CS350 Safehome Project
Software Design Specification

Team 4

20180048 Gyeongyeon Kim
20190480 Wonjoon Lee
20200468 Sumin Lee
20210623 Yeongjun Joo

I. OVERVIEW

1. Introduction

This document describes the design specifications of Safehome and provides a detailed design model based on software requirements specification (Team 1's HW1; hereinafter "SRS") and HW2_safehome_design_guideline. Considerations have been made for low coupling, high cohesion, and modularization of common functionalities. The architectural structure, class diagram, CRC cards, and state diagrams are presented to illustrate the overall design of the system.

2. Goal

1. Follow the requirements and analysis model.
2. Ensure low coupling, high cohesion, and modular design.
3. Pursue efficiency, reliability, maintainability, and testability.
4. Minimize complexity while enabling reusability and flexibility.
5. Provide fault tolerance, scalability, and secure data handling.
6. Maintain consistency and traceability across all design artifacts.

3. How the Design Work Proceeded

1. Analysis and discussion of the official specification (SRS; Team 1's HW1)
2. Consideration of design details implied but not explicitly stated in HW1
3. Class design and derivation of shared modules after dividing functions by feature area
4. Functional decomposition and architectural refinement based on the subsystem architecture
5. Final design documentation and CRC card creation
6. Quantitative design evaluation through metric calculation
7. Class relationship validation and consistency check across design artifacts
8. Peer review and scenario-based verification

4. Assumption

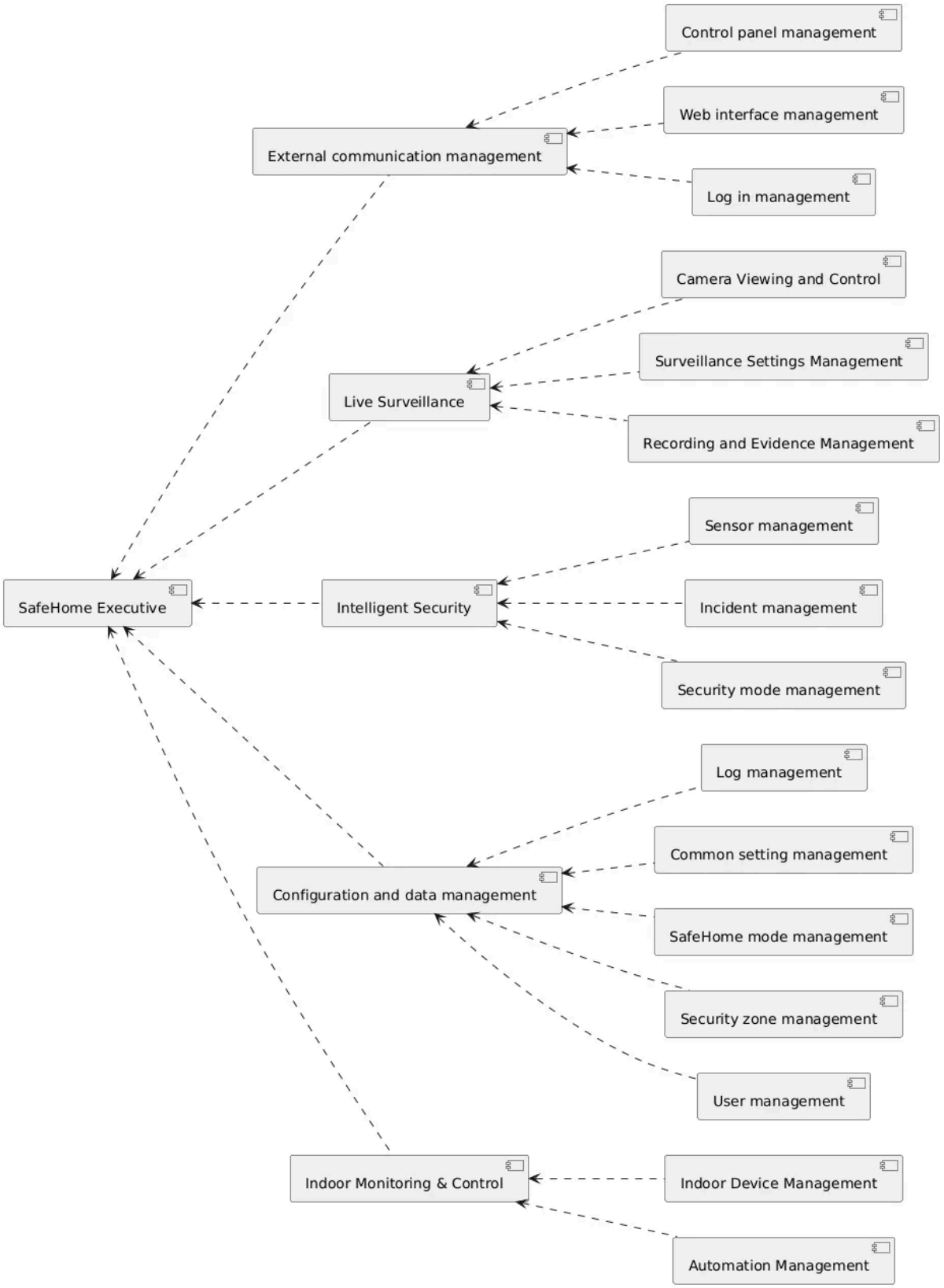
1. In section III. CLASS DIAGRAMS, for Control Panel, Web Interface only the classes are defined, while detailed attributes and methods are omitted since the specific GUI components are given which is out of scope in this specification. (Refer to Meeting Log 11/13)
2. In Chapter II. ARCHITECTURAL STRUCTURE, we described the roles of each class with a focus on their functional responsibilities (typically expressed with the suffix -management). In III. CLASS DIAGRAMS, we described the roles with a focus on the concrete class composition (commonly expressed with the suffix -manager).
3. The LogManager class both handles Activity Log(Refer to SRS Glossary p131) and Audit Log(Refer to SRS Glossary p134).
4. The ModeManager class was defined based on the One-Touch Modes (Away, Home, Sleep) (SRS UC 1.3.1) requirements to handle security mode transitions and delay/bypass logic.
5. The BypassManager class was defined based on the Sensor Bypass (SRS UC 1.3.2) requirement to implement functionality for temporarily disabling faulty sensors.

6. The BaseSensorAdapter class serves as an interface between physical sensors and the hub, and was derived from the Sensor Monitoring (SRS UC 1.1.x) requirements, including Physical Intrusion Detection (SRS UC 1.1.1) and Environmental Hazard Detection (SRS UC 1.1.2).
7. The IncidentController class was derived from the Incident Management (SRS UC 1.2.x) requirements to manage the creation and handling of incidents based on sensor events.
8. The VerificationManager class was defined based on the Alarm Verification Step (SRS UC 1.2.2) requirement to handle the verification workflow that provides visual evidence and waits for the homeowner's response.
9. The EscalationManager and DispatchService classes were derived from the Emergency Service Integration and Auto Call (SRS UC 1.2.3) requirement to process emergency alerts and external service requests upon verified alarms.
10. The NotificationService class was derived from notification-related requirements across the system, including Alarm Verification Step (SRS UC 1.2.2) and Notification Policy and Cooldown (SRS UC 2.3.2), to support sending push notifications and SMS messages to users.
11. The CameraController, SafeHomeCamera, and Camera classes were derived from Camera Viewing and Control (SRS UC 2.1.x) requirements — such as Single Camera Live View (SRS UC 2.1.1), Two-Way Audio (SRS UC 2.1.2), Protect Sensitive Camera Feed with a Password (SRS UC 2.1.3), and Camera Activation and Deactivation (SRS UC 2.1.4) — to implement real-time streaming and camera control features.
12. The RecordingManager and Recording classes were derived from the Search and Playback Recordings (SRS UC 2.2.1) and Evidence Sharing and Export (SRS UC 2.2.2) requirements to support searching, playback, and sharing of recorded video clips.
13. The ConfigurationManager, SystemSettings, and SecurityZone classes were defined based on Configure Alarm Conditions by Security Mode (SRS UC 1.2.1) and System Status Dashboard (SRS UC 3.2.1) requirements to provide persistent management of system-wide settings and security zone information.
14. The User class was defined to encapsulate user profiles and permissions, supporting Edit Profile Information (SRS UC 4.1.5) and User Role and Access Control (SRS UC 3.3.1) requirements.
15. The LogInManager and LogInInterface classes were derived from Account Management (SRS UC 4.1.x) requirements — including Log In (SRS UC 4.1.2), Log Out (SRS UC 4.1.3), Password Recovery and Reset (SRS UC 4.1.4), and Change Password (SRS UC 4.1.6) — to handle authentication and session management.
16. The IndoorDeviceRequestManager and SmartLight classes were defined based on the Indoor Device Control (SRS UC 5.1.1) requirement to handle manual device control requests.
17. The AutomationRequestManager, Automation, AirQualitySensor, and VentilationSystem classes were derived from the Indoor Air Quality Monitoring and Ventilation Integration (SRS UC 5.2.1) requirement to manage and execute automation rules.
18. The SmartMeter class was defined to support the Real-Time Power Consumption Monitoring and Reporting (SRS UC 5.2.2) requirement by representing the entity that monitors household power usage.

19. The SmartDevice class was defined as a hardware abstraction layer (HAL) to handle all device interactions in Indoor Device Control (SRS UC 5.1.1) and Indoor Monitoring System (SRS UC 5.2.x).

