anomaly detection.
- **Edge-level or Configuration Adversaries**: Attackers attempting to tamper with network configurations, manipulate MQTT messages, disrupt telemetry collection, or inject falsified flow data at the edge node. Such attackers aim to impair SafeOn's detection capabilities or conceal malicious behavior.
- **Firmware-level Adversaries**: Threat actors capable of modifying device firmware or exploiting insecure update channels. These adversaries may alter device networking behavior, disable security features, or cause persistent anomalous patterns without triggering obvious alarms.

These adversary categories reflect a shift toward more stealth-oriented and behavior-centric attacks, emphasizing the need for continuous, flow-based behavioral analysis rather than reliance on traditional signature-based defenses.

II-D Attack Vectors

Within this adversarial landscape, potential attack vectors against the SafeOn ecosystem can be categorized into five primary groups:

- **Network-level Attacks**: Unauthorized connection attempts, port scanning, brute-force access, and volumetric denial-of-service (DoS/UDP flooding) directed at vulnerable IoT devices or the OpenWrt-based gateway. These attacks aim to disrupt device availability or establish an initial foothold.
- **Flow-level Manipulation Attacks**: Adversaries may attempt to craft malicious packet sequences or alter traffic distributions to imitate benign flow patterns. By intentionally shaping packet sizes, timing intervals, or destination endpoints, attackers may try to evade flow-based anomaly detection.
- **Behavioral Evasion Attacks**: Since SafeOn models device behavior using flow windows and statistical feature patterns, adversaries may employ low-and-slow exfiltration, intermittent malicious bursts, or interleaving legitimate and malicious flows to remain below anomaly thresholds.
- **Firmware-level Attacks**: Attackers with deeper access may attempt to modify IoT device firmware to change networking behavior, introduce covert channels, or disable outward symptoms of compromise. Such threats may manifest as persistent anomalous patterns at the network layer.
- **Privacy-oriented Attacks**: Passive observers may attempt to infer user activity patterns, occupancy schedules, or device usage habits from unencrypted metadata, DNS queries, or timing characteristics of IoT communications.

These attack vectors reflect the multi-layered and behavior-driven nature of modern IoT threats, where deviations often appear not in payloads but in network flow characteristics.

II-E Security Goals

The SafeOn framework is designed to achieve the following core security objectives:

- **Confidentiality**: All telemetry—including flow windows, device states, and anomaly scores—is processed locally on the edge gateway. No raw packet data or device behavior is uploaded to external cloud services.
- **Integrity**: Machine learning model files, configuration parameters, and historical logs are protected from tampering through controlled file permissions and local verification mechanisms, ensuring accurate and trustworthy inference.
- **Availability**: The detection pipeline operates autonomously on the OpenWrt gateway, continuing to function even during WAN outages. Local MQTT communication ensures uninterrupted data flow between collection, backend, and ML components.
- **Accuracy and Robustness**: The combination of **Isolation Forest** for unsupervised anomaly scoring and **Random Forest** for supervised flow classification enhances resilience against both known and previously unseen attack patterns.
- **Accountability**: All alerts, classification outputs, blocking decisions, and system responses are logged locally on the backend, enabling

transparent forensic investigation after an incident.

These goals reinforce SafeOn's commitment to secure, privacy-preserving, and reliable operation within consumer home networks.

II-F Threat–Defense Mapping

SafeOn incorporates multiple defensive components aligned with its attack model:

- **Network-level Attacks**: Mitigated through gateway-level traffic monitoring, device isolation within a dedicated IoT subnet, and continuous flow extraction that exposes abnormal connection attempts or traffic bursts.
- **Flow-level Manipulation**: Countered by the Isolation Forest model, which identifies deviations in multivariate flow statistics such as packet counts, byte distributions, and inbound/outbound ratios.
- **Behavioral Evasion Attempts**: Detected through combined analysis of anomaly scores and Random Forest classification, which captures subtle deviations across aggregated flow windows even under low-rate or intermittent attack strategies.
- **Firmware-level Manipulation**: Persistent or unnatural communication patterns emerging from compromised firmware are reflected in long-term flow trends and can be identified through statistical abnormalities and repeated anomaly scoring.
- **Privacy Attacks**: Minimized by keeping all telemetry local, discarding unnecessary metadata, and avoiding any cloud-based data transfer.

Together, these mechanisms provide layered defense and visibility into IoT network behavior without requiring modifications to device hardware or firmware.

III. Requirements

This section summarizes the high-level requirements of the SafeOn system. A more detailed specification, including UI flows and system diagrams, will be provided in later sections.

III-A Functional Requirements(FR)

- **FR-01: User Authentication**
  The system shall provide secure user registration and login using JWT-based authentication.
- **FR-02: IoT Device Discovery**
  The system shall automatically detect newly connected IoT devices from the OpenWrt gateway and display them to the user as unclaimed devices.
- **FR-03: Device Claiming**
  Users shall be able to register ownership of discovered devices.
- **FR-04: Flow Generation and Transmission**
  The gateway shall capture raw packets using `tcpdump`, aggregate them into 3-second flow windows, and transmit flow features to the backend via MQTT.
- **FR-05: Flow Ingestion and Storage**
  The backend shall receive flow data from MQTT, validate it, and store it in TimescaleDB for further analysis.
- **FR-06: Machine Learning Inference**
  The backend shall forward flow features to the ML server and receive anomaly scores (Isolation Forest) and classification labels (Random Forest).
- **FR-07: Anomaly Detection and Alerting**
  The system shall generate an alert when suspicious activity is detected (e.g. repeated anomalies, malicious prediction, external IP access). Alerts shall be delivered to the frontend in real time.
- **FR-08: Real-Time Traffic Monitoring**
  The backend shall provide a WebSocket endpoint to stream device traffic statistics to the dashboard.
- **FR-09: Device Blocking**
  Users shall be able to block a compromised device, and the backend shall send a block command to the gateway to isolate the device.

III-B Non-functional Requirements(NFR)

- **NFR-01: Performance**
  End-to-end anomaly detection latency should remain within a few seconds.
- **NFR-02: Privacy**
  No raw packet payloads shall be transmitted outside the local system.
- **NFR-03: Availability**

The system shall continue operating locally even if internet connectivity is temporarily unavailable.

- **NFR-04: Security**
  All APIs and WebSocket connections shall require authentication.
- **NFR-05: Usability**
  The dashboard shall clearly visualize device status, alerts, and real-time traffic in an accessible manner.

III-C Constraints

- **C-01: Edge Hardware Limitations**
  The OpenWrt gateway has limited CPU and memory; therefore, all ML inference must be performed on an external ML server.
- **C-02: Wireless Network Variability**
  IoT device traffic depends on Wi-Fi signal stability and may fluctuate.
- **C-03: MQTT Dependency**
  Flow transmission relies on a stable MQTT connection between the gateway and backend.
- **C-04: Storage Constraints**
  High-frequency flow data requires efficient time-series storage and retention policies.

III-D AI Model

To provide real-time detection of abnormal behavior in heterogeneous IoT devices, *SafeOn* adopts a flow-based anomaly detection pipeline tightly integrated with the edge-first architecture. The AI component of the system is organized into three stages:

1 Dataset Construction from Real IoT Traffic

we build our dataset directly from packet traces collected in a realistic smart-home setting. Using `tcpdump` on the OpenWrt-based gateway, we capture raw network traffic generated by an ESP32-CAM under both benign usage and attacker-driven scenarios.

The captured packets are aggregated into 3-second time-windowed flows and converted into flow records containing statistics such as packet count, byte count, duration, and throughput. Each flow is then labeled as *normal* or *attack* based on the ground-truth scenario during data collection.

The final dataset consists of:

- 16 366 **normal** flows,
- 10 504 **attacker** flows.

This results in a moderately imbalanced but realistic traffic distribution that reflects typical smart-home operation with intermittent malicious activity. The dataset is split into training and evaluation subsets, and feature-wise normalization is applied where required.

2 Flow-based Feature Engineering and Preprocessing

Each flow window is represented as a fixed-length feature vector derived from the underlying packet sequence. The features include basic statistics (packet and byte counts, flow duration), rate-based metrics (packets per second, bytes per second), and simple structural attributes (source/destination ports, protocol type, time bucket).

These features are designed to be lightweight enough for real-time processing at the edge, while still capturing meaningful differences between benign and malicious behaviors such as scanning attempts, abnormal bursts, or unusual external communications. Categorical attributes (e.g., protocol) are encoded appropriately, and numerical features are standardized to stabilize model training.

