SafrOn: An Anomaly Detection Framework for Smart Home IoT Security

* LG SmartHome IoT Security Research Initiative

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

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

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

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

Abstract

Abstract—The Internet of Things (IoT) has come a long way with the development of home cameras, smart locks, and household appliances. Although these devices increasingly improve convenience, they also create broad attack surfaces and jeopardize the privacy of individual households. Traditional security methods are often insufficient to detect threats that manifest as unusual behavior rather than known vulnerabilities. In this work, we propose a smart home security framework based on an Endpoint Detection and Response (EDR) approach integrating multimodal data, rule-based detection, and machine learning techniques such as IsolationForest for instant anomalies and LSTM for temporal behavioral analysis. A Digital Twin simulates expected device behavior for comparative monitoring. The system provides real-time alerts, logs forensic evidence, and demonstrates a practical, privacy preserving approach to safeguarding IoT devices in everyday home environments. The framework not only provides real time alerts and proactive threat detection but also logs detailed forensic evidence to facilitate post incident analysis and accountability. Experimental evaluations demonstrate the effectiveness of the system in identifying both known and previously unseen anomalies while maintaining privacy-preserving operations suitable for deployment in everyday home environments. This work highlights a practical, scalable, and privacy-conscious approach to securing IoT devices against increasingly sophisticated cyber threats.

Index Terms

Smart Home Security, IoT, Endpoint Detection and Response (EDR), Isolation-Forest, LSTM Autoencoder, Digital Twin, 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 cameras, smart locks, vacuum robots, and household appliances. These smart devices improve convenience, automation, and remote control, allowing users to manage their homes through mobile apps or voice assistants. However, this hyper-connectivity also expands the attack surface, introducing significant privacy and security risks. A single compromised IoT device can serve as a gateway for lateral movement across the network, allowling attackers to gain unauthorized access, conduct surveillance, or exfiltrate sensitive personal information. Unlike traditional IT assets, most consumer IoT devices are resource-constrained and lack basic protection mechanisms such as encryption, authentication, or secure firmware updates.

Empirical studies illustrate the magnitude of these vulnerabilities: in one of the largest global analyses, researchers scanned 16 million households and over 83 million IoT devices, uncovering that many devices remain insecure: supporting open services, weak default credentials, and outdated protocols. A complementary crowd-sourced measurement study found that a significant number of smart home devices still use weak TLS versions or advertise weak cipher suites. Moreover, the rise of generative AI and automated exploitation tools has lowered the barrier for cybercriminals to launch complex, adaptive attacks targeting smart home environments. Therefore, there is a pressing need for an intelligent self-learning defense framework capable of analyzing device behavior directly at the edge - without relying solely on centralized cloud monitoring or static signatures.

The objective of this software is to detect, analyze, and respond to security anomalies in interconnected home devices while preserving data privacy and maintaining lightweight performance suitable for consumer environments. The system integrates multiple modern approaches, including anomaly detection algorithms, behavioral learning, and digital twin simulation, to deliver proactive defense capabilities against evolving IoT threats.

I-B Problem Statement

Traditional security approaches, such as signature-based antivirus or firewall systems, are insufficient for detecting novel threats that manifest themselves as subtle behavioral anomalies rather than known attack signatures. Unlike desktops or servers, IoT devices rarely support full-fledged endpoint protection software. Their heterogeneous nature—different operating systems, communication protocols, and limited processing capabilities—makes unified security enforcement extremely difficult. Furthermore, centralized cloud-based monitoring introduces latency, dependency, and privacy concerns, as sensitive device data must be transmitted to third-party infrastructures for analysis. This model is unsustainable for large-scale consumer IoT deployments, where trust boundaries and data ownership are critical.

Another core challenge lies in the changing nature of cyberattacks. Modern adversaries increasingly use stealth techniques that mimic normal device behavior, slowly altering network patterns or response times to avoid detection. Such threats cannot be identified by static rules alone—they require continuous, time-aware behavioral learning.

To address these challenges, this research proposes a privacy-preserving, edge-oriented security framework designed to operate as an Endpoint Detection and Response (EDR) agent for smart homes.