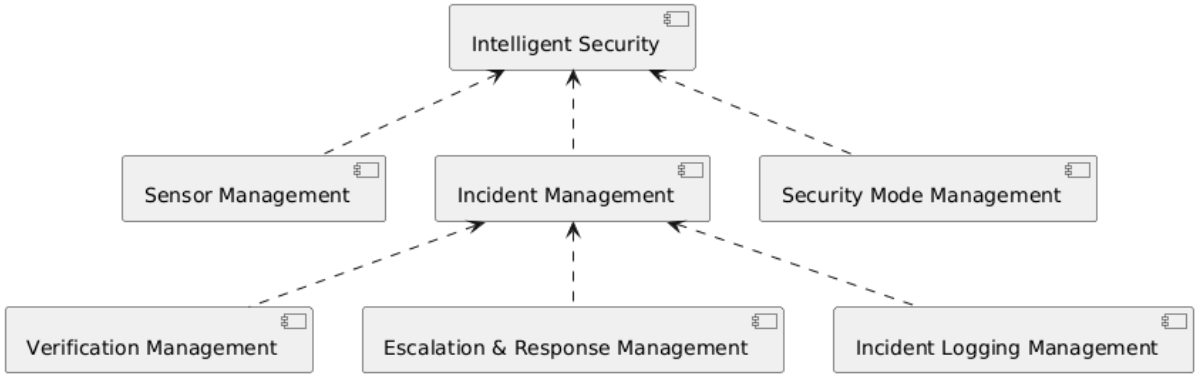
II. ARCHITECTURAL STRUCTURE

1. Overall Architecture

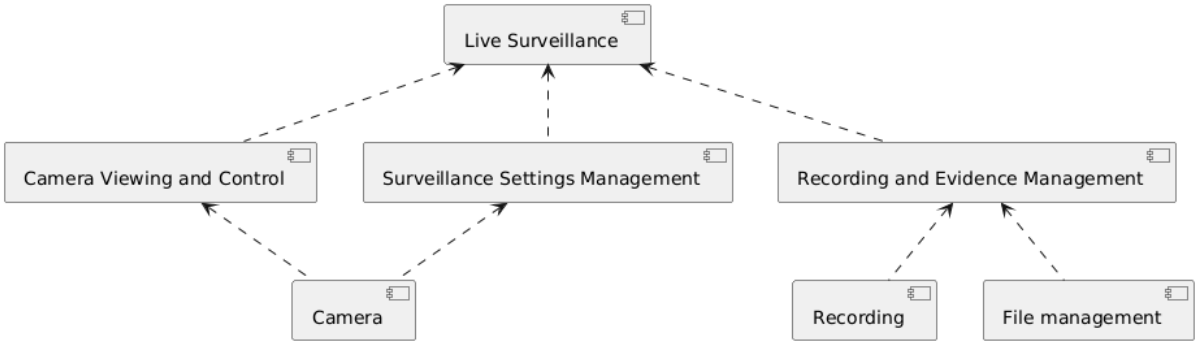


(Reference on Assumption 20)