3 Hybrid Anomaly Detection with Isolation Forest and Random Forest

The core AI engine consists of two complementary models:

- **Isolation Forest** is used as an unsupervised model to compute an anomaly score for each flow based on how easily it can be isolated in the feature space.
- **Random Forest** is trained in a supervised manner on the labeled dataset to classify flows as benign or attacker traffic.

Both models are implemented on a separate FastAPI-based ML server and exposed via lightweight inference endpoints. During operation, the backend forwards newly arrived flow features to the ML server and receives an Isolation Forest score and a Random Forest prediction in response.

SafeOn combines these outputs into a hybrid decision rule: flows with high anomaly scores and malicious Random Forest predictions are

marked as suspicious. When consecutive suspicious flows are observed from the same device, SafeOn generates an alert and notifies the user through the dashboard. Blocking is not performed automatically; instead, the user may initiate a containment action. Upon user request, the backend transmits a block command to the OpenWrt gateway, which enforces device isolation via firewall rules. This design enables robust detection of both clearly malicious and more subtle anomalous behaviors while remaining computationally efficient for deployment in consumer smart-home environments.

IV. Development environment

IV-A Choice of Software Development Platform

1) Development Platform

    1) Windows

        Windows is Microsoft's operating system that provides a comprehensive development environment through its extensive tools and frameworks. Its core IDE Visual Studio enables development across multiple languages, while the .NET framework supports diverse application development. The platform features Windows Terminal, Package Manager, and Windows Subsystem for Linux (WSL) for modern development workflows. Windows offers strong enterprise integration, PowerShell automation, DirectX support for gaming, and robust security features. Its Azure cloud integration and widespread market presence ensure broad compatibility and testing capabilities, making it a versatile development platform for various applications.

    2) macOS

        macOS is Apple's operating system providing a robust development environment centered around Xcode for Apple ecosystem development. Its Unix foundation offers powerful command line capabilities, while package managers like Homebrew facilitate tool management. The platform excels in native development through Swift and Objective C support, and includes comprehensive debugging and performance optimization tools. macOS integrates smoothly with Apple services like iCloud and TestFlight for deployment, while maintaining security through features like Gatekeeper. Its UNIX certification and virtual machine support enable versatility across different development scenarios.

2) Language and Framework

Figure 1: Python

    1) Python

        Python is a versatile and widely used high-level programming language, praised for its simplicity and readability. This makes it particularly attractive for both beginners and experienced developers. Python's extensive standard library and rich ecosystem of third-party libraries provide powerful tools for various tasks, including web development, data analysis, and artificial intelligence. The language's strong support for object-oriented, imperative, and functional programming paradigms allows developers to choose the style that best fits their needs. Furthermore, Python is heavily utilized in the AI community due to its robust frameworks and libraries that facilitate tasks such as data preprocessing, model building, and evaluation.

    2) Dart

        Dart is a modern, object-oriented language developed by Google, designed for both efficiency and readability. It combines the simplicity of dynamic languages with the performance advantages of static

Figure 2: Dart

typing and supports multiple paradigms such as object-oriented, imperative, and functional programming. Dart features clean and expressive syntax, a dual compilation model—JIT for fast development and AOT for optimized builds—and strong asynchronous programming support. With its rich ecosystem and integration as the core language of Flutter, Dart enables high-performance, cross-platform application development.

3) Java

Figure 3: Java

Java is a powerful, object-oriented programming language designed for portability, reliability, and performance. Its core principle, "write once, run anywhere," is made possible by the Java Virtual Machine (JVM), allowing code to run seamlessly across platforms. Java emphasizes strong typing, automatic memory management, and clear syntax, which enhance code stability and readability.
With its extensive standard library and frameworks like Spring, Java supports a wide range of development—from web and mobile apps to enterprise-scale systems. Its support for multithreading, modularity, and robust security makes it a preferred choice for large and scalable software projects. Furthermore, Java continues to evolve, incorporating modern features such as lambdas, streams, and modular programming to improve productivity and expressiveness.

4) FastAPI

Figure 4: FastApi

FastAPI is a modern, high-performance Python framework for building APIs, designed to simplify development through Python type hints and automatic validation. Built on Starlette and Pydantic, it delivers fast execution comparable to Node.js and Go. One of its strengths is the automatic generation of interactive API documentation via OpenAPI, including Swagger UI and ReDoc. FastAPI also supports asynchronous programming with async/await, making it suitable for high-concurrency workloads. With features such as dependency injection, data validation, and clear schema enforcement, FastAPI reduces boilerplate code and enhances reliability. Its speed, simplicity, and strong tooling make it an ideal choice for RESTful APIs, ML model serving, and lightweight microservices.

5) Flutter

Figure 5: Flutter

Flutter is an open-source UI framework

by Google that enables cross-platform application development from a single codebase. It provides a rich set of customizable widgets and tools for building responsive, visually appealing UIs, using Dart for fast compilation and expressive syntax. A key feature of Flutter is Hot Reload, which lets developers instantly see code changes and boosts productivity. Its layered architecture offers fine-grained control over rendering and performance. With strong third-party package support and a rapidly growing ecosystem, Flutter has become a popular choice for efficient, flexible, and high-quality cross-platform development.

6) Spring Boot

Figure 6: Spring Boot

Spring Boot is a powerful framework that simplifies the development of Java-based web applications and microservices. It builds on the Spring framework by providing automatic configuration, embedded servers, and production-ready features out of the box. With built-in starter dependencies, developers can quickly set up projects without dealing with complex configuration files. Spring Boot also offers tools for monitoring, health checks, and performance management, making it easy to deploy and maintain scalable applications. Its seamless integration with the broader Spring ecosystem—such as Spring Security, Spring Data, and Spring Cloud—enables efficient development of secure, data-driven, and distributed systems. By reducing boilerplate code and configuration overhead, Spring Boot allows developers to focus on writing business logic and delivering high-quality, maintainable software.

3) Development Environment

1) On Local Machine

Table II: On Local Machine

Name	**Computer Resource**
Jaemin Jeong (Software Developer)	Apple M2 16GB RAM
Juseong Jeon (Security Developer)	Linux mint AMD 8GB
Seungmin Son (AI/MLdeveloper)	Apple M4 16GB RAM
Wonyoung Shin (AI/MLdeveloper)	Apple M4 16GB RAM

IV-B Software in use

1) Visual Studio Code (VS Code)

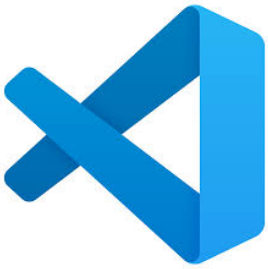

Figure 7: VS Code

Visual Studio Code (VS Code) is a flexible, open-source code editor known for strong performance and broad language support. It includes an integrated terminal, debugging tools, and Git features, making it a capable environment for both solo and team development.

Its extensible plugin ecosystem allows developers to tailor workflows to different languages and frameworks, while IntelliSense provides intelligent code completion and real-time error checking. With its intuitive interface and customizable settings, VS Code has become one of the most widely used tools in modern software engineering.

2) MQTT Broker

Figure 8: MQTT Broker

The MQTT Broker serves as the core messaging infrastructure of our system, enabling lightweight, low-latency communication between the edge gateway, backend server, and AI inference module. MQTT follows a publish–subscribe architecture, allowing devices to exchange data asynchronously with minimal network overhead—an essential requirement for IoT environments with constrained bandwidth.

In SafeOn, the broker receives flow-level telemetry published by OpenWrt and relays it to the backend for storage and analysis. Its ability to handle persistent sessions, Quality of Service (QoS) levels, and scalable topic management ensures reliable data transmission even under intermittent connectivity. This makes MQTT a critical component for achieving real-time monitoring and responsive anomaly detection.

3) OpenWrt

Figure 9: OpenWrt

OpenWrt is a Linux-based embedded operating system widely used for routers and edge devices, providing a highly customizable and extensible platform for network management. It offers full root-level access, a package-based architecture, and powerful networking utilities, making it suitable for advanced routing, monitoring, and packet inspection tasks.

In our system, OpenWrt operates as the edge security gateway responsible for capturing raw IoT traffic, extracting flow metadata, and publishing telemetry to the MQTT Broker. Its lightweight footprint and robustness enable real-time processing directly at the network boundary, reducing latency and improving the responsiveness of SafeOn's anomaly detection pipeline.