The system combines multimodal telemetry, rule-based heuristics, and hybrid machine learning techniques, including:

- **Isolation Forest** for real-time, unsupervised anomaly detection,
- **LSTM Autoencoder** for temporal behavior analysis,
- **Digital Twin Simulation** for modeling expected device behavior and detecting deviations, and

This hybrid design enables local inference and global intelligence without exposing raw user data. It ensures that home devices remain autonomous, yet secure, adapting dynamically to evolving network behaviors.

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 enhance the security and resilience of smart-home ecosystems. It functions as a lightweight endpoint detection and response (EDR) agent that integrates a Digital Twin simulation engine to model the expected behavior of connected devices. Deployed on an edge gateway such as an OpenWrt-based Raspberry Pi, SafeOn continuously collects and analyzes telemetry data from household IoT devices including cameras, smart locks, air purifiers, and sensors. Its hybrid anomaly detection core combines Isolation Forest for spatial anomaly identification and LSTM Autoencoder for temporal behavior analysis. By comparing real-world operational data against the digital twin's simulated baseline, SafeOn detects both instantaneous and time-dependent deviations without relying on external cloud services. This architecture ensures low latency, local decision-making, and strict privacy preservation while maintaining high detection accuracy for dynamic IoT environments.

II-B Assets

SafeOn protects a range of digital and operational assets within the home IoT network. The key asset categories are as follows:

- **Data Assets**: Device telemetry logs, network activity traces, and digital-twin simulation outputs that describe the operational state of appliances.
- **Functional Assets**: The SafeOn anomaly detection modules, EDR agent logic, and the digital twin simulator that collectively perform detection and response actions.
- **Control Assets**: Configuration files, local rule sets, alert policies, and automated mitigation scripts that determine how SafeOn responds to detected threats.

- **Privacy Assets**: User behavioral information implicitly reflected in sensor readings, device usage timelines, and environmental context data.

Each of these asset categories is considered critical for maintaining the confidentiality, integrity, and reliability of the smart-home security environment.

II-C Adversarial Model

The SafeOn threat model considers three main categories of adversaries:

- **External Adversaries (WAN-side)**: Attackers located outside the local network who attempt to exploit open interfaces, outdated communication protocols, or weak authentication mechanisms. Their goals may include network reconnaissance, data exfiltration, or denial-of-service attacks.
- **Internal Adversaries (LAN-side)**: Compromised IoT devices or malicious insiders within the home network. Such attackers can mimic normal device behavior, inject spoofed telemetry, or interfere with other devices to evade anomaly detection.
- **Firmware-level Adversaries**: Entities capable of tampering with device firmware or model update channels. These attackers may attempt to replace anomaly detection modules, alter digital twin parameters, or introduce backdoors during firmware updates.

This adversarial landscape reflects the transition from purely external network attacks toward more subtle behavioral and firmware-level manipulations typical of modern IoT environments.

II-D Attack Vectors

Within this model, potential attack vectors are classified into five categories.

- **Network-level Attacks**: Unauthorized access attempts, port scanning, and denial-of-service (DoS) or UDP flooding that target the gateway or connected devices.
- **Telemetry-level Attacks**: Injection or modification of device sensor data to imitate normal patterns and conceal malicious activity.
- **Behavioral-level Attacks**: Manipulation of timing or sequence dependencies to confuse LSTM-based detection, such as delayed commands or interleaved legitimate and malicious actions.
- **Firmware-level Attacks**: Corruption or rollback of model parameters and digital twin baselines through compromised over-the-air (OTA) updates.
- **Privacy-level Attacks**: Passive inference of user behavior or household occupancy patterns from unencrypted metadata and device communication timing.

These vectors collectively represent the layered and dynamic nature of IoT threats, where anomalies may occur at both the data and behavioral levels.

II-E Security Goals

The SafeOn framework is designed to achieve the following security objectives.

- **Confidentiality**: All telemetry data is processed and stored locally; no raw data leaves the home network.