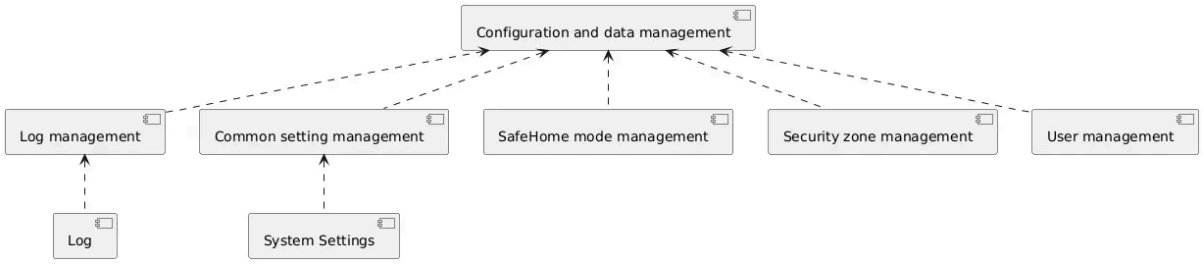
2. Intelligent Security



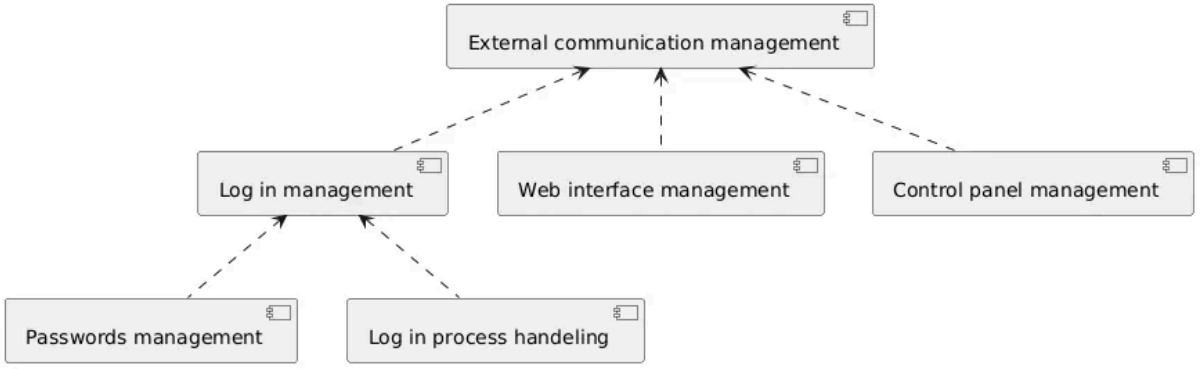
3. Live Surveillance



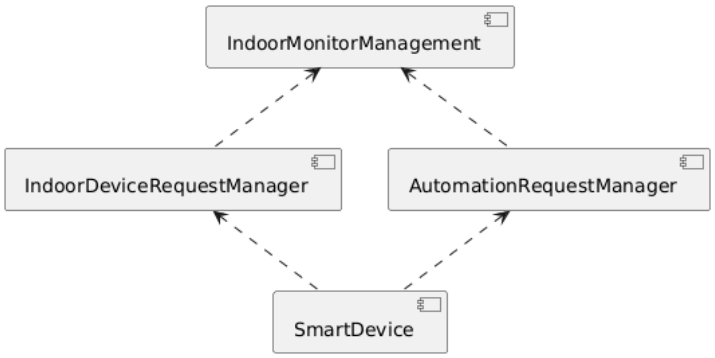
4. Configuration and data management



5. External communication management

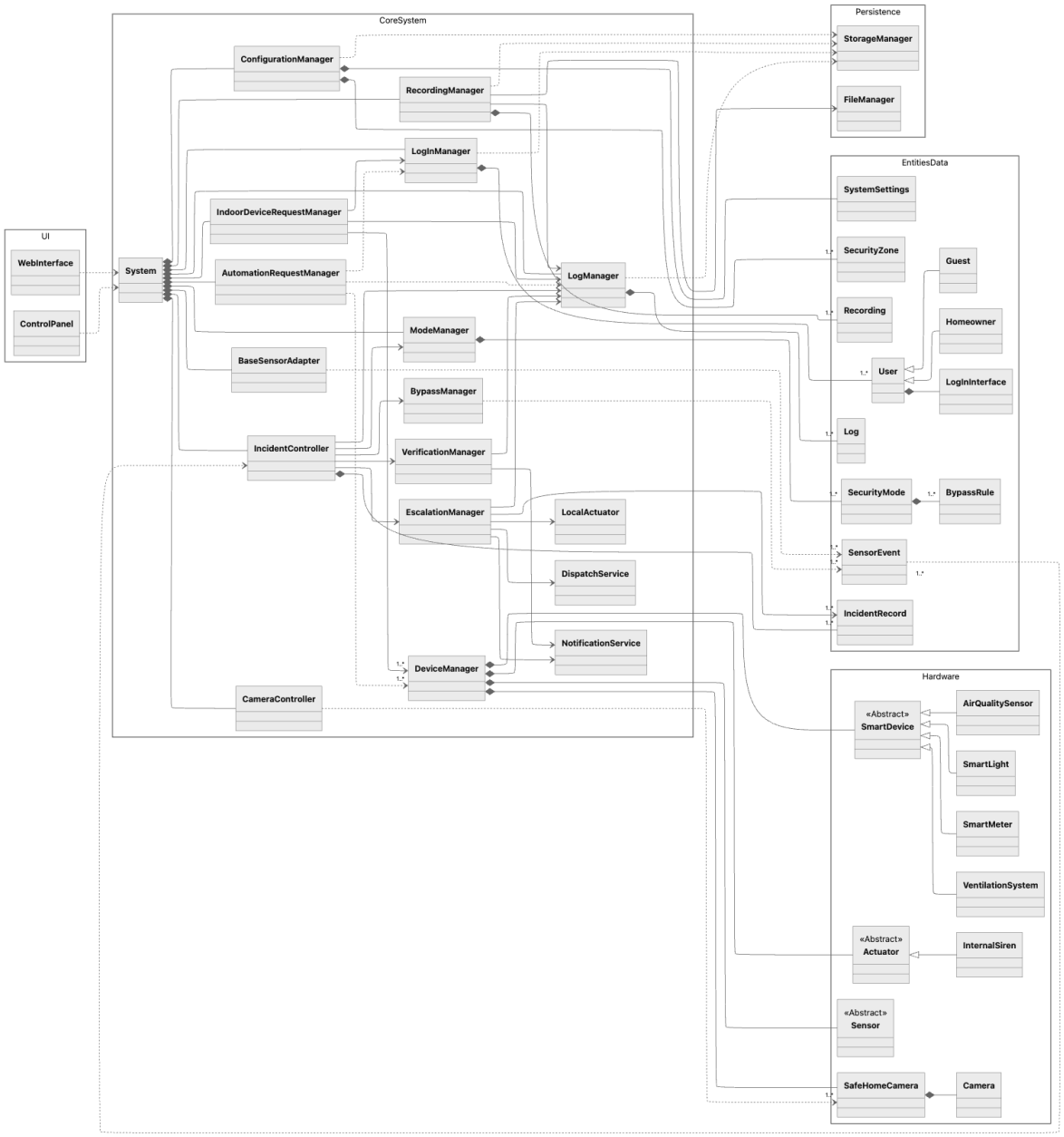


6. Indoor Monitoring and Device Control



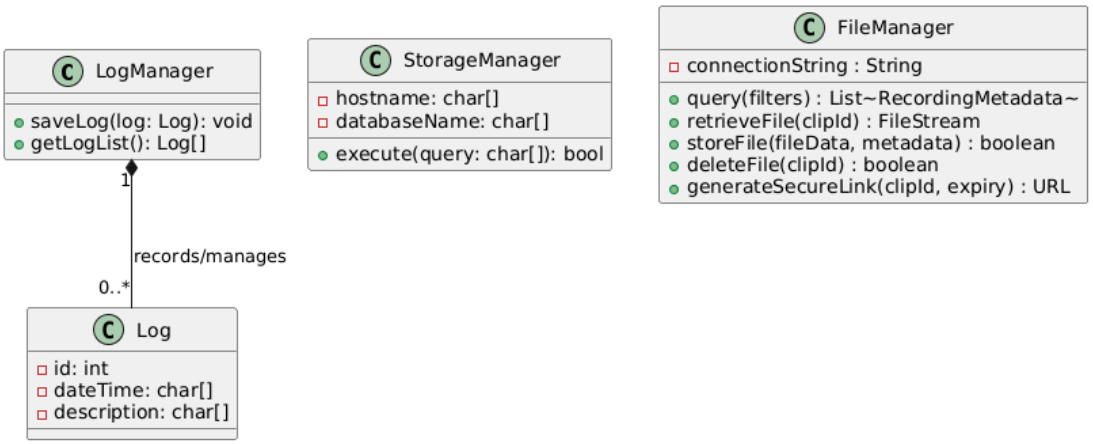
III. CLASS DIAGRAMS

1. Overview

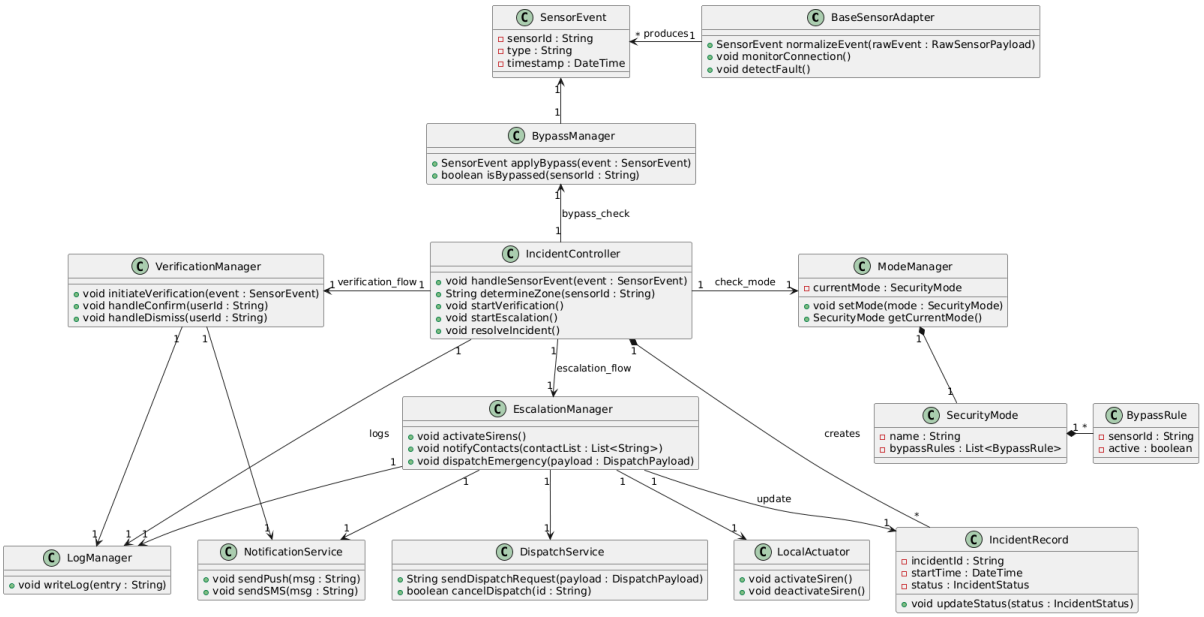


*Multiplicity values of 1 have been omitted from the diagram.

2. Infrastructure

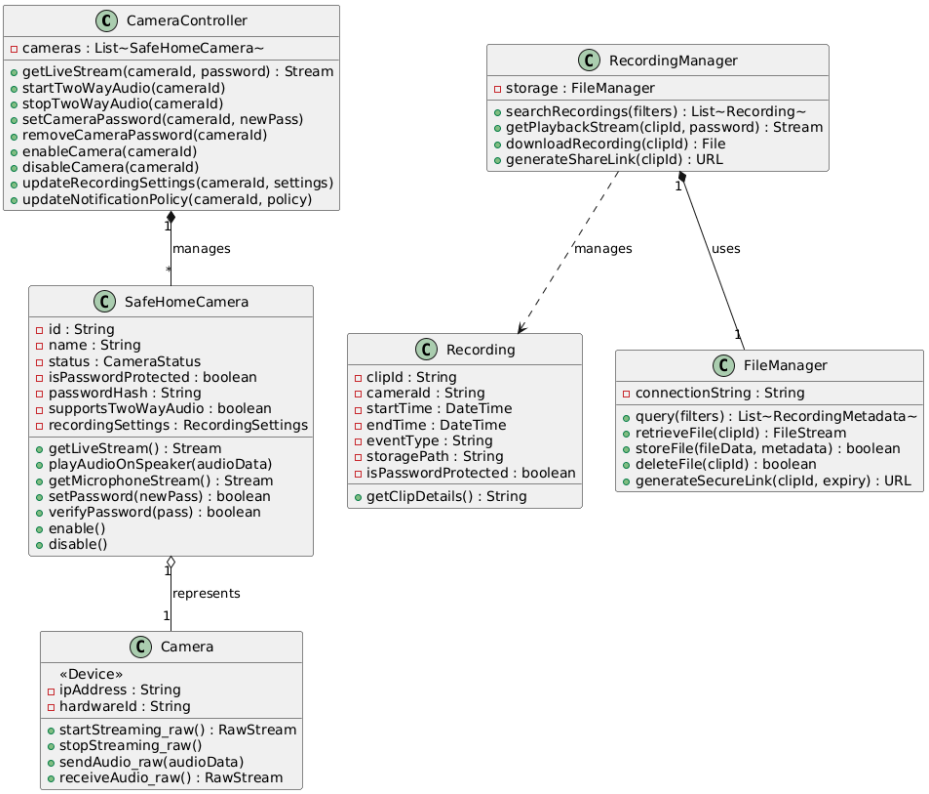


3. Intelligent Security

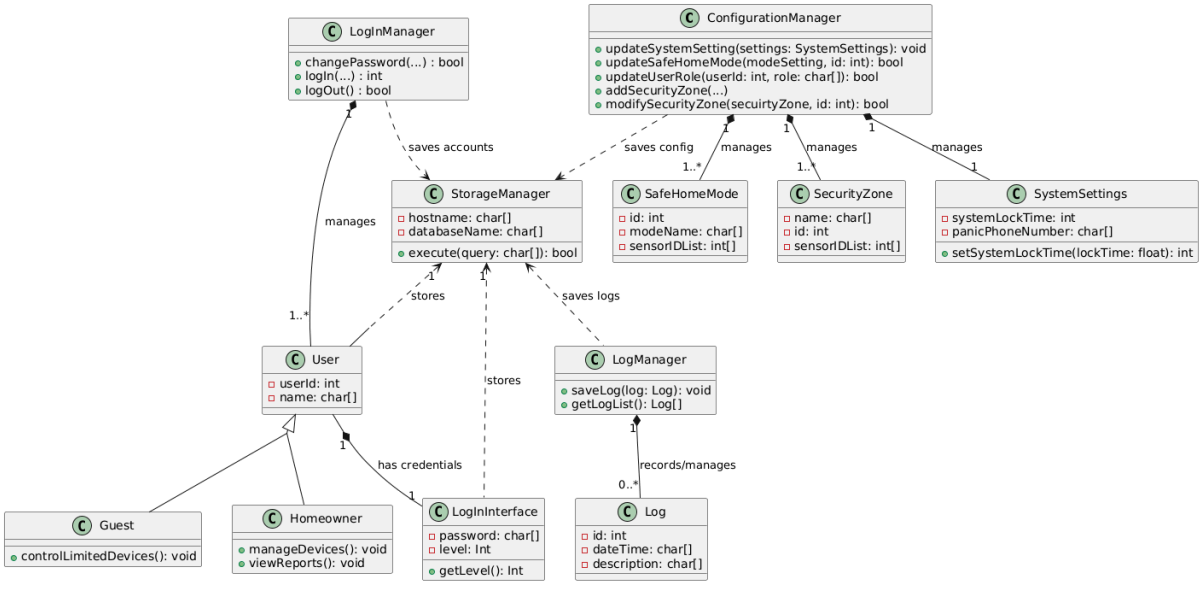


(Reference on Meeting Log 11/13 - 16:00 - 7.Discussion regarding Security Zone and ModeManager)