4) TCPDump

Figure 10: TCPDump

TCPDump is a command-line packet capture

tool capable of intercepting and analyzing raw network traffic at the packet level. It provides fine-grained access to headers and payloads, supporting various filtering expressions to isolate specific protocols, ports, or device traffic.
Within SafeOn, TCPDump runs on OpenWrt to record network activity from IoT devices, generating the packet streams used to construct flow-level features for downstream machine learning analysis. Its reliability, efficiency, and ability to operate in resource-constrained environments make it an ideal choice for continuous monitoring in home IoT networks.

5) PyTorch

Figure 11: PyTorch

PyTorch is a widely used open-source deep learning library known for its flexibility and intuitive design. Built for Python, it offers a dynamic computational graph that makes constructing and debugging neural networks straightforward while maintaining strong performance.
Its ecosystem includes tools for data processing, training, evaluation, and deployment, all with seamless GPU acceleration for fast experimentation. Supported by an active open-source community and frequent updates, PyTorch remains a leading framework in modern AI research and real-world applications.

6) scikit-learn

Figure 12: scikit-learn

Scikit-learn is a widely adopted machine learning library in Python, offering efficient implementations of classical algorithms for classification, clustering, and anomaly detection. Its simple API and strong integration with NumPy and SciPy enable rapid experimentation and prototyping.
In this project, scikit-learn provides models such as Isolation Forest for snapshot-level flow anomaly detection, serving as a lightweight and interpretable baseline for identifying irregular device behavior. Its consistent performance and ease of deployment make it well-suited for real-time inference within the SafeOn architecture.

7) Vim

Figure 13: Vim

Vim is a highly efficient, keyboard-centric text editor renowned for its speed, precision, and flexibility. Built upon the legacy of the original Vi editor, Vim introduces a modal editing system that separates tasks into distinct modes—such as insert, command, and visual—allowing developers to perform text manipulation and navigation with exceptional speed and accuracy.
Although Vim presents a steeper learning curve compared to conventional editors, many developers find that mastering its commands and shortcuts greatly enhances overall produc-

tivity and editing efficiency. Its lightweight design and terminal-based operation make it particularly well-suited for remote development, server environments, and quick file modifications across multiple projects.
By emphasizing efficiency through keystroke-driven control, Vim remains an indispensable tool for developers who prioritize speed, focus, and minimalism in their workflow.

8) Figma

Figure 14: Figma

Figma is a web-based UI/UX design platform that supports real-time collaboration and streamlines the workflow from wireframing to interactive prototyping. Its design system features help maintain visual consistency and manage reusable components efficiently.
Developers can easily extract assets and specifications, enabling a smooth handoff to implementation. With its component-based approach and strong alignment with frameworks like React Native, Figma has become an essential tool for modern, collaborative design teams.

9) Xcode 16.4

Figure 15: Xcode 16.4

Xcode 16.4 is Apple's integrated development environment for building, testing, and deploying apps across iOS, iPadOS, macOS, watchOS, and tvOS. It includes the iOS 18.5 SDK and provides tools such as the real-time iOS Simulator, performance profiling, memory diagnostics, and UI testing.
In React Native projects, Xcode is essential for generating iOS builds, managing signing certificates, and debugging native components. Faster build times, improved Swift/SwiftUI support, and integration with TestFlight and App Store Connect further enhance developer productivity. Xcode remains the core environment for creating reliable, production-ready applications on Apple platforms.

10) PostgreSQL

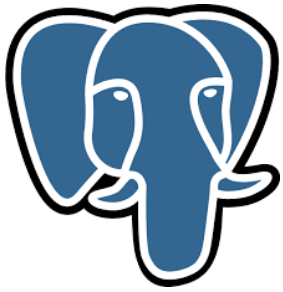

Figure 16: PostgreSQL

PostgreSQL is a robust open-source relational database known for reliability, extensibility, and full SQL compliance. It supports advanced features such as window functions, JSON operations, and geospatial data through PostGIS.
With ACID guarantees and MVCC, PostgreSQL delivers consistent, high-performance transaction processing and supports replication, partitioning, and high availability for large-scale systems. Its extensible design allows custom data types and functions, while strong security features—including role-based access control and SSL/TLS—ensure data integrity. These capabilities make PostgreSQL a trusted choice for modern, data-intensive applications.

11) TimescaleDB

Figure 17: TimescaleDB

TimescaleDB is an open-source time-series

database built on PostgreSQL, designed to efficiently store and analyze large volumes of time-stamped data. It combines PostgreSQL's reliability with optimized time-series features, making it well suited for IoT monitoring, system metrics, and sensor telemetry.
With full SQL compatibility, TimescaleDB offers powerful tools such as hypertables, continuous aggregates, and automated data retention, enabling high-ingest performance without complex configuration. It also integrates smoothly with visualization tools like Grafana and Prometheus, providing a complete environment for real-time analytics. Its scalability and ease of integration make it a strong solution for modern time-series workloads.

12) Discord

Figure 18: Discord

Discord is a communication and collaboration platform that provides real-time voice, video, and text messaging, enabling teams to coordinate efficiently from anywhere. Its server-based structure organizes discussions into topic-specific channels, supporting clear communication and task management.
With robust API and bot integration, Discord allows automated workflows and integrations with tools like GitHub or Jenkins. Its cross-platform support ensures seamless access across devices. Combining reliability, flexibility, and ease of use, Discord has become a versatile environment for project collaboration, technical discussions, and community management.

13) GitHub

Figure 19: GitHub

GitHub is a web-based platform for Git version control that allows developers to host, share, and collaborate on code efficiently. It supports pull requests, issue tracking, and code reviews, enabling transparent teamwork. Beyond version control, GitHub provides CI/CD automation through GitHub Actions, project boards for organization, and security features like dependency scanning and secret detection. Used widely across industry and open-source communities, GitHub has become a core tool for modern software development.

14) Overleaf

Figure 20: Overleaf

Overleaf is a web-based LaTeX editing platform that simplifies the creation and management of scientific documents. It supports real-time collaboration, allowing multiple users to edit simultaneously with automatic synchronization. The platform includes built-in compilation, syntax highlighting, autocompletion, and instant PDF previews, removing the need for local LaTeX installations.
With a wide range of academic templates and seamless integration with tools like Zotero and Mendeley, Overleaf streamlines citation

management and document organization. Its accessibility, collaboration features, and comprehensive LaTeX environment make it a widely used tool for academic writing and research.

15) Notion

Figure 21: Notion

Notion is a collaborative workspace that combines document management, note-taking, and project organization tools. Its block-based structure allows users to create and arrange various content types within a hierarchical page system. The platform offers templates for different purposes and real-time collaboration features. Notion includes database capabilities, version history, and integration options with external tools. The flexible interface lets users customize their workspace organization and layout to match specific workflows.

16) ChatGPT

Figure 22: ChatGPT

ChatGPT is an OpenAI language model that functions as a conversational AI assistant and customizable model platform. Based on transformer architecture and extensive training data, it handles diverse tasks from answering questions to coding assistance. The system maintains contextual awareness in conversations and generates responses according to its training. It features multilingual support, adaptable communication styles, and capabilities in summarization, translation, and creative writing. ChatGPT includes safety measures and clearly indicates its knowledge limitations through its training cutoff date.

IV-C Hardware & Runtime Environment

1) Raspberry Pi (Edge Gateway)

The Raspberry Pi serves as the edge gateway for the SafeOn system, operating as the central hub for network monitoring within the home environment. Its low-power ARM architecture, sufficient processing capability, and strong compatibility with embedded Linux systems make it suitable for continuous packet capture and lightweight computation tasks. Running OpenWrt, the device performs TCP-based traffic collection, real-time flow generation, and MQTT-based telemetry publishing to the backend server. By functioning as a dedicated security node independent of the main router, the Raspberry Pi ensures reliable, localized threat detection and minimizes latency in end-to-end communication.

2) IoT Devices (ESP32-CAM)

Various IoT devices within the home network serve as data sources for the SafeOn monitoring pipeline. Devices such as the Ecovacs robot vacuum and the ESP32-CAM module generate traffic patterns that reflect typical household IoT behavior. The ESP32-CAM, in particular, produces high-frequency video-stream data, enabling the system to analyze heavy-flow conditions, motion-triggered events, and API communication patterns. These devices provide diverse network activities that support accurate modeling of normal and anomalous behavior, forming the basis for flow-level anomaly detection and device fingerprinting.