- **Integrity**: Model files, digital twin configurations, and logs are protected using checksums and digital signatures to prevent tampering.
- **Availability**: The detection and response components continue functioning autonomously even when the gateway is disconnected from the Internet.
- **Accuracy and Robustness**: The combination of Isolation Forest and LSTM Autoencoder ensures comprehensive detection of both static and time-evolving anomalies.
- **Accountability**: Every alert, decision, and automated mitigation action is locally logged to enable transparent forensic analysis.

These goals collectively ensure that SafeOn maintains trustworthy operation while respecting the user's data privacy and autonomy.

II-F Threat–Defense Mapping

Each type of threat is mitigated through a corresponding mechanism within SafeOn's architecture:

Network-level attacks are countered by rule-based traffic filtering and connection throttling at the gateway.

Telemetry manipulation is detected through Isolation Forest analysis of multivariate device metrics.

Behavioral anomalies are identified using LSTM Autoencoder models trained on time-series device patterns.

Firmware tampering is prevented through digital signature verification and rollback protection.

Privacy inference risks are reduced by performing all processing locally and minimizing the amount of metadata retained.

II-G Discussion

Compared with traditional cloud-based or router-level intrusion prevention systems, SafeOn extends protection from packet inspection to behavioral and operational validation. Its use of Digital Twin simulation allows for contextual interpretation of anomalies — distinguishing between benign deviations and truly malicious behavior. The combination of edge-side analytics, real-time AI inference, and privacy-by-design principles enables SafeOn to serve as an autonomous, self-adaptive defense mechanism for smart-home environments.

By operating entirely on local infrastructure, it not only eliminates third-party data exposure but also establishes a scalable foundation for resilient IoT security.

III. System Architecture

III-A System Overview

The SafeOn framework is designed as an intelligent endpoint detection and response (EDR) system tailored for smart home IoT environments. Modern residential networks contain heterogeneous devices such as smart cameras, lighting systems, thermostats, and sensors that communicate continuously over wireless protocols including Wi-Fi, Zigbee, and Bluetooth Low Energy. These devices frequently depend on external cloud APIs for control and data synchronization, resulting in a widened attack surface. Limited computational capacity and weak security enforcement in consumer IoT hardware further exacerbate this vulnerability, exposing networks to potential firmware exploits, unauthorized remote access, and lateral movement attacks.

To address these challenges, SafeOn introduces an edge-level EDR layer that operates on Raspberry Pi gateways running OpenWrt. Instead of directly modifying the firmware of end devices, the system passively monitors network traffic to observe behavioral patterns. It constructs statistical and temporal representations of device communication flows and establishes a baseline of normal activity. By utilizing an Isolation Forest for anomaly detection and an LSTM-based temporal model for sequence prediction, SafeOn identifies deviations that indicate abnormal or malicious behavior in real time.

Complementing this detection pipeline, SafeOn employs a digital twin–based simulation environment that mirrors the operational state of connected devices. This virtual model continuously compares live data against expected behavioral profiles, allowing the framework to detect zero-day attacks and previously unseen anomalies that would otherwise bypass conventional signature-based defenses. The overall architecture integrates the following core components:

1) Edge Monitoring Layer — OpenWrt-based Raspberry Pi nodes capture and preprocess IoT network traffic, extracting relevant metadata for analysis.
2) Anomaly Detection Engine — A hybrid model combining Isolation Forest and LSTM networks evaluates behavioral deviations and assigns threat scores.
3) Digital Twin Module — A virtualized representation of each device reproduces normal operational states to validate real-world observations.
4) Cloud Dashboard and Alert System — Aggregated insights are transmitted to a centralized dashboard for visualization, reporting, and automated incident response.

Through this distributed and model-driven architecture, SafeOn achieves low-latency, real-time threat detection without requiring invasive modifications to IoT devices. By merging data-driven analytics with digital twin simulation, the framework provides a resilient and scalable approach to securing smart home ecosystems against both known and emerging threats.

III-B System Components

The SafeOn architecture comprises four primary layers: edge monitoring, data processing and inference, digital twin simulation, and management and visualization. Each layer cooperatively contributes to end-to-end anomaly detection, threat response, and system observability.