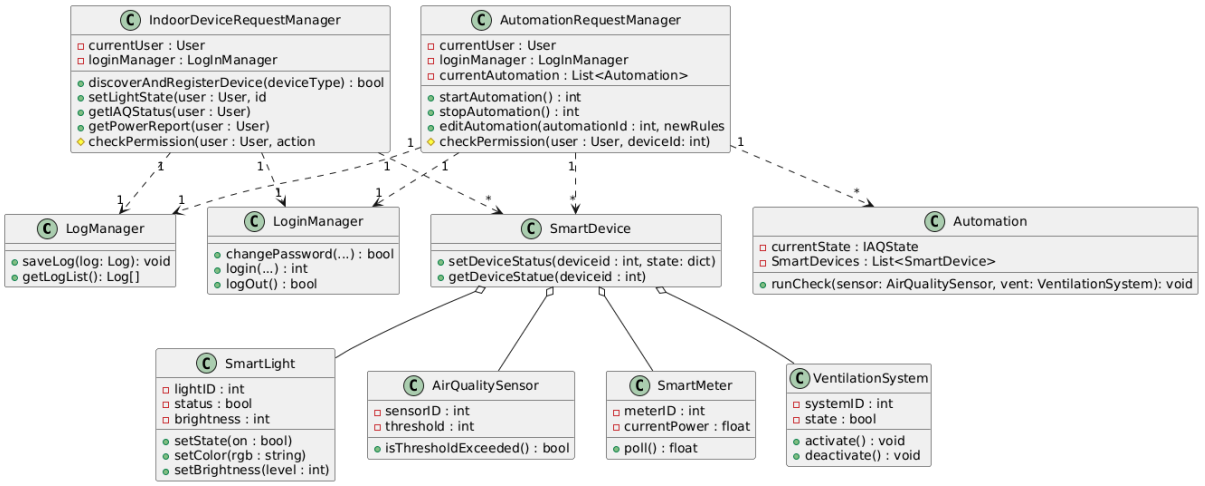
4. Live Surveillance



5. System and User Management



6. Indoor Monitoring & Device Control



IV. CRC CARDS

1. Infrastructure

Class: LogManager	
Manages the information for each log instance, handling storage and retrieval of event records.	
Responsibilities	**Collaborators**
Saves log instance	StorageManager
Gets list of log events from storage	StorageManager

Class: Log	
Data structure that holds details for a single event, such as ID, time, and event description.	
Responsibilities	**Collaborators**
Reads/writes event log ID	
Reads/writes datetime of the event	
Reads/writes description of the event	
Class: Log	
Data structure that holds details for a single event, such as ID, time, and event description.	

Class: FileManager	
An abstraction layer for the physical storage system (e.g., cloud storage, local hub database). It handles the low-level querying and retrieval of files and metadata.	
Responsibilities	**Collaborators**
Query the database for recording metadata.	RecordingManager
Retrieve a specific video file from storage.	RecordingManager
Store new video files and their metadata.	
Delete files based on retention policies.	
Generate a secure, time-limited access link for a file.	RecordingManager

Class: StorageManager	
An infrastructure class responsible for handling DB transactions. Connecting to the database and executing queries.	
Responsibilities	**Collaborators**
Manages database connection state	
Executes database read/write operations	

2. Intelligent Security

Class: BaseSensorAdapter	
A low-level abstraction layer that normalizes raw hardware sensor signals into unified SensorEvent objects. It also monitors sensor connectivity and detects potential sensor faults.	
Responsibilities	**Collaborators**
Normalize raw hardware events into SensorEvent	SensorEvent
Monitor sensor connectivity	
Detect sensor fault state	

Class: SensorEvent	
A data structure representing a normalized sensor trigger. Contains only sensor-level metadata, since zone membership is determined later by IncidentController.	
Responsibilities	**Collaborators**
Store normalized event data (sensorId, zoneId, type, timestamp)	

Class: ModeManager	
Manages the home's current security mode (e.g., Home, Away, Sleep). Provides mode lookup functions to allow IncidentController to determine how incoming events should be handled.	
Responsibilities	**Collaborators**
Set current security mode	SecurityMode
Return current mode to callers	SecurityMode
Provide access to bypass rules indirectly through SecurityMode	SecurityMode

Class: SecurityMode	
Represents a specific security mode and its associated policy behavior. Holds a set of BypassRule objects that define which sensors should be ignored while the mode is active.	
Responsibilities	**Collaborators**
Maintain active mode name (Home, Away, Sleep)	ModeManager
Hold list of BypassRule objects	BypassRule

Class: BypassRule	
Defines a single sensor's bypass condition (e.g., temporarily disabled during stay mode). Encapsulates the sensorId and whether the bypass is active.	
Responsibilities	**Collaborators**
Represent a sensor-specific bypass state	

Class: BypassManager	
Applies bypass logic to incoming SensorEvent objects. Determines whether a given event should be ignored based on the active SecurityMode's rules.	
Responsibilities	**Collaborators**
Apply bypass logic to incoming SensorEvent	SensorEvent
Check if a specific sensor is currently bypassed	BypassRule, ModeManager

Class: IncidentController	
The central coordinator for Intelligent Security. Receives SensorEvents, determines applicable zones, checks bypass/mode conditions, and routes the event to VerificationManager or EscalationManager. Manages IncidentRecord lifecycle and logging.	
Responsibilities	**Collaborators**
Receive SensorEvent	SensorEvent
Lookup security zones based on sensorId	ConfigurationManager, SecurityZone
Check system mode	ModeManager
Check bypass rules	BypassManager
Trigger verification sequence	VerificationManager
Trigger escalation sequence	EscalationManager
Create and update IncidentRecord	IncidentRecord

Class: VerificationManager	
Handles the full alarm verification workflow for non-life-safety events. Sends verification notifications to homeowners, processes confirm/dismiss responses, and records verification actions.	
Responsibilities	**Collaborators**
Initiate user verification process	NotificationService
Handle user confirm action	LogManager, IncidentRecord
Handle user dismiss action	LogManager, IncidentRecord

Class: EscalationManager	
Executes full alarm escalation after confirmation or timeout. Controls the siren via LocalActuator, sends emergency notifications, triggers external dispatch requests, and updates the IncidentRecord.	
Responsibilities	**Collaborators**
Activate local sirens or alarms	LocalActuator
Notify homeowners via push/SMS	NotificationService
Dispatch emergency services(IVR/Police API)	DispatchService
Log escalation actions	LogManager
Update IncidentRecord status	IncidentRecord

Class: IncidentRecord	
Stores the lifecycle state of a single incident (start time, status, resolution). Updated by IncidentController and EscalationManager to maintain an audit trail.	
Responsibilities	**Collaborators**
Store incident metadata(ID, time, status)	
Update incident status (verified/escalated/resolved)	

Class: NotificationService	
Provides communication utilities for sending push notifications and SMS messages to homeowners during verification and escalation stages.	
Responsibilities	**Collaborators**
Send push notifications	
Send SMS messages	

Class: DispatchService	
Integrates with external emergency services. Sends dispatch requests (e.g., police/emergency API) and handles cancellation messages when an incident is resolved.	
Responsibilities	**Collaborators**
Send emergency dispatch request	
Cancel dispatch request	

Class: LocalActuator	
Controls local physical alarm devices such as sirens or lights. Used during escalation to activate or suppress audible alarms.	
Responsibilities	**Collaborators**
Active siren / local alarm	
Deactive siren	

3. Live Surveillance

Class: CameraController	
A controller class that manages all camera-related operations. It acts as the intermediary between the UI/System layer and the individual camera devices.	
Responsibilities	**Collaborators**
Request a live video stream from a specific camera.	SafeHomeCamera
Initiate and stop a two-way audio session	SafeHomeCamera
Set, remove, and verify a camera's privacy password.	SafeHomeCamera
Enable (activate) a camera's streaming functions.	SafeHomeCamera
Disable (deactivate) a camera for privacy.	SafeHomeCamera
Update a camera's recording settings.	SafeHomeCamera
Update a camera's notification policy.	SafeHomeCamera
Maintain a list of all managed cameras.	SafeHomeCamera

Class: SafeHomeCamera	
A logical representation (entity) of a physical camera. It manages the camera's state (e.g., status, password) and abstracts the low-level hardware interactions.	
Responsibilities	**Collaborators**
Provide the live video stream.	Camera
Play audio on the device's speaker.	Camera
Provide the audio stream from the device's microphone.	Camera
Store and verify its privacy password.	
Set its operational status to enabled or disabled.	Camera
Store its configuration (name, ID, recording settings).	

Class: Camera	
A low-level class (stereotype <<Device>>) representing the actual hardware device driver. It provides the raw, unprocessed streams and hardware control functions.	
Responsibilities	**Collaborators**
Begin transmitting the raw video feed from the hardware.	SafeHomeCamera
Stop transmitting the raw video feed.	SafeHomeCamera
Send raw audio data to the device's speaker.	SafeHomeCamera
Get the raw audio stream from the device's microphone.	SafeHomeCamera

Class: Recording	
A data-holding class (entity) that stores metadata about a single recorded video clip, such as its ID, timestamp, event type, and storage location.	
Responsibilities	**Collaborators**
Store metadata (clipId, cameraId, startTime, storagePath, etc.).	
Provide its metadata details to other services.	RecordingManager
Store whether it is password-protected.	

Class: RecordingManager	
A controller class that manages all operations related to finding, playing, and sharing recorded video clips.	
Responsibilities	**Collaborators**
Search for recordings based on filter criteria.	FileManager
Retrieve a specific recording for playback.	FileManager
Prepare a recording file for download.	FileManager, Recording
Generate a secure, time-limited link for sharing.	FileManager

4. System and User Management

Class: ConfigurationManager	
Central class for managing access to and updates of persistent system configuration data (settings, modes, security zones).	
Responsibilities	**Collaborators**
Manages access to configuration data	SystemSettings, SafeHomeMode, SecurityZone
Updates system settings	SystemSettings
Modifies security zone information	SecurityZone
Stores updated system configuration data	StorageManager

Class: SystemSettings	
Defines global operating configurations, such as system lock time and the panic phone number.	
Responsibilities	**Collaborators**
Manages global system settings information	

Class: SecurityZone	
Manages information for each defined security zone, including sensor lists and armed status.	
Responsibilities	**Collaborators**
Defines sensors list for a SecurityZone	
Identifies the armed status of SecurityZone	

Class: User	
Manages persistent profile data, assigned roles/permissions, and acts as the container for user credentials. (Primarily a User Management concern).	
Responsibilities	**Collaborators**
Reads/writes user profile information (e.g., name, phone number)	
Reads/writes assigned user role and permissions	
Provides access to user credentials	LogInInterface
Stores profile and role data	StorageManager