3) Router

The home router establishes the local network environment and provides the upstream internet connection for all IoT devices. While

it performs standard network functions such as DHCP, NAT, and wireless access management, the SafeOn architecture delegates security monitoring to the Raspberry Pi gateway to avoid router performance bottlenecks. The router ensures stable routing paths for device communication, while the edge gateway intercepts and processes traffic passively, enabling non-intrusive, real-time network analysis without modifying the router's firmware or configuration.

IV-D Computer resources

Table III: Computer resources

Name	Task
Jaemin Jeong	Apple M2 16GB RAM
Juseong Jeon	Linux mint AMD 8GB
Wonyoung Shin	Apple M4 16GB RAM
Seungmin Son	Apple M4 16GB RAM

IV-E Task distribution

Table IV: Task distribution

Name	Task
Jaemin Jeong	Back-end Development
Juseong Jeon	Security Development
Wonyoung Shin	Front-end, ML Development
Seungmin Son	ML Development

V. Requirement Specification

V-A User-Facing Functional Specifications

V-A1 1) Entry

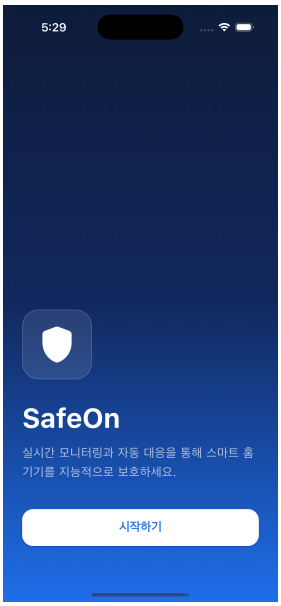

Figure 23: Entry

ID	Name	Description
001	SafeOn-Entry-Page	The user is introduced to the SafeOn service and can proceed by tapping the Start button. This page serves as the entry point to the authentication flow.

V-A2 2) User Authentication Screens

a) Sign-up Page

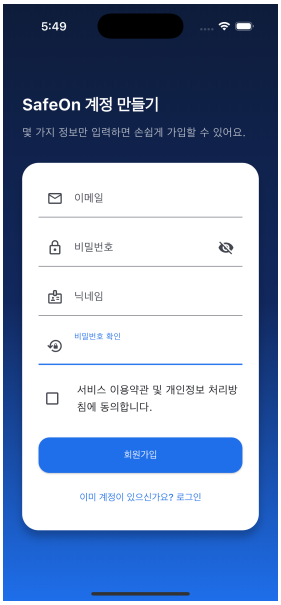

Figure 24: Sign-up

ID	Name	Description
002	SafeOn-Signup-Page	The user provides email, password, nickname, and privacy agreement. On success, a new account is created in the backend.
003	SafeOn-Signup-Email	The email address is a required field used as the user's unique identifier. It must include a valid format with '@' and domain, and the system checks for duplicates in real time. Errors such as "Please enter a valid email address" or "This email is already in use" are shown for invalid or duplicate inputs.
004	SafeOn-Signup-Password	The password is a mandatory field for securing the user's account. It must meet SafeOn's password policy, including minimum length and valid character composition. If the input is invalid, the system shows messages like "Password does not meet security requirements."
005	SafeOn-Signup-Username	The username is a required profile identifier that must follow allowed character rules and remain unique. Invalid or duplicate entries trigger messages such as "Please enter a valid username" or "This username is already taken."
006	SafeOn-Signup-Privacy-Agreement	The privacy agreement is a required consent field for creating an account. If the user does not agree, the system blocks signup and displays guidance such as "You must agree to the privacy policy to continue."

b) Login Page

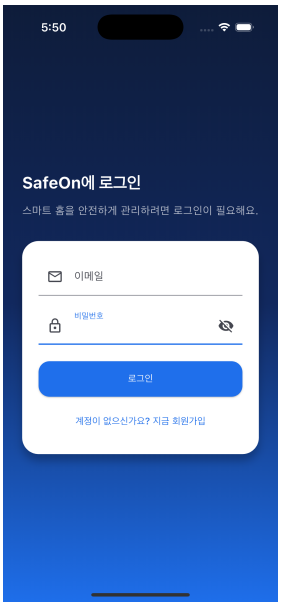

Figure 25: Login

ID	Name	Description
007	SafeOn-Login-Page	The user enters email and password to authenticate. Successful login returns a JWT token used for all subsequent API calls.
008	SafeOn-Login-Success	Successful login occurs when the user provides valid email and password credentials. Upon verification, the system issues a JWT token, which is required for all subsequent authenticated requests. The user is then redirected to the main dashboard.
009	SafeOn-Login-Failure	Login failure occurs when the provided credentials do not match any registered account. The system displays error messages such as "Invalid email or password," and authentication is denied.

V-A3 3) Mainpage

i) Mainpage DashBoard

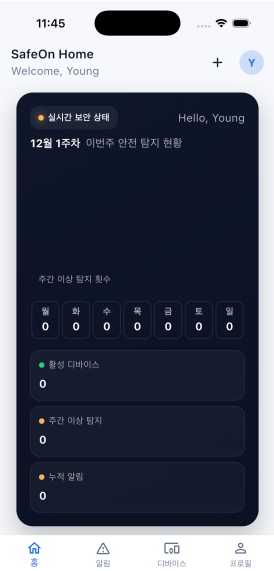

Figure 26: Mainpage DashBoard

ID	Name	Description
010	SafeOn-Mainpage-DashBoard	Displays weekly anomaly statistics by day and presents the overall security status of the network. Real-time updates are delivered through SSE, allowing users to monitor ongoing activity at a glance.
011	SafeOn-Mainpage-devices	Shows the number of active devices currently connected to the SafeOn router. Users can view device presence and network activity in real time.
012	SafeOn-Mainpage-alarts	Displays the cumulative number of anomaly alerts detected by the SafeOn system, providing users with a quick measure of recent security activity.

ii) Mainpage Filled DashBoard

Figure 27: Mainpage Filled DashBoard

ID	Name	Description
013	SafeOn-Mainpage-Filled-Dashboard	Displays a fully populated dashboard view showing real-time system status, including the total number of registered devices, current network health, cumulative alert count, and recent alert activity. This view provides users with an at-a-glance summary of their smart-home security state based on the latest data received from the backend.

V-A4 4) Device Management Screens

i) Newly Discovered Device (Empty State)

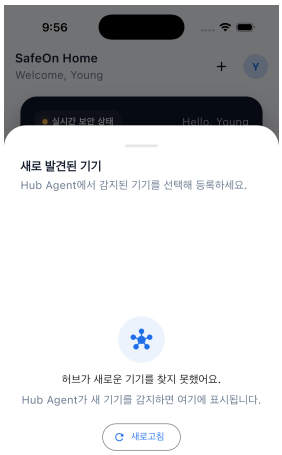

Figure 28: New Device Wait

ID	Name	Description
013	SafeOn-Devicepage-EmptyState	This screen appears when the Hub Agent has not detected any new devices. It informs the user that no discoverable devices are currently available and provides a “Refresh” button to request a new scan. The popup remains active and automatically updates when the Hub Agent reports a newly discovered device.

ii) Newly Discovered Device (Detection State)

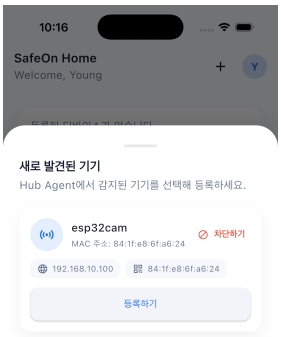

Figure 29: New Device Find

ID	Name	Description
014	SafeOn-Devicepage-DetectionState	When the Hub Agent detects a new device on the network, the device is displayed along with its name, MAC address, and IP address.
015	SafeOn-Devicepage-RegisterDevice	Selecting “Register” adds the device to the SafeOn device list. The backend records the device’s metadata, assigns it a unique identifier, and marks the device as trusted.