1 Edge Monitoring Layer:

This layer is deployed on OpenWrt-based Raspberry Pi gateways, which serve as the primary observation nodes within the local IoT network. Network packets are captured via lightweight agents that employ libpcap and Netfilter hooks for low-latency data extraction. The collected traffic is parsed to extract metadata such as device identifiers, communication frequency, and protocol usage. Feature vectors are generated and cached locally for short-term temporal correlation. By performing this preprocessing near the data source, network congestion and cloud dependency are minimized, ensuring real-time responsiveness.

2 Data Processing and Inference Layer:

At this stage, preprocessed feature vectors are analyzed by a hybrid anomaly detection engine combining Isolation Forest (IF) and Long Short-Term Memory (LSTM) models. The Isolation Forest algorithm captures statistical outliers in multidimensional feature spaces, effectively identifying abnormal device interactions or data bursts. In contrast, the LSTM model learns temporal dependencies, detecting sequential deviations from normal communication patterns. The ensemble of both models improves robustness by correlating spatial and temporal anomalies, thus reducing false positives.

3 Digital Twin Simulation Layer:

The digital twin module establishes a virtual counterpart of each IoT device within a controlled simulation environment. Each twin continuously synchronizes with real-time telemetry from the edge nodes, generating predictive behavioral baselines through historical learning. Discrepancies between the expected (twin-generated) and observed behaviors are quantified as deviation scores, which contribute to overall threat evaluation. This approach enables early detection of zero-day exploits, device misconfigurations, and stealthy command injections that bypass conventional detection methods.

III-C Data Flow and Detection Pipeline

The SafeOn data pipeline is designed for continuous monitoring, adaptive learning, and responsive mitigation within smart home IoT networks.

1 Traffic Capture and Feature Extraction

The edge node continuously captures packet streams from connected IoT devices. Traffic is filtered to exclude benign background processes and reduced into statistical and temporal features such as packet intervals, payload size variance, and destination entropy. Each feature vector represents a time-windowed snapshot of device behavior.

2 Local Inference and Edge Scoring

The extracted features are passed through the embedded Isolation Forest model to obtain a local anomaly score. Only summarized metadata and abnormal traces are forwarded to the central inference layer, thereby reducing bandwidth usage and maintaining user privacy.

3 Centralized Correlation and Twin Validation

The cloud-level LSTM model aggregates input from multiple edge nodes, reconstructing multi-device interaction patterns. Detected anomalies are crossreferenced against their corresponding digital twins to validate whether deviations represent genuine threats or benign anomalies. This dual verification mechanism enhances detection precision.

4 Alert Generation and Feedback Loop

Once an event surpasses the defined threat threshold, the system generates a structured alert containing device identifiers, timestamps, and anomaly metrics. Alerts are logged in the dashboard and optionally relayed to external systems for automated containment (e.g., network isolation or access control policy updates). Feedback from analysts or user responses is incorporated into periodic model retraining to sustain long-term adaptability.

III-D AI model

To create AI services that detect abnormal behaviors of various IoT devices in a smart home environment in real time , *SafeOn* has divided its service into three components:

1 Dataset Creation for IoT Anomaly Detection

To learn various security threats that can arise in smart home environments, an additional learning dataset specialized for LG Electronics' home appliance environment was constructed based on the open dataset ToN-IoT (Telemetry and Network IoT Dataset). ToN-IoT includes sensor telemetry logs and network traffic collected from real IoT devices (temperature, illumination, power, vibration sensors, etc.), with normal and abnormal (DoS, Spooping, Reconnaissance, etc.) states labeled. Based on this dataset, we created an additional 10,000 synthetic data that simulates the usage patterns and operation logs of LG home appliances (refrigerators, washing machines, air purifiers, robot vacuum cleaners, etc.) using Generative AI to increase real-world applicability. The generated data reflects factors such as power consumption, sensor change rate, operating cycle, usage time zone, and network call pattern with reference to actual LG ThinQ platform operation scenarios, and is designed to include normal/abnormal status in a balanced manner based on actual user behavior patterns.
The final hybrid dataset was built through the following steps:

1) Refining the ToN-IoT Source Log and Sorting Time Series
2) The sliding window produces 30 seconds of continuous input
3) Normal/Abnormal Ratio Readjustment (SMOTE)
4) Quality verification after full integration with 10,000 additional generated AI-based synthetic data
5) Structuring federated learning applications with device-specific sharding via IoT devices

This process completes a hybrid IoT security dataset that reflects the generalized pattern of open datasets and the realistic characteristics of LG home appliance environments at the same time. This ensures high detection accuracy and real-time responsiveness even in certain brand/device environments.