Class: LogInManager	
Manages the entire logging process, including authentication and password changes, using LogInInterface instances.	
Responsibilities	**Collaborators**
Manages log in	LogInInterface
Manages log out	LogInInterface
Manages changing passwords	LogInInterface

Class: LogInInterface	
Manages log in information (username, password, access level) for all users based on their log in interface.	
Responsibilities	**Collaborators**
Reads/writes username	
Reads/writes password	
Saves log in information	StorageManager

5. Indoor Monitoring & Device Control

Class: IndoorDeviceRequestManager	
Manages direct, manual user requests for device control and data reporting.	
Responsibilities	**Collaborators**
Authenticates user requests for manual control.	LogInManager
Change each device's option	SmartDevice
Apply user requested mode on indoor devices	SmartDevice

Class: AutomationRequestManager	
Manages the lifecycle of persistent, automated rules.	
Responsibilities	**Collaborators**
Authenticates user requests for manual control.	LogInManager
Creates a new automation rule	Automation
Stops and removes a rule.	Automation
Modifies an existing rule.	Automation

Class: SmartDevice	
A Hardware Abstraction Layer (HAL) that abstracts and isolates physical device communication.	
Responsibilities	**Collaborators**
Relays commands to a physical device.	SmartLight, VentilationSystem
Polls a physical device for its state.	AirQualitySensor, SmartMeter…
Discover and register new device	

Class: Automation	
Encapsulates the logic and state for a single running automation rule (e.g., IAQ management).	
Responsibilities	**Collaborators**
Periodically evaluates sensor data.	AirQualitySensor
Commands devices when a trigger is met.	SmartDevice, VentilationSystem

Class: SmartMeter	
An entity representing the central, main power meter for the entire home	
Responsibilities	**Collaborators**
Reports total home power usage for a certain period.	

Class: SmartLight	
An entity representing a single smart light bulb.	
Responsibilities	**Collaborators**
Sync/Changes the on/off state.	
Sync/Changes the brightness level.	
Sync/Changes the color.	

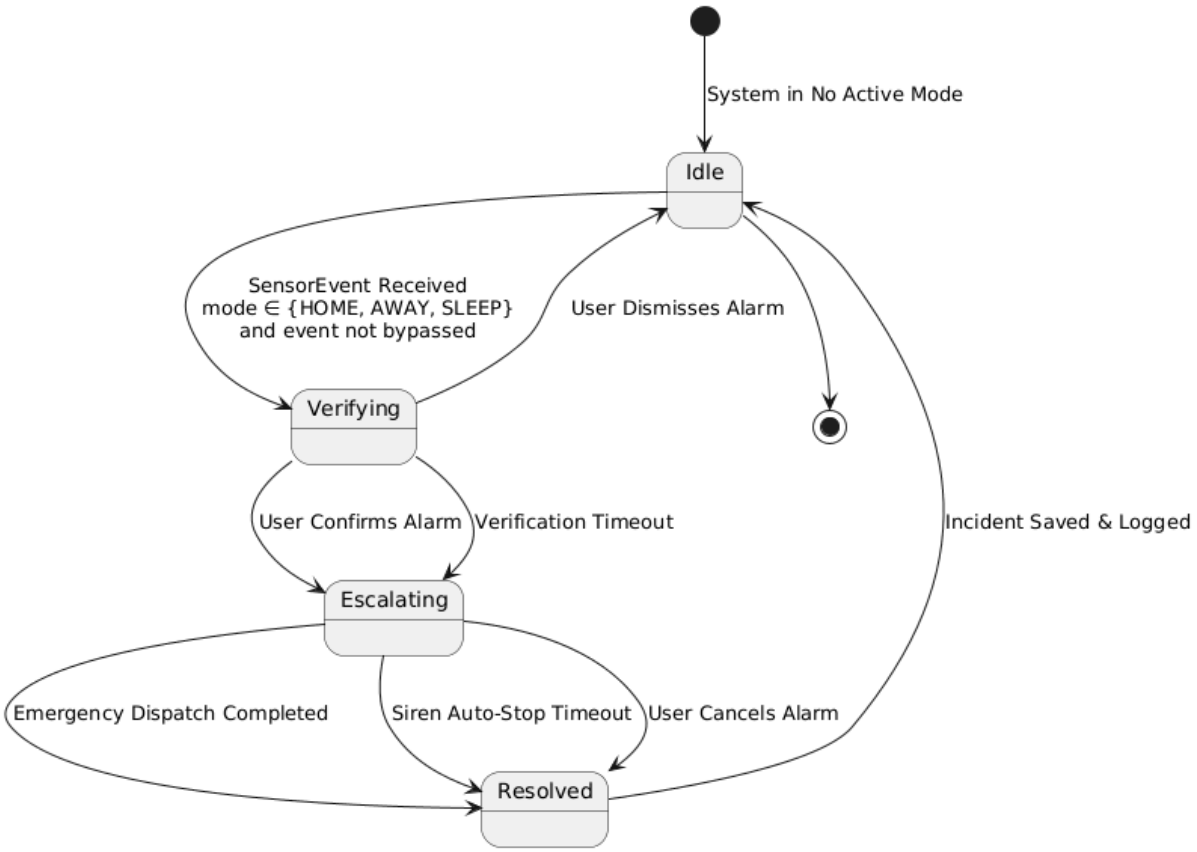
Class: AirQualitySensor	
An entity that monitors indoor air quality (e.g., CO_2)	
Responsibilities	**Collaborators**
Provides the current pollutant level.	Automation
Checks whether IAQ exceeded threshold	Automation

Class: VentilationSystem	
An entity representing a smart fan or air purifier.	
Responsibilities	**Collaborators**
Turns the system on/off.	Automation