iii) Device Block

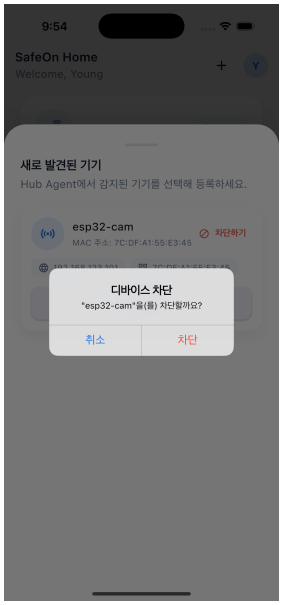

Figure 30: Device Block

iv) Registered Device List Page

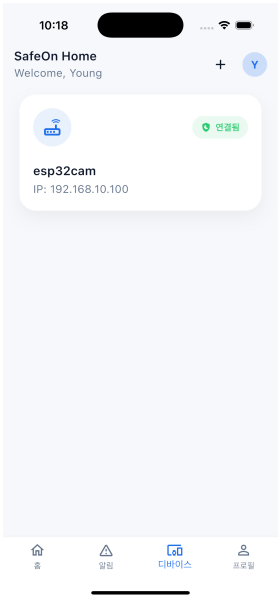

Figure 31: Connected Device List

ID	Name	Description
016	SafeOn-Devicepage-BlockDevice	Choosing “Block” triggers the device-blocking workflow: The app sends a block request to the backend, which updates the OpenWrt firewall to deny the device’s network access. The device is not added to the trusted list and remains blocked, preventing unknown or suspicious devices from connecting to the smart-home network.

ID	Name	Description
017	SafeOn-Devicepage-Connected-List	Shows all devices currently registered to the user’s SafeOn network. Each device card displays the device name, IP address, and real-time connection status. This page allows users to quickly verify active devices and manage their smart-home environment.

v) Device Detail Page

A) Device Information Section

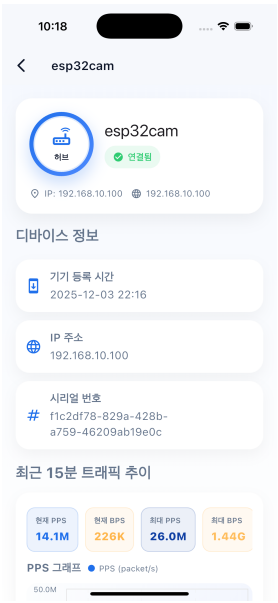

Figure 32: Device Detail Information

ID	Name	Description
018	SafeOn-Devicepage-Detail-Information	Provides key identification data for the device, including IP address, MAC address, registration time, and serial number. These details allow users to verify the device's identity, based on authoritative information reported by the Hub Agent.

B) Real-Time Traffic Monitoring Section

Figure 33: Device Detail Flows

ID	Name	Description
019	SafeOn-Devicepage-Detail-Flows	Shows two flow-based charts visualizing the last 15 minutes of activity: • PPS: current and peak packet rates • BPS: current and peak bandwidth usage The graphs update with recent edge-collected flow data and help detect abnormal patterns such as unexpected spikes or bursts.

V-A5 5) Alert Management Page

i) Alert List Page (Empty State)

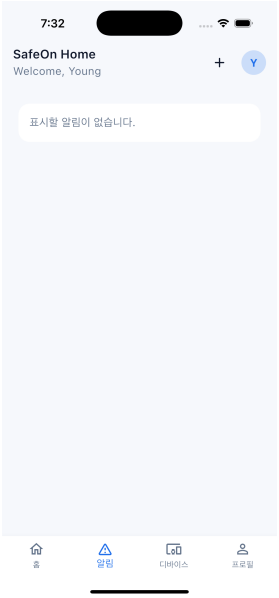

Figure 34: Empty Alert List

ID	Name	Description
020	SafeOn-Alertpage-EmptyList	Displayed when no anomaly events have been generated by the ML inference engine. The screen informs the user that there are currently no alerts and remains idle until a new anomaly is detected. This state provides a clean baseline view, indicating that the smart-home network is operating normally.

ii) Alert List Page (Filled State)

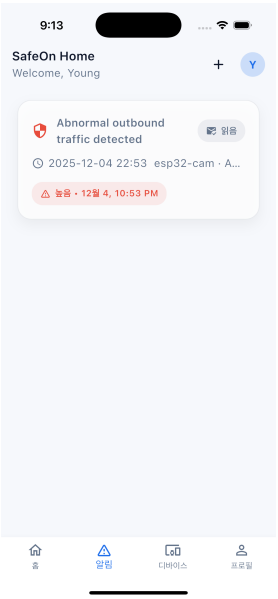

Figure 35: Filled Alert List

ID	Name	Description
021	SafeOn-Alertpage-FilledList	Displays a chronological list of anomaly alerts generated by the ML engine. Each alert card shows the timestamp, risk level, and device information. Users can browse recent security events and select an entry to view its detailed information.
022	SafeOn-Alertpage-ReadStatus	Provides a read/unread indicator for each alert. New alerts are marked as unread, and the status updates when the user opens or acknowledges the alert. This feature helps users distinguish newly generated events from previously reviewed ones.

iii) Alert Detail Page (Information)

Figure 36: Alert Detail Page

iv) Alert Detail Page (Block)

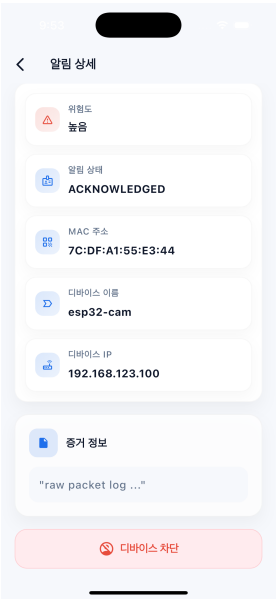

Figure 37: Alert Detail Page Block

ID	Name	Description
023	SafeOn-Alertpage-Detail	Provides full information about a selected alert, including anomaly type, risk level, device IP/MAC, and the generated timestamp. The page also indicates the alert state (e.g., acknowledged) and includes metadata derived from the ML pipeline such as anomaly confidence or category. This view helps users understand the context of the event and determine appropriate follow-up actions.

ID	Name	Description
023	SafeOn-Alertpage-Detail-Block	Enables the user to immediately block the device associated with the alert. When activated, a block command is sent to the gateway, isolating the device from the network to prevent further suspicious activity.

V-A6 6) My Page Screens

A) Profile Overview Page

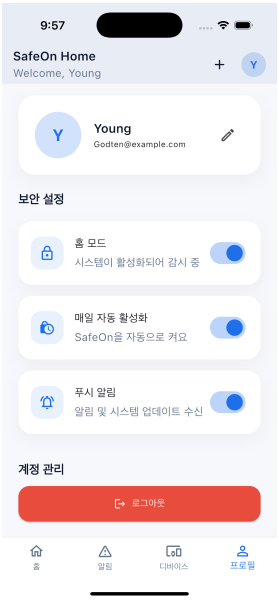

Figure 38: Profile Overview Page

ID	Name	Description
024	SafeOn-Mypage	Displays the user's basic account information including email address and nickname. The page also provides security-related settings such as Home Mode activation, daily auto-activation, and push notification preferences. A logout button is available for ending the session.

B) Profile Edit Page

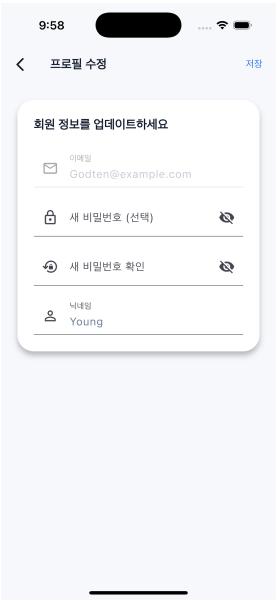

Figure 39: Profile Edit Page

ID	Name	Description
025	SafeOn-Mypage-Edit	Allows the user to update personal information. The email field is shown as read-only, while the user may optionally set a new password and update their nickname. Changes are submitted to the backend, and upon success, the updated profile is reflected throughout the app.

C) Logout

Figure 40: Logout

ID	Name	Description
025	SafeOn-Mypage-Logout	Allows the user to securely terminate their session and return to the login screen. Upon logout, all locally cached user information is cleared, ensuring privacy and preventing unauthorized access.