2 Development of LSTM-based AI Model for Real-Time Detection

The core AI engine for abnormality detection is designed as a time series analysis model based on long short-term memory (LSTM). LSTM is a type of recurrent neural network (RNN), which can effectively learn long-term dependence in time series data. This enables immediate warning and response before security threats or system errors occur, and strengthens safety and reliability in smart homes.

This model continuously receives data streams collected from IoT devices and detects anomalies (abnormal power patterns, data transmission abnormalities, response delays, etc.) that deviate from the normal state in real time. The detection results can be transmitted to the internal security module and lead to immediate alerts, log records, or automated response scenarios. These structures enable real-time proactive security, not just post-analysis, and increase the safety and reliability of the entire smart home.

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

   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 programming language developed by Google, designed to provide both efficiency and readability. It combines the simplicity of dynamically typed languages with the performance benefits of statically typed ones. Dart supports multiple programming paradigms—object-oriented, imperative, and functional—allowing developers to select the most effective approach for their specific tasks.
      The language is widely appreciated for its clean and expressive syntax, making it accessible to both novice and experienced programmers. Dart's dual compilation model, which includes Just-In-Time (JIT) for rapid prototyping and Ahead-Of-Time (AOT) for optimized production builds, provides flexibility throughout the development process. Moreover, Dart's robust asynchronous programming capabilities, comprehensive standard library, and thriving ecosystem (including pub.dev) make it well-suited for building high-performance, cross-platform applications. As the foundation of the Flutter framework, Dart has become an essential tool in modern mobile and web development.

   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

FastAPI is a modern, high-performance web framework for building APIs with Python. It is designed to streamline API development by utilizing Python 3.7+ type hints, which enhance both code validation and readability. Built upon Starlette for the web layer and Pydantic for data handling, FastAPI achieves performance comparable to frameworks written in Node.js and Go.
One of FastAPI's most distinctive features is its ability to automatically generate interactive API documentation based on the OpenAPI specification, with Swagger UI and ReDoc integrated by default. The framework also offers full support for asynchronous programming through async and await, making it particularly well-suited for high-concurrency applications.
Developers benefit from built-in dependency injection, automatic data validation, and serialization, which significantly reduce boilerplate code and lower the risk of runtime errors. FastAPI's strong type checking ensures that request and response data strictly conform to defined schemas, thereby improving reliability and maintainability.
Due to its speed, simplicity, and robust feature set, FastAPI has become a preferred choice for developing RESTful APIs, deploying machine learning models, and building microservices in modern production environments.

5) Flutter

Flutter is an open-source UI framework developed by Google that enables developers to create cross-platform applications from a single codebase. It provides a rich set of pre-designed widgets and tools that facilitate the rapid development of visually appealing and highly responsive user interfaces. Flutter leverages Dart as its core language, allowing developers to benefit from Dart's fast compilation and expressive syntax while building seamless and native-like applications.
One of Flutter's most notable features is its Hot Reload capability, which allows developers to instantly view code changes without restarting the app, significantly enhancing productivity. Additionally, Flutter's layered architecture—comprising the framework, engine, and embedding layers—offers fine-grained control over UI rendering and performance optimization. The framework's growing ecosystem and extensive support for third-party packages make it highly adaptable for projects ranging from prototypes to large-scale enterprise applications. Because of its efficiency, flexibility, and elegant design system, Flutter has rapidly become one of the most popular frameworks for modern cross-platform development.