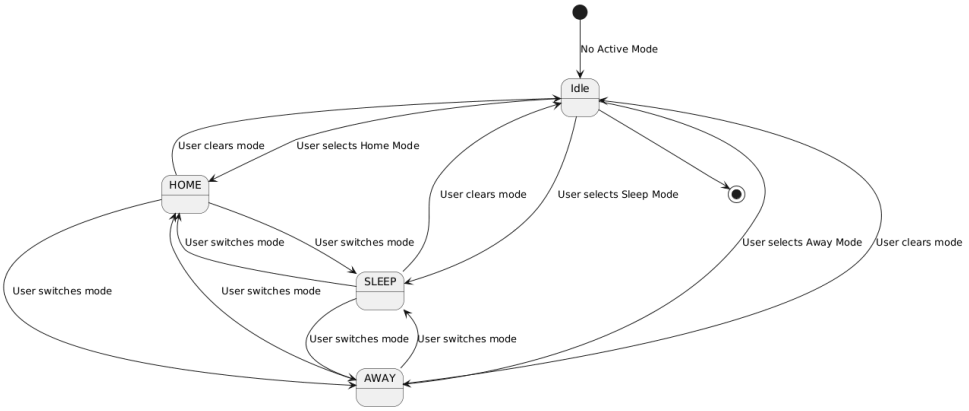
V. STATE DIAGRAM

1. Intelligent Security

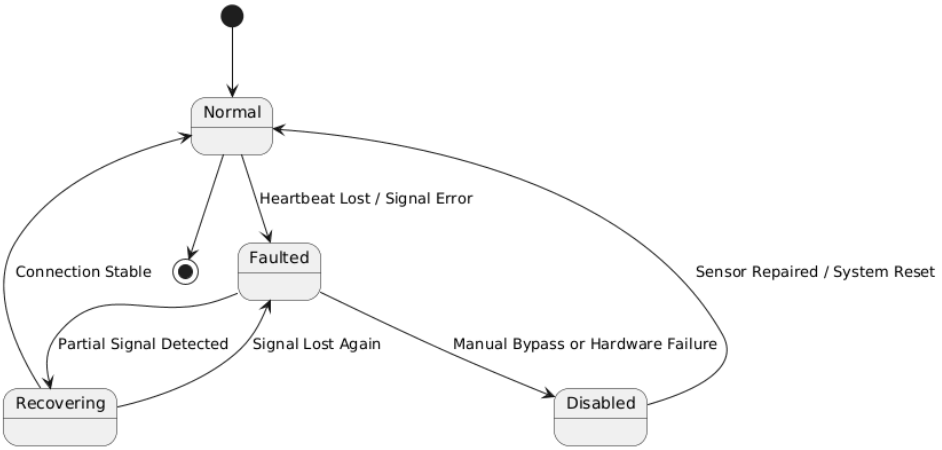
1.1. Incident Lifecycle



1.2. Security Mode

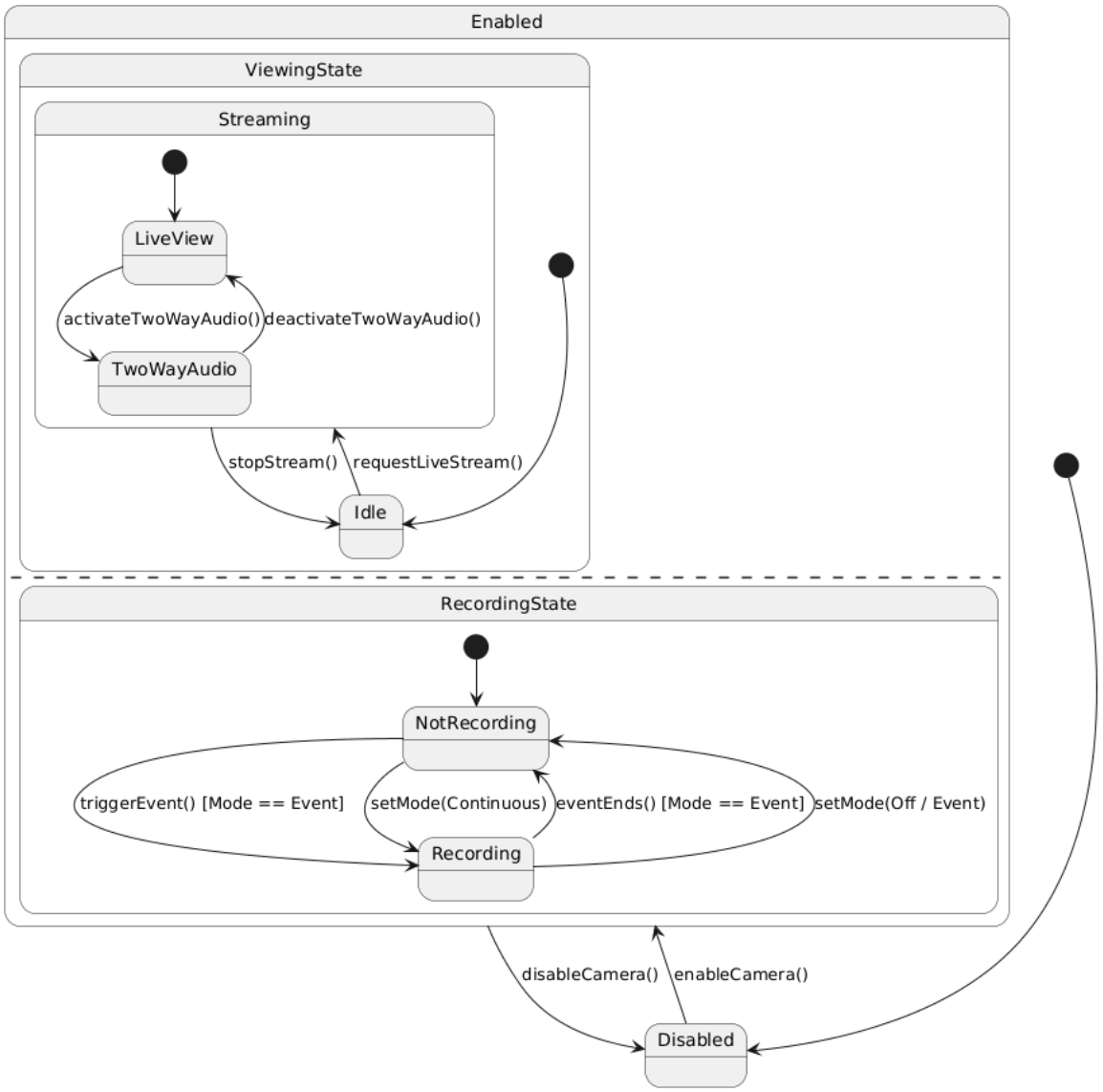


1.3. Sensor Connectivity & Fault



2. Live Surveillance

2.1. Camera status



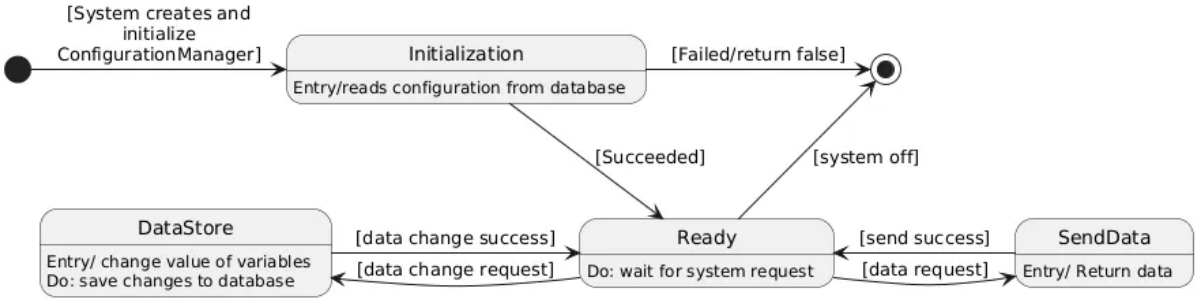
2.2. State Diagrams for the remaining class

Following classes in the Live Surveillance diagram, unlike SafeHomeCamera, do not have distinct state diagrams since:

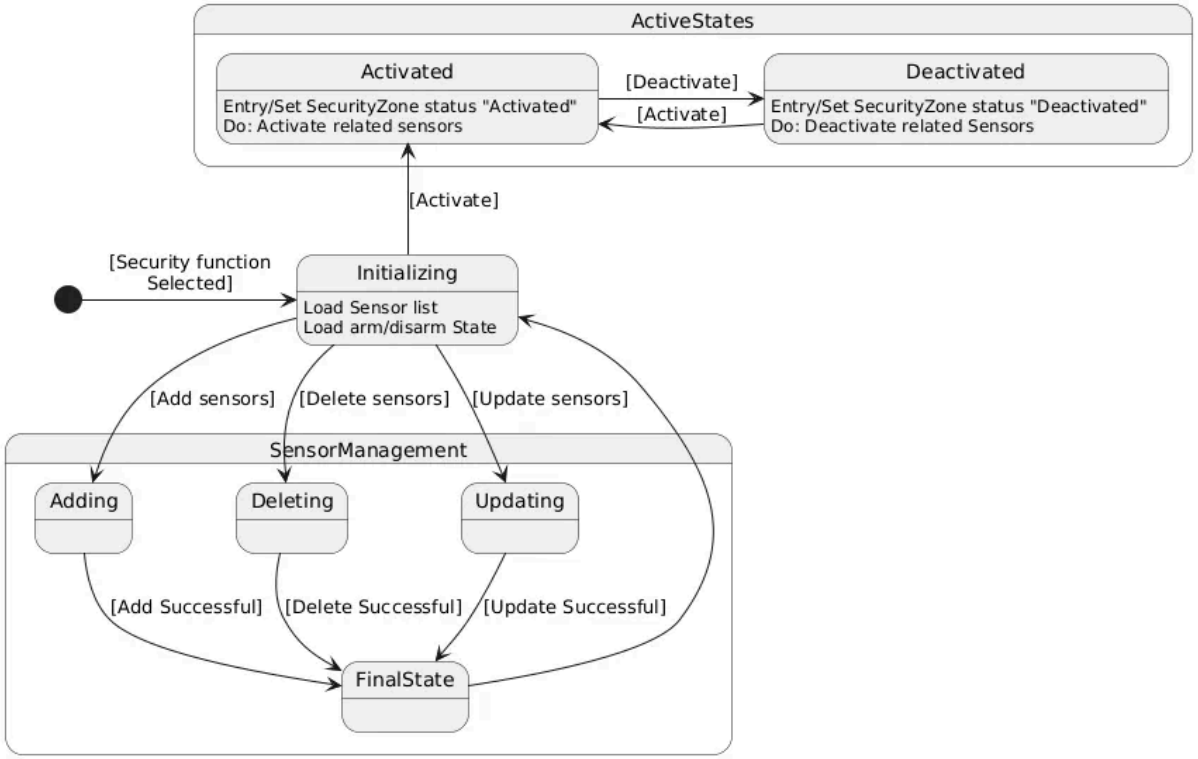
1. CameraController, RecordingManager: These classes function as 'managers'. They manage and coordinate the states of other objects, such as SafeHomeCamera and FileManager, rather than possessing complex internal states of their own, like "Recording" or "Streaming".
2. Recording: This class is a pure data container. It only holds data such as clipId, startTime, and storagePath, and does not have its own internal states like "Playing" or "Paused".
3. FileManager: This class is a service that abstracts access to the database or file system. While it might internally have states such as "Connecting" or "Connected," these are encapsulated and are not relevant to the RecordingManager. From the RecordingManager's perspective, the FileManager is an entity that "always" provides functions like query() or retrieveFile().
4. Camera (Device): This class represents the actual physical camera hardware. This object naturally possesses complex states (e.g., "Online", "Offline", "Streaming"); however, we have already drawn a State Diagram for the SafeHomeCamera class, which logically represents this physical state. The CameraController does not control the Camera (Device) directly; instead, it manages the hardware's state through the logical SafeHomeCamera object. Therefore, the SafeHomeCamera state diagram represents all necessary camera states within this system.