VI. System Architecture

VI-A System Overview

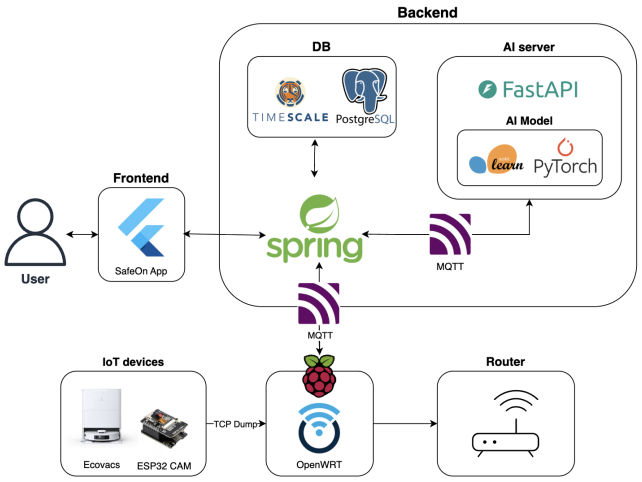

Figure 41: System Architecture

SafeOn adopts a four-layer edge-first architecture designed to secure heterogeneous smart-home IoT environments. All IoT devices connect through **an OpenWrt-based gateway layer**, which captures network traffic and generates flow-level telemetry. **The backend layer** processes these flows, stores time-series data, and coordinates communication with the machine learning server. **The ML layer** performs anomaly detection using Isolation Forest and Random Forest models, while **the frontend layer** dashboard provides real-time alerts, device monitoring, and blocking controls. This layered architecture enables privacy-preserving, low-latency detection and response without modifying IoT device firmware or relying on external cloud services.

VI-B System Components

The SafeOn architecture is composed of four cooperating layers that together realize continuous monitoring, anomaly detection, threat response, and user interaction.

VI-B1 1) Packet Collection Layer (OpenWrt Router)

This layer is deployed on a Raspberry Pi running OpenWrt, functioning as a dedicated IoT security gateway. All IoT devices associate to this access point, allowing SafeOn to observe every inbound and outbound packet in the IoT subnet.

Three core services operate on the edge node:

- **Device Detection Service**: Monitors DHCP and wireless events to automatically detect newly connected devices and publishes structured discovery messages to the backend via MQTT. These devices initially appear as "unclaimed" until a user registers ownership through the dashboard.
- **Flow Generation Service**: Captures raw packets using tcpdump and aggregates them into 3-second flow windows. Each window is transformed into a statistical feature vector covering packet counts, byte volume, interval variance, and directional ratios before being transmitted to the backend.
- **Command Listener Service**: Listens for block commands issued by the backend and enforces isolation through firewall rules, immediately restricting communication for compromised devices.

This design ensures strict privacy because only statistical flow summaries are transmitted; no packet payloads ever leave the home.

VI-B2 2) Backend Processing Layer (Spring Boot)

The backend serves as the central processing hub that binds the router, machine learning server, and dashboard. Its responsibilities include:

- Subscription to router-origin MQTT topics for receiving device discovery messages and flow telemetry.
- Feature normalization and time-series storage using PostgreSQL + TimescaleDB.
- Forwarding validated flow vectors to the ML server for inference.
- Applying SafeOn's hybrid detection policy, which considers both ML scores and contextual rules such as the presence of external IP addresses.
- Issuing device-block commands when the user requests containment.
- Managing user-device associations and dis-

tributing real-time security alerts.
- Serving REST APIs for authentication, device management, alert retrieval, and dashboard data queries.

Additionally, the backend exposes a WebSocket endpoint that streams real-time device traffic statistics to the dashboard, enabling continuous visualization.

VI-B3 3) Machine Learning Layer (FastAPI)

This layer hosts the anomaly detection models:
- **Isolation Forest** — computes unsupervised anomaly scores based on multivariate statistical flow features;
- **Random Forest** — performs supervised classification to distinguish malicious from benign flows.

The ML server returns inference results with minimal latency through MQTT, supporting near real-time detection. Because the models rely solely on flow-level features, they generalize effectively across heterogeneous IoT devices.

VI-B4 4) Frontend Dashboard Layer (Flutter App)

The dashboard provides users with:
- Real-time device traffic visualization via a secure WebSocket stream;
- Automatic display of alerts, including reasons, severity, and packet metadata;
- Device-claiming and device-blocking workflows;
- Historical trend graphs and anomaly logs retrieved from the backend.

Through this interface, users maintain situational awareness and can take immediate action in response to anomalies.

VI-C Data Flow and Detection Pipeline

The overall SafeOn pipeline consists of four continuous stages:

VI-C1 1) Traffic Capture and Flow Extraction

IoT packets are captured by the OpenWrt gateway using tcpdump. Packets are grouped into flow windows, and features such as packet counts, byte statistics, interval variance, and inbound/outbound ratios are computed.

VI-C2 2) Edge Publishing and Backend Ingestion

Summarized flow vectors are published to the backend through MQTT topics. The backend validates the data, normalizes feature values, stores them in TimescaleDB, and forwards them to the ML server.

VI-C3 3) Machine Learning Inference and Threat Scoring

The ML server computes:
- Anomaly score via Isolation Forest
- Classification label via Random Forest

If both suggest suspicious behavior—or if the anomaly score exceeds a predefined threshold—the backend generates a security alert.

VI-C4 4) Response, Alerting, and Device Blocking

When an anomaly is confirmed:
- The backend issues a real-time alert through the SSE
- A structured alert is delivered to the Flutter dashboard.
- The user may optionally request immediate device blocking from the UI.
- The backend then transmits a block command to the OpenWrt gateway, which enforces isolation via firewall rules.

All alerts and actions are logged for later forensic analysis, and periodic feedback can be incorporated into future model updates.

VI-D Directory Organization

Table V: Directory Organization–Frontend (SAFEON-FE, Part 1)

Directory	*File / Folder Name*	*Modules used*
SAFEON-FE	analysis_options.yaml pubspec.yaml pubspec.lock README.md safeonapp.iml android/ ios/ lib/ assets/ linux/ macos/ web/ windows/	flutter cupertino_icons http intl flutter_local_notifications fl_chart characters
SAFEON-FE /lib	main.dart models/ screens/ services/ theme/ widgets/	flutter material.dart http intl flutter_local_notifications fl_chart characters
SAFEON-FE /lib/models	alert.dart daily_anomaly_count.dart dashboard_overview.dart device_traffic_point.dart device.dart user_profile.dart user_session.dart	dart:core dart:convert collection
SAFEON-FE /lib/screens	alert_detail_screen.dart dashboard_screen.dart device_detail_screen.dart login_screen.dart onboarding_screen.dart profile_edit_screen.dart	flutter material.dart intl fl_chart app_theme.dart safeon_api.dart
SAFEON-FE /lib/services	notification_service.dart safeon_api.dart	http flutter_local_notifications dart:async dart:convert
SAFEON-FE /lib/widgets	alert_tile.dart device_card.dart section_header.dart security_graph_card.dart stat_card.dart status_chip.dart	flutter material.dart fl_chart intl app_theme.dart
SAFEON-FE /lib/theme	app_theme.dart	flutter material.dart

Table VI: Directory Organization–Frontend (SAFEON-FE, Part 2)

Directory	*File / Folder Name*	*Modules used*
SAFEON-FE /assets	logo.svg workflow.png	Flutter asset system
SAFEON-FE /android	(Android native project files)	Flutter engine AndroidX Gradle
SAFEON-FE /ios	Podfile Runner/ Flutter/	Flutter engine CocoaPods UIKit Foundation
SAFEON-FE /linux	(Linux runner files)	Flutter Linux embedding GTK CMake
SAFEON-FE /macos	(Mac runner files)	Flutter macOS embedding AppKit CocoaPods
SAFEON-FE /windows	(Windows runner files)	Flutter Windows embedding Win32 API CMake
SAFEON-FE /web	favicon.png icons/ index.html manifest.json	Flutter web runtime HTML PWA manifest

Table VII: Directory Organization–Backend (SAFEON-BE, Part 1)

Directory	*File / Folder Name*	*Modules used*
SAFEON-BE	build.gradle settings.gradle gradle.properties gradlew gradlew.bat gradle/wrapper/gradle-wrapper.jar gradle/wrapper/gradle-wrapper.properties src/ bin/ safeon.session.sql safeon-backend-ml-tcp1921681231621883 safeon-backend-router-tcp1921681231621883 README.md	spring-boot-starter-web spring-boot-starter-data-jpa spring-boot-starter-security spring-boot-starter-validation spring-boot-starter-websocket springdoc-openapi-starter-webmvc-ui org.eclipse.paho.client.mqttv3 jjwt-api jjwt-impl jjwt-jackson lombok postgresql