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) Library

   1) Hugging Face Transformers Library

      The Hugging Face Transformers Library is a widely adopted and powerful framework for natural language processing (NLP) tasks. It offers an extensive collection of pretrained transformer-based models, including well-known architectures such as BERT, GPT, and T5, which can perform a variety of language-related tasks such as text classification, machine translation, summarization, and question answering.

      One of the library's greatest strengths lies in its intuitive and user-friendly API, which allows developers to integrate state-of-the-art models into applications with minimal configuration. Furthermore, the library supports fine-tuning, enabling users to adapt pretrained models to domain-specific datasets and tasks, thereby achieving higher accuracy and relevance.

      The Transformers Library is also highly versatile, offering seamless compatibility with major deep learning frameworks such as PyTorch and TensorFlow. Backed by comprehensive documentation and an active open-source community, Hugging Face has become a cornerstone in both academic research and industrial applications. Its robust ecosystem continues to simplify the development and deployment of cutting-edge NLP solutions, making it a central tool in modern artificial intelligence.

   2) PyTorch

      PyTorch is an open-source machine learning and deep learning library that has gained widespread popularity among researchers and developers for its flexibility and ease of use. Built primarily for Python, PyTorch provides a dynamic computational graph, allowing developers to construct, modify, and debug neural network architectures in an intuitive and interactive manner without compromising speed or efficiency.

      The framework's extensive ecosystem of tools and libraries supports every stage of the deep learning workflow—from data preprocessing and model training to evaluation and deployment in production environments. Its seamless integration with GPU acceleration, coupled with high-level APIs, enables efficient experimentation and rapid prototyping of complex models. Moreover, PyTorch benefits from a strong and active open-source community, continuous innovation, and frequent updates that keep it at the forefront of modern deep learning research. Today, it stands as one of the most influential frameworks powering AI applications, academic research, and large-scale industry deployments across the globe.

3) Microsoft Azure Digital Twins

   Microsoft Azure Digital Twins is a comprehensive Internet of Things (IoT) platform that enables the creation of digital replicas of real-world systems, environments, and assets. By modeling physical entities and their relationships in a virtual environment, Azure Digital Twins allows organizations to simulate, monitor, and optimize complex systems in real time. This digital representation facilitates data-driven decision-making by providing a unified view of physical operations and enabling predictive insights through continuous data synchronization from IoT devices and sensors.
   The platform leverages the Azure Digital Twins Definition Language (DTDL) to define entities, interactions, and behaviors, allowing developers to construct rich and flexible digital models. Through seamless integration with other Azure services—such as Azure IoT Hub, Time Series Insights, and Azure Machine Learning—it supports large-scale data ingestion, analytics, and AI-driven optimization.
   Azure Digital Twins is particularly effective in industries such as manufacturing, smart buildings, energy management, and urban planning, where real-time system visualization and predictive maintenance are crucial. By bridging the gap between the physical and digital worlds, the platform empowers organizations to enhance operational efficiency, reduce costs, and accelerate innovation through intelligent digital simulations.

4) Cloud Platform

   1) AWS EC2 (Elastic Compute Cloud)

      Amazon Web Services (AWS) Elastic Compute Cloud (EC2) is a fundamental component of Amazon's cloud computing ecosystem, offering scalable and flexible computing resources in the cloud. It enables users to deploy and manage virtual servers—known as instances—providing secure and resizable compute capacity for running a wide range of applications and services.
      EC2 supports various instance types optimized for different workloads, including compute-optimized, memory-optimized, and general-purpose configurations. It also provides advanced features such as auto-scaling, which dynamically adjusts computing power to meet fluctuating demand, and load balancing, which efficiently distributes incoming traffic across multiple instances. Additionally, Amazon Virtual Private Cloud (VPC) integration enhances network isolation and security for cloud environments.

Operating on a pay-as-you-go pricing model, EC2 allows organizations to optimize costs by paying only for the compute capacity they consume. With its high reliability, global infrastructure, and consistent performance, AWS EC2 serves as the backbone for many cloud-based applications, enabling businesses to deploy, scale, and maintain their computing resources with exceptional efficiency and flexibility.