3. System & User Management

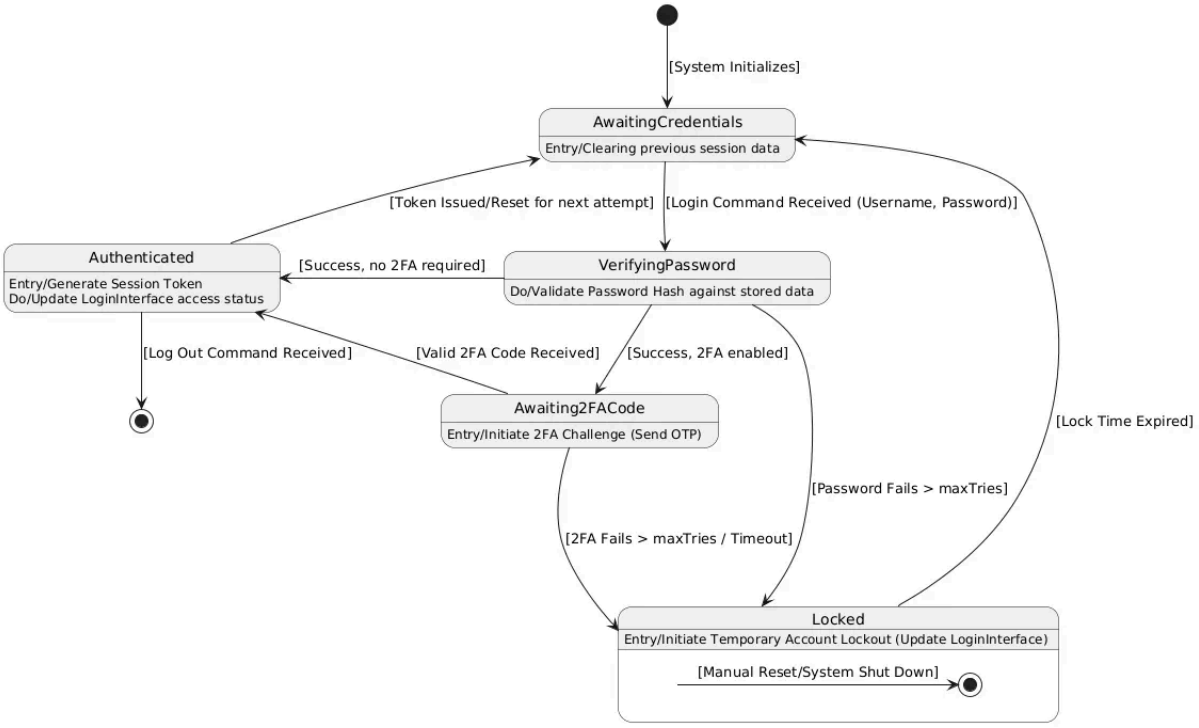
3.1. Configuration Management



3.2. Security Zone

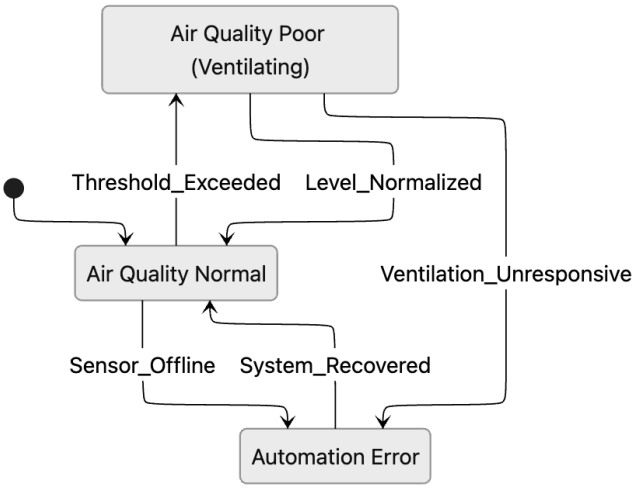


3.3. Log in Management

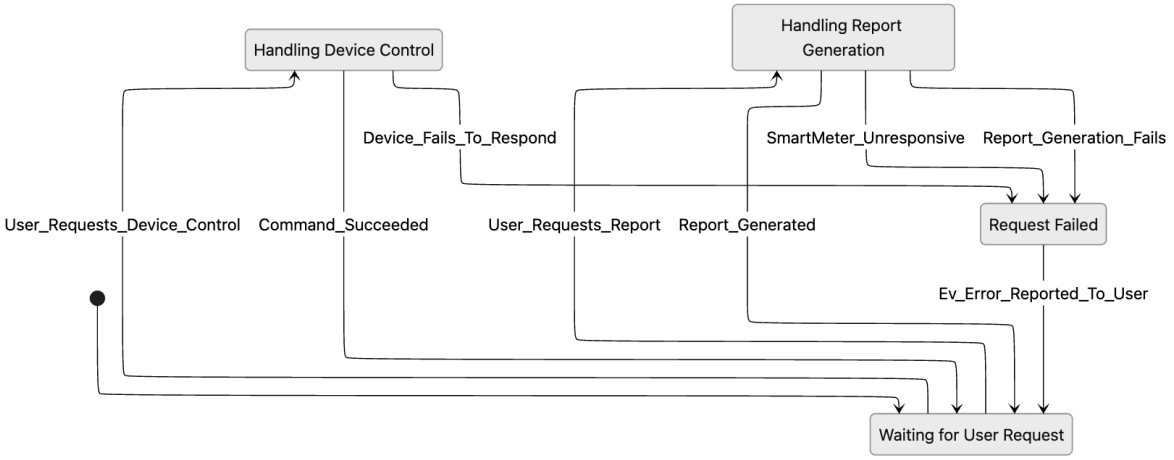


4. Indoor Monitoring & Device Control (Device State Management)

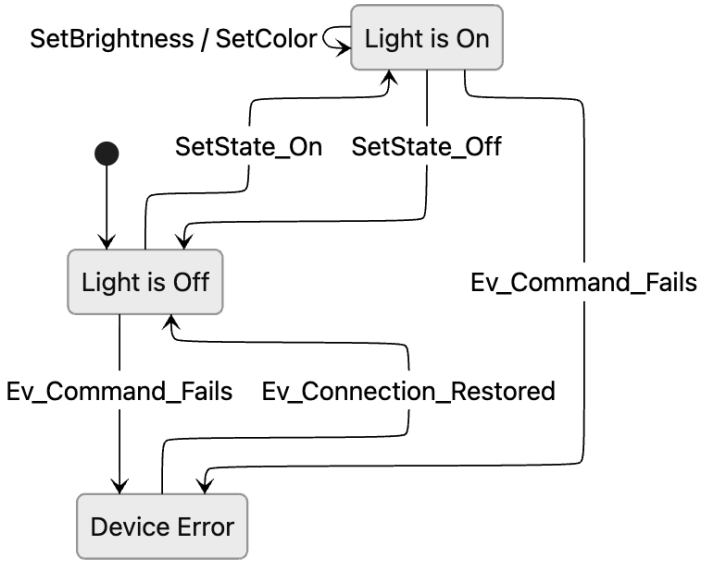
4.1. Automation



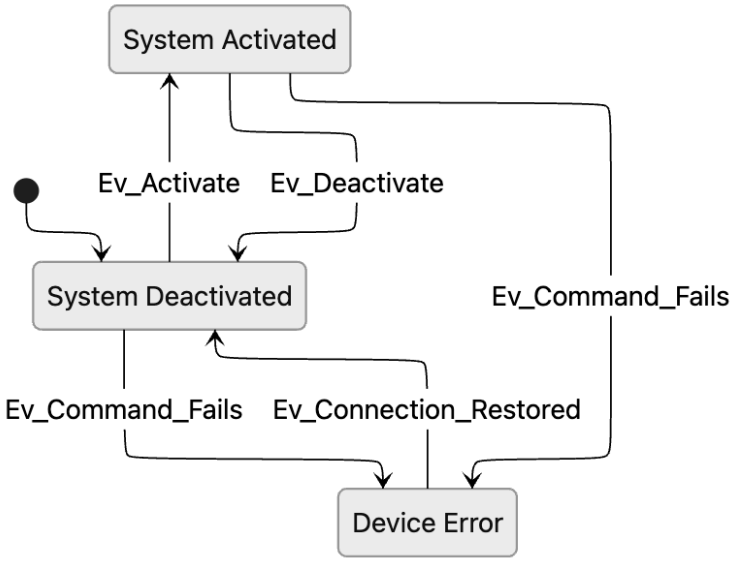
4.2. IndoorDeviceRequestManager



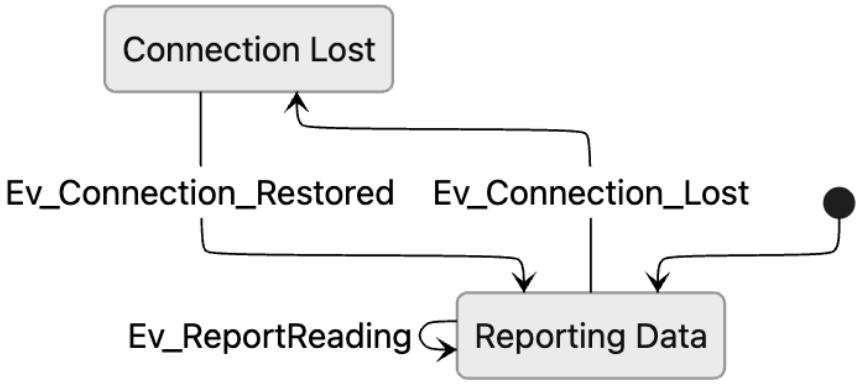
4.3. SmartLight



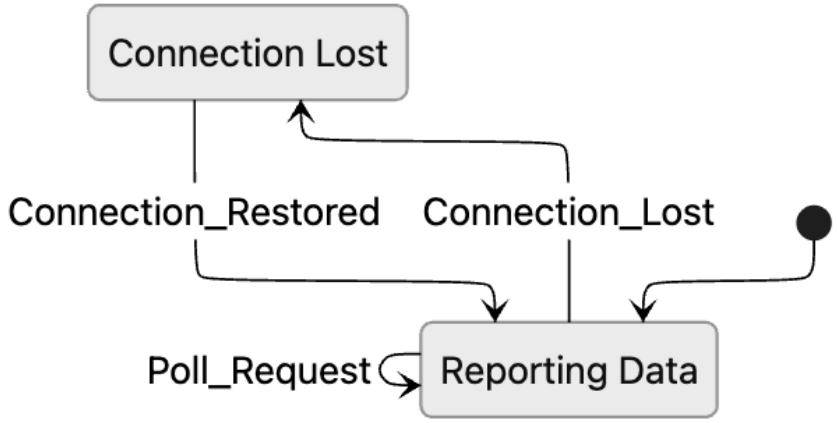
4.4. VentilationSystem



4.5. AirQualitySensor



4.6. SmartMeter



VI. DESIGN EVALUATION

1. Architectural Design Metrics - Fenton's simple morphology metrics

# of nodes (N)	= 43	Control Panel, Web Interface is excluded here. But System has been counted as root node
# of arcs (E)	= 64	
size = N + E	= 43 + 64 = 106	
depth (longest path)	= 3	System ~IndoorDeviceRequestManager ~ SmartDevice ~ Sensor
width	= 12	
arc-to-node ratio	≈ 1.488	

2. CK Metrics

Depth of the inheritance tree (DIT)	0 ~ 1, avg 0.23
Number of Children (NoC)	0 ~ 4, avg 0.33
Coupling Between Object Classes (CBO)	0 ~ 11, avg 2.4
Lack of Cohesion in Methods (LCOM)	0 ~ 2, avg 0.087

3. MOOD Metrics

3.1. MIF (Method Inheritance Factor)

MIF ≈ 0.08

- The ‘System’ node is excluded from calculating the MIF index.
- Reason for the low value:
  Although inheritance was applied to classes expected to be similar, only a few shared attributes or methods were defined, resulting in such a low score.
  This should be improved by increasing the number of classes that implement functionality through inheritance.