Table VIII: Directory Organization–Backend (SAFEON-BE, Part 2)

Directory	*File / Folder Name*	*Modules used*
SAFEON-BE /src/main/java/config	MlMqttProperties.java RouterMqttProperties.java SecurityConfig.java SwaggerConfig.java TaskSchedulerConfig.java WebSocketConfig.java	spring-boot-starter-web spring-boot-starter-security spring-boot-starter-websocket spring-boot-starter-validation springdoc-openapi-starter-webmvc-ui org.eclipse.paho.client.mqttv3
SAFEON-BE /src/main/java/controller	AlertController.java AuthController.java DashboardController.java DeviceController.java MyPageController.java	spring-boot-starter-web spring-boot-starter-security spring-boot-starter-validation
SAFEON-BE /src/main/java/domain	Alert.java AnomalyScore.java Device.java PacketMeta.java TwinResidual.java User.java UserAlert.java UserDevice.java	spring-boot-starter-data-jpa jakarta.persistence-api lombok
SAFEON-BE /src/main/java/dto	ApiResponseDto.java SimpleMessageResponseDto.java alert/AlertCreateRequestDto.java alert/AlertResponseDto.java dashboard/DailyAnomalyCountDto.java dashboard/DailyAnomalyCountResponseDto.java dashboard/DashboardDeviceDto.java dashboard/DashboardDeviceListResponseDto.java dashboard/DashboardOverviewDto.java dashboard/RecentAlertListResponseDto.java device/DeviceBlockRequestDto.java device/DeviceDiscoveryRequestDto.java device/DeviceResponseDto.java device/DeviceTrafficPointDto.java user/JwtResponseDto.java user/LoginRequestDto.java user/SignUpRequestDto.java user/UserResponseDto.java user/UserUpdateRequestDto.java	spring-boot-starter-web spring-boot-starter-validation lombok
SAFEON-BE /src/main/java/mqtt	MlMqttClientService.java MlRequestPublisher.java MlResultMqttListener.java MqttPacketListener.java RouterMqttClientService.java RouterMqttPacketListener.java	org.eclipse.paho.client.mqttv3 spring-boot-starter spring-boot-starter-web spring-boot-starter-websocket
SAFEON-BE /src/main/java/repository	AlertRepository.java AnomalyScoreRepository.java DeviceRepository.java PacketMetaRepository.java UserAlertRepository.java UserDeviceRepository.java UserRepository.java	spring-boot-starter-data-jpa postgresql
SAFEON-BE /src/main/java/security	AuthContext.java AuthenticatedUser.java JwtAuthenticationFilter.java JwtTokenProvider.java	spring-boot-starter-security jjwt-api jjwt-impl jjwt-jackson

Table IX: Directory Organization–Backend (SAFEON-BE, Part 1)

Directory	*File / Folder Name*	*Modules used*
SAFEON-BE /src/main/java/service	AlertService.java AuthService.java DashboardService.java DeviceService.java DeviceTrafficService.java MyPageService.java	spring-boot-starter-web spring-boot-starter-data-jpa spring-boot-starter-security spring-boot-starter-validation org.eclipse.paho.client.mqttv3
SAFEON-BE /src/main/java/util	UuidParser.java	java.util java.util.UUID
SAFEON-BE /src/main/java/websocket	DeviceTrafficWebSocketHandler.java JwtHandshakeInterceptor.java	spring-boot-starter-websocket spring-boot-starter-security
SAFEON-BE /src/main/resources	application.yml	spring-boot-autoconfigure spring-boot-starter
SAFEON-BE /src/test/java	DemoApplicationTests.java	spring-boot-starter-test

Table X: Directory Organization–ML Service

Directory	***File / Folder Name***	***Modules used***
SAFEON-ML	requirements.txt README.md	fastapi uvicorn pydantic numpy pandas scikit-learn torch joblib paho-mqtt
SAFEON-ML /app	__init__.py main.py model.py mqtt_bridge.py train.py	fastapi pydantic numpy pandas torch joblib scikit-learn paho-mqtt dotenv uvicorn
SAFEON-ML /models	autoencoder.pth transformer_ae.pth isolation_forest.pkl rf_model.pkl clf_model.pkl scaler.pkl enc_src_ip.pkl enc_dst_ip.pkl enc_proto.pkl meta.json	torch scikit-learn joblib json
SAFEON-ML /app/main.py	FastAPI entrypoint routing and /predict endpoint	fastapi pydantic uvicorn
SAFEON-ML /app/model.py	ModelService for inference and anomaly scoring	numpy pandas torch joblib scikit-learn logging
SAFEON-ML /app/mqtt_bridge.py	MQTT ingestion and publish	paho-mqtt json dotenv logging
SAFEON-ML /app/train.py	Training script for anomaly detection models	numpy pandas torch scikit-learn joblib

VII. UseCase

VII-A Entry

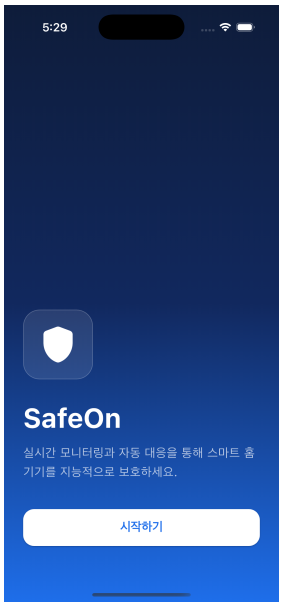

Figure 42: Entry

The Entry page briefly introduces SafeOn and provides a starting point for users before authentication. By pressing the “Start” button, users proceed to either sign up for a new account or log in with an existing one.

VII-B Sign Up

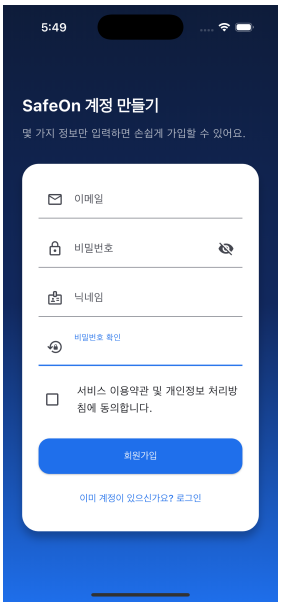

Figure 43: Signup page

Users can create an account by entering their email address, password, nickname, and agreeing to the privacy policy. After completing the form, they are redirected to the login screen where they can access SafeOn services.

VII-C Login

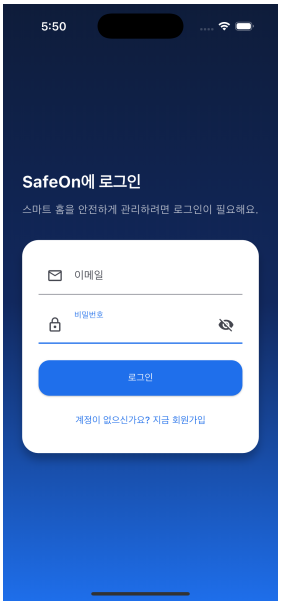

Figure 44: Login page

The Login page allows users to sign in by verifying their email and password. Once authenticated, users gain access to the main

dashboard and system features.

VII-D Main Page

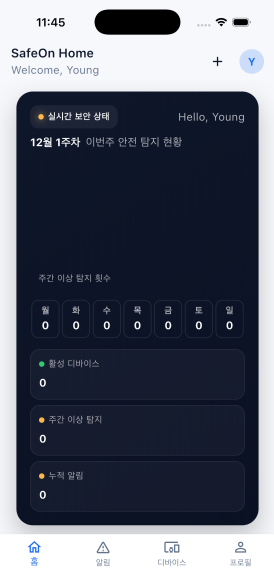

Figure 45: Main DashBoard

The Main Page summarizes the system's security status, including total device count, network health, cumulative alerts, and recent alert activity. These indicators help users quickly understand the overall condition of their smart-home environment.

VII-E IoT Device Discovery & Claiming

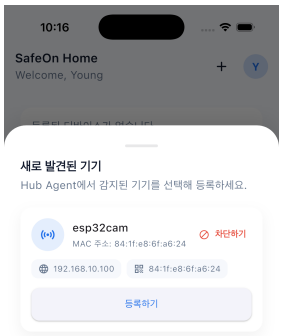

Figure 46: New device discovery popup

When a new device joins the network, SafeOn automatically displays a discovery popup containing its MAC and IP information. Users may register the device to claim ownership or block the device immediately to prevent access.