2) AWS SageMaker

Amazon SageMaker is a fully managed machine learning (ML) service within the Amazon Web Services (AWS) ecosystem that simplifies and accelerates the end-to-end process of building, training, and deploying ML models at scale. It provides an integrated development environment equipped with tools for data preparation, model creation, training, and evaluation, supporting widely used frameworks such as TensorFlow, PyTorch, and Scikit-learn.
SageMaker offers a comprehensive suite of advanced features, including automatic model tuning (hyperparameter optimization) and distributed training, which enhance both efficiency and performance during development. For deployment, the platform supports multiple options such as real-time inference endpoints, batch processing, and edge deployment, providing flexibility to meet diverse production needs.
By integrating seamlessly with other AWS services—such as S3 (storage), IAM (security), and AWS Data Pipeline—SageMaker enables organizations to focus on model innovation while relying on a secure, scalable, and cost-effective infrastructure. As a result, it has become one of the most widely adopted platforms for enterprise-scale machine learning operations (MLOps), bridging the gap between research and real-world AI deployment.

5) 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)

Visual Studio Code (VS Code) is a powerful, open-source source code editor developed by Microsoft, widely recognized for its flexibility, performance, and extensive language support. It provides built-in features such as an integrated terminal, debugging tools, and Git version control, making it a comprehensive development environment suitable for both individual and

collaborative projects.
One of VS Code's most notable strengths lies in its extensible architecture, supported by a rich marketplace of plugins and extensions that enable developers to customize their workflow across various programming languages and frameworks. Features like IntelliSense deliver intelligent code completion, real-time syntax checking, and context-aware suggestions, significantly enhancing coding efficiency and accuracy.
The editor's intuitive interface and highly configurable settings allow users to personalize their workspace to fit specific project needs and preferences. By combining versatility, productivity, and ease of use, Visual Studio Code has become one of the most widely adopted development tools in modern software engineering.

2) 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 productivity 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.

3) Figma

Figma serves as the primary tool for UI/UX design within our project, providing a powerful web-based platform that supports real-time collaboration among team members. It streamlines the entire design workflow—from wireframing and interface design to interactive prototyping—all within a single integrated environment.
One of Figma's key advantages lies in its design system management capabilities, which enable teams to maintain visual consistency across all components while efficiently managing reusable UI elements. Developers can easily extract CSS values, assets, and component specifications, facilitating a smooth transition from design to implementation.
Moreover, Figma's component-based design approach aligns naturally with React Native's component architecture, fostering a seamless connection between the design and development phases. By combining collaboration, flexibility, and scalability, Figma has become an indispensable tool for modern UI/UX workflows and cross-functional product teams.

4) Xcode 16.4

   Xcode 16.4 is the latest integrated development environment (IDE) from Apple, serving as a cornerstone for iOS, iPadOS, macOS, watchOS, and tvOS application development. It includes the iOS 18.5 SDK and offers a complete suite of tools for building, testing, debugging, and deploying applications across Apple's ecosystem. With its real-time iOS Simulator, developers can test applications seamlessly across multiple device types and operating system versions.
   In React Native projects, Xcode 16.4 plays a critical role in generating iOS builds, managing signing certificates, and performing platform-specific debugging to ensure smooth interoperability between native and cross-platform components. The IDE also features enhanced performance profiling, memory diagnostics, and UI testing tools that help developers optimize efficiency and reliability.
   Moreover, Xcode 16.4 improves overall developer productivity with faster build times, refined Swift and SwiftUI support, and deeper integration with TestFlight and App Store Connect for streamlined distribution. Through its robust toolchain and continuous updates, Xcode remains the essential environment for creating high-performance, production-ready iOS applications.

5) PostgreSQL

   PostgreSQL is a powerful, open-source relational database management system (RDBMS) known for its reliability, extensibility, and advanced feature set. It offers full SQL compliance along with modern capabilities such as window functions, JSON support, and geospatial data handling through PostGIS.
   With strong ACID compliance and Multiversion Concurrency Control (MVCC), PostgreSQL ensures consistent, high-performance transaction processing even under heavy workloads. It supports replication, partitioning, and high availability, making it suitable for large-scale and mission-critical systems.
   PostgreSQL's extensible architecture allows developers to define custom data types, operators, and functions, while robust security features like role-based access control and SSL/TLS encryption help safeguard data integrity. These strengths make PostgreSQL a preferred choice for modern, data-intensive applications that demand both flexibility and stability.

6) TimescaleDB

   TimescaleDB is an open-source time-series database built as an extension of PostgreSQL, designed to efficiently store, query, and analyze time-stamped data at scale. It combines the reliability and robustness of relational databases with the performance and scalability of time-series systems, making it ideal for use cases such as IoT data monitoring, financial analytics, system metrics, and sensor telemetry.
   Leveraging PostgreSQL's mature ecosystem, TimescaleDB supports full SQL compatibility, enabling developers to use familiar relational queries while

taking advantage of powerful time-series functions such as continuous aggregates, data retention policies, and hypertables for automatic data partitioning. These features allow for efficient handling of large, high-ingest datasets without compromising performance.
TimescaleDB also integrates seamlessly with Grafana, Prometheus, and other visualization or monitoring tools, providing a complete stack for real-time analytics and observability. Its focus on scalability, flexibility, and ease of integration has established TimescaleDB as a leading solution for managing temporal and high-frequency data in modern data-driven applications.

7) Discord

Discord is a widely used communication and collaboration platform originally developed for gaming communities but now adopted across diverse fields, including education, software development, and professional teamwork. It provides real-time voice, video, and text communication, enabling teams to coordinate efficiently regardless of location.
One of Discord's greatest strengths lies in its server-based architecture, which allows users to create organized spaces with multiple channels dedicated to specific topics or projects. This structure facilitates clear communication, task management, and community engagement within development teams or organizations.
Discord also offers a robust API and bot integration system, allowing developers to automate workflows, manage notifications, and integrate tools such as GitHub, Jenkins, or monitoring services directly into communication channels. Its cross-platform availability—spanning desktop, mobile, and web—ensures accessibility and continuity across devices.
By combining reliability, customization, and ease of use, Discord has evolved into a versatile collaboration environment, supporting everything from project coordination and code reviews to community management and technical discussions in real time.

8) GitHub

GitHub is a web-based Git version control and collaboration platform that enables developers to host, share, and manage code repositories efficiently. It provides essential features such as pull requests, issue tracking, and code reviews, fostering seamless collaboration and transparency within development teams.
Beyond version control, GitHub integrates powerful automation and CI/CD capabilities through GitHub Actions, allowing developers to automate build, test, and deployment workflows directly within their repositories. It also offers built-in documentation tools, project boards, and discussion spaces, enabling comprehensive project management and knowledge sharing in one platform.
GitHub supports branching and forking workflows, allowing teams to work in parallel and safely merge contributions through structured review processes. With extensive integration options for popular development tools and strong

security features such as dependency scanning and secret detection, GitHub provides a robust and secure environment for software development.
Adopted globally by both individuals and large enterprises, GitHub has become a cornerstone of modern software engineering, unifying collaboration, automation, and code management in a single ecosystem.

9) Overleaf

   Overleaf is a web-based LaTeX editing and collaboration platform designed to simplify the process of creating and managing scientific and technical documents. It enables real-time collaboration, allowing multiple users to edit the same document simultaneously while automatically synchronizing changes across all participants.
   The platform provides a full LaTeX environment with built-in compilation, eliminating the need for local installations or manual configuration. It offers features such as syntax highlighting, auto-completion, error detection, and instant PDF preview, enhancing both the accuracy and efficiency of document creation. Additionally, Overleaf includes a rich collection of templates for academic papers, reports, theses, and presentations, making it accessible for users across various disciplines.
   Overleaf also supports version control and integrates seamlessly with reference management tools like Zotero and Mendeley, streamlining the citation and bibliography workflow. With comprehensive documentation and tutorials, Overleaf has become an essential platform for academic writing, research collaboration, and professional publication, bridging the gap between technical precision and collaborative efficiency.

10) 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.

11) 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.