VII-F Real-Time Device Traffic Monitoring

Figure 47: Device detail page with traffic graphs

Users can view detailed information for each registered device, including its IP address, MAC address, registration time, and serial number. Real-time PPS and BPS graphs show traffic behavior during the last 15 minutes, allowing users to detect abnormal patterns.

VII-G Alert Center

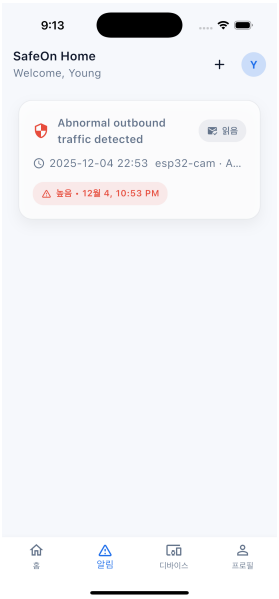

Figure 48: Alert list (filled)

The Alert Center displays a chronological list of detected anomalies. Each alert card includes severity, timestamp, and device information. Unread alerts are visually marked until the user views them.

VII-H Alert Detail

Figure 49: Alert detail page

Selecting an alert opens a detailed view containing risk level, MAC address, device name, alert status, and detection time.

VII-I Device Blocking

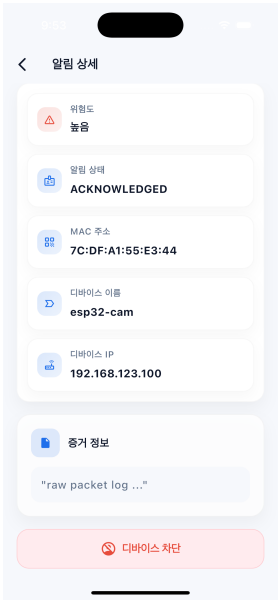

Figure 50: Device blocking workflow

Users may block a suspicious device directly from the alert detail page. Once confirmed, SafeOn updates firewall rules on the OpenWrt gateway, immediately preventing the device from accessing the network.

VII-J My Page

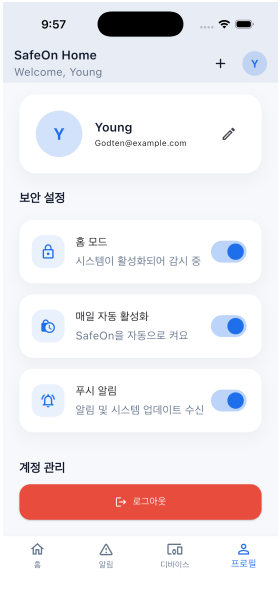

Figure 51: My Page

The My Page screen displays the user's profile information and provides access to security settings. Users can modify their nickname, update

their password, and adjust system preferences.

VII-K Logout

Figure 52: Logout

Users can safely terminate their session by selecting “Logout” from My Page. After logging out, the application returns to the Entry page.

VIII. Evaluation

The evaluation of SafeOn focuses on validating the feasibility of an edge-first EDR architecture within smart-home IoT environments. Since the primary goal of this work was to design a complete end-to-end pipeline—from packet acquisition to anomaly detection and user-facing alert delivery—the assessment emphasizes functional correctness and qualitative performance rather than large-scale quantitative benchmarking.

First, the data collection pipeline using ESP32-CAM devices successfully produced approximately 1,000 flow samples representing typical smart-home behaviors such as video streaming, motion-triggered bursts, HTTP/API communication, and bidirectional control traffic. These data were processed through the feature engineering module, demonstrating stable extraction of features including packet intervals, burst indicators, and inbound/outbound ratios.

Second, the analytical pipeline integrating Isolation Forest, LSTM Autoencoder, and Digital Twin residual scoring was tested in a controlled environment. Each model generated anomaly predictions that were consistently routed through the backend and displayed on the dashboard in near real time. The overall inference latency remained within a practical range for a Raspberry Pi-class device, confirming the viability of local, on-edge inference.

Finally, the front-end dashboard verified that SafeOn can effectively visualize traffic patterns, deliver anomaly alerts, and support device-level management. Overall, the results demonstrate the technical feasibility and usability of the proposed system, though broader quantitative evaluation remains part of future work.

IX. Limitations and Future Work

Despite completing a fully operational prototype, SafeOn has several limitations that inform future research directions.

First, the dataset size is relatively small (approximately 1,000 samples) and lacks diversity in device types, behavioral contexts, and attack scenarios. This restricts the generalizability of the anomaly detection models, particularly the LSTM Autoencoder and Digital Twin residual predictor. Future work will expand the dataset to include a wider range of IoT devices and simulated attack vectors.

Second, the ESP32-CAM traffic used for modeling exhibits irregular frame rates and bursty patterns, which reduces the expressiveness of the current feature set. Incorporating additional traffic sources such as smart speakers, environmental sensors, and home appliances may provide richer behavioral signatures.

Third, real-time local inference introduces latency constraints on the Raspberry Pi. While current performance is acceptable, scaling the system to accommodate multiple devices or continuous video workloads may require model optimization, quantization, or leveraging lightweight edge accelerators.

In addition, the system largely relies on IP- and flow-level metadata, which is effective for lightweight anomaly detection but insufficient for fine-grained behavioral understanding. As a major extension, we propose integrating a **Digital Twin** framework capable of simulating

expected device states and network behavior. A more advanced Digital Twin would enable predictive validation, richer context-aware anomaly scoring, and proactive defense mechanisms by comparing real-time telemetry with simulated ideal trajectories.

Finally, future work will also explore encrypted traffic fingerprinting, federated learning for privacy-preserving model updates, and improved alert interpretation to support a broader deployment of SafeOn in heterogeneous smart-home environments.

X. Discussion

X-A Technical Difficulties

The development of SafeOn posed several technical challenges. The ESP32-CAM produced highly inconsistent traffic patterns, including fluctuating frame intervals and motion-triggered bursts, making feature extraction non-trivial. Constructing a unified pipeline—comprising PCAP capture, feature engineering, ML inference, MQTT messaging, and backend aggregation—required careful synchronization to avoid packet loss and timestamp misalignment.

Real-time inference on constrained hardware also introduced performance difficulties, as the Raspberry Pi needed to handle both continuous packet ingestion and anomaly scoring concurrently. Additionally, the MQTT-based communication between the ML service and backend required strict topic structuring and retry logic to ensure message integrity. Visualizing anomaly scores in a way that users could interpret meaningfully further required iterative UI refinement.

X-B Non-Technical Difficulties

The project required continuous coordination among team members working on backend, AI, and frontend components. Aligning design decisions, resolving merge conflicts, and maintaining consistent API contracts occasionally caused delays. Time constraints, especially when balancing academic schedules, also made it challenging to refine experimental features or explore alternative modeling approaches.

X-C Conclusion

Despite these challenges, SafeOn successfully demonstrates the feasibility of an edge-first anomaly detection framework for smart-home IoT systems. The project highlighted the importance of modular system design, efficient communication between components, and lightweight inference on constrained hardware. The experience gained—both technically and collaboratively—establishes a strong foundation for future enhancements, including more diverse datasets, optimized models, and an expanded Digital Twin-based predictive monitoring architecture.

References

[1] "OpenWrt Documentation," https://openwrt.org/docs/start, 2024.

[2] "ESP32-CAM AI-Thinker Module," Espressif Systems, https://www.espressif.com/en/products/modules/esp32, 2024.

[3] "OASIS MQTT Version 3.1.1 Standard," https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/, 2024.

[4] "Eclipse Paho MQTT Client," https://www.eclipse.org/paho/, 2024.

[5] "FastAPI," https://fastapi.tiangolo.com/, 2024.

[6] "PyTorch," https://pytorch.org/, 2024.

[7] "scikit-learn Machine Learning Library," https://scikit-learn.org/stable/, 2024.

[8] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *IEEE International Conference on Data Mining*, 2008.

[9] "PostgreSQL Documentation," https://www.postgresql.org/docs/, 2024.

[10] "Spring Boot Reference Documentation," https://spring.io/projects/spring-boot, 2024.

[11] "Flutter Framework," https://flutter.dev/, 2024.