	Class	Md	Mi	Ma(Md + Mi)
1	ConfigurationManager	5	0	5
2	LogManager	2	0	2
3	StorageManager	1	0	1
4	SystemSettings	1	0	1
5	SafeHomeMode	0	0	0
6	SecurityZone	0	0	0
7	Log	0	0	0
8	LogInManager	3	0	3
9	User	0	0	0
10	Homeowner	2	0	2
11	Guest	1	0	1
12	LogInInterface	1	0	1
13	SmartDevice	2	0	2
14	AutomationRequestManager	4	0	4
15	IndoorDeviceRequestManager	4	0	4
16	SmartLight	3	2	5
17	AirQualitySensor	1	2	3
18	SmartMeter	1	2	3
19	VentilationSystem	2	2	4
20	Automation	1	0	1
21	BaseSensorAdapter	3	0	3
22	SensorEvent	0	0	0
23	ModeManager	2	0	2
24	SecurityMode	0	0	0
25	BypassRule	0	0	0
26	BypassManager	2	0	2
27	IncidentController	4	0	4

28	VerificationManager	3	0	3
29	EscalationManager	3	0	3
30	IncidentRecord	1	0	1
31	IncidentLogger	1	0	1
32	LocalActuator	2	0	2
33	NotificationService	2	0	2
34	DispatchService	2	0	2
35	CameraController	9	0	9
36	RecordingManager	4	0	4
37	SafeHomeCamera	8	0	8
38	Camera	4	0	4
39	Recording	1	0	1
40	FileManager	5	0	5
41	Actuator	0	0	0
42	Sensor	0	0	0

3.2. CF (Coupling Factor)

CF = 26 / (42 * 42 - 42) ≈ 0.015

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
18	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

VII. WHO DID WHAT

Gyeongyeon Kim (20180048)
1. Complete of Architecture structure for Indoor Monitoring and Device Control 2. Drawing Class Diagram for Indoor Monitoring and Device Control 3. Writing CRC Cards for Indoor Monitoring and Device Control 4. Drawing State Diagram for Indoor Monitoring and Device Control 1. Complete of Architecture structure for Infrastructure 2. Drawing Class Diagram for Infrastructure 3. Writing CRC Cards for Infrastructure 4. Drawing State Diagram for Infrastructure 1. Design detailed DB table columns based on HW2; not included in the report since columns may change significantly with GUI and sensor interface specifications. 2. Perform metric calculations.
Wonjoon Lee (20190480)
1. Complete of Architecture structure for System and User Management 2. Drawing Class Diagram for System and User Management 3. Writing CRC Cards for System and User Management 4. Drawing State Diagram for System and User Management 1. Complete of Architecture structure for Remote Access and Account 2. Drawing Class Diagram for Remote Access and Account 3. Writing CRC Cards for Remote Access and Account 4. Drawing State Diagram for Remote Access and Account 1. Refine document readability
Sumin Lee (20200468)
1. Complete of Architecture structure for Intelligent Security 2. Drawing Class Diagram for Intelligent Security 3. Writing CRC Cards for Intelligent Security 4. Drawing State Diagram for Intelligent Security 5. Recorded the team meeting session and wrote the summarized meeting notes for documentation and review. 6. Compiled and documented all SDS-specific terminology, identifying terms not present in the original SRS and generating the corresponding entries for the Glossary section.
Yeongjun Joo (20210623)
1. Complete of Architecture structure for Live Surveillance 2. Drawing Class Diagram for Live Surveillance 3. Writing CRC Cards for Live Surveillance 4. Drawing State Diagram for Live Surveillance

VIII. MEETING LOG

Date	2025/11/06 19:00 - 20:00
Location	E9 StudyRoom 2B and Zoom
Attendees	Gyeongyeon Kim, Wonjoon Lee, Sumin Lee, Yeongjun Joo
Summary	**Meeting Objective** To assign SDS module responsibilities, decide on documentation and diagram tools, and set internal deadlines for the SDS submission. **Key Discussion Points** **1. Task Division and Module Assignment** ○ The team confirmed that the SDS will cover **five main subsystems**: Intelligent Security, Live Surveillance, System & User Management, Remote Access & Account, and Indoor Monitoring & Device Control. ○ Each member will take charge of one or more modules based on their previous SRS experience: ■ Sumin Lee – Security modules ■ Yeongjun Joo – Live Surveillance ■ Wonjoon Lee – System & User Management, Remote Access & Account ■ Gyeongyeon Kim – Indoor Monitoring & Device Control ○ Team members agreed to reuse their previous parts to ensure consistency. **2. Tools and Workflow** ○ The team decided to continue using **PlantUML** for all diagrams (class, state, CRC, architecture). ○ All diagrams and code will be uploaded to **Notion**, while text sections will be compiled in **Google Docs** for the final submission. ○ For collaborative editing, each member will create and share their **own Notion page link** to prevent version conflicts. **3. Schedule and Deadlines** ○ Official deadline: **November 14** (submission). ○ Internal deadlines: **November 10** – First draft of all assigned modules (including diagrams) **November 12, 10 AM** – Second review meeting for integration **November 13, 7 PM** – Final revision meeting The team emphasized the need to finish the first version by Monday to avoid last-minute rushing. **4. Document Standards** ○ **CRC Card Format:** Follow example template exactly. ○ **Architecture Overview:** One member will draft the overall system diagram showing how all five subsystems interact via hub, cloud, and devices. ○ The “Data Interface Flow Specification” section will be omitted for now as it was judged unnecessary at this stage. ○ The team will replicate Team 1’s Glossary format and add terms encountered while writing each module. **5. Work Process and Quality Focus** ○ Members agreed that each should **individually create diagrams first**

	to deepen understanding before merging them. ○ Emphasis was placed on achieving **consistency across subsystem diagrams** (naming, style, and attribute structure). **Decisions Made** 1. Module responsibilities finalized (four members across five subsystems). 2. PlantUML selected for all diagrams; Notion and Google Docs as documentation tools. 3. Internal deadlines set (Nov 10: first draft, Nov 12: review, Nov 13: final meeting). 4. Team 1's Glossary templates will be followed. 5. Example's CRC templates will be followed. 6. The "Data Interface Flow Specification" section will be omitted unless later required. **Action Items** 1. **All members:** Complete diagrams and descriptions for their assigned modules by Monday. 2. **Next meeting:** Wednesday, November 12, 9:00–11:00 AM – First integration review meeting.

Date	2025/11/10 12:00 - 12:50
Location	E3-2
Attendees	Gyeongyeon Kim, Wonjoon Lee, Sumin Lee, Yeongjun Joo
Summary	**Meeting Objective** To review each member's SDS progress, ensure consistency in diagram formats, and discuss integration of subsystem outputs before the final compilation. **Key Discussion Points** **1. SDS Progress Review (Subsystem Updates)** **a. Security Module (Sumin Lee):** i. Organized five main classes: Sensor Monitoring, Mode Configuration, Incident Handling, Alarm Transmission, and Logging. ii. Generated diagrams using GPT based on SRS content; English translation pending. iii. State diagram currently in draft form, requires validation. **b. Live Surveillance (Yeongjun Joo):** i. Covered camera and audio streaming, recording, and storage functions. ii. Potential overlap with Security logging function identified; coordination planned. iii. Architecture divided into UI, SafeHome Cloud, and Home Network layers for clarity. **c. System & User Management / Remote Access & Account (Wonjoon Lee):**

	i. Developed log in and session management flow; discussed unified logging system vs. separate logs. ii. Clarified flow for two-factor authentication and lockout conditions. iii. CRC Cards and diagrams uploaded to Google Docs for integration. **d. Indoor Monitoring & Device Control (Gyeongyeon Kim):** i. Expanded module scope to include air-quality sensors (CO_2/VOC), device mode changes, and energy report generation. ii. Adjusted state diagrams to reflect device activation/deactivation and periodic monitoring (e.g., 5-minute cycles). **2. Tool and Format Alignment** a. Both **Mermaid** and **PlantUML** allowed for diagram creation. b. **CRC Cards:** Must include color-coded sections and explicit Responsibility–Collaborator linkage. c. **Final Submission:** Google Docs master file → converted to Word (.docx) for formal submission. d. Internal note: Maintain consistent structure across diagrams and CRC cards. **3. Workflow and Integration Plan** Workflow: Use SRS as input → generate subsystem architecture → class diagram → CRC cards. a. File management: Code and images stored locally; shared via Google Drive. b. Each member to align CRC templates, fill responsibilities, and add collaboration links before integration. c. Google Docs serves as the live draft; formatting and internal links will be finalized in Word export. **4. Quality and Documentation Standards** a. Members agreed to refine diagrams generated by GPT, ensuring accuracy and logical consistency. b. CRC template uniformity emphasized for visual coherence in the final report. c. Plan to cross-check shared classes (e.g., logging, authentication) for naming consistency. **Decisions Made** ● Each subsystem's first draft (class/state/CRC diagrams) completed or near completion. ● Agreed on hybrid diagram tool use (Mermaid + PlantUML). ● Unified CRC format and English translation to be done after integration. ● Overlapping functionalities (logging, authentication) to be resolved collaboratively. **Action Items** ● **All members:** Finalize CRC cards and diagram consistency by Wednesday. ● **Sumin Lee:** Integrate subsystem diagrams into overall architecture. ● **Wonjoon Lee:** Review overlapping logging components with Yeongjun Joo. ● **Gyeongyeon Kim:** Add Homeowner and Control Panel interactions to Indoor module.

	● **Next meeting:** November 12 (Wed), 9:00–10:00 AM – First integration review meeting.

Date	2025/11/13 16:00 - 18:00
Location	E9 room 3C, Library
Attendees	Gyeongyeon Kim, Wonjoon Lee, Sumin Lee, Yeongjun Joo
Summary	**Meeting Objective** To review each member's SDS progress, and discuss overall design. **Key Discussion Points** 1. Diagram format a. Since Notion provides real-time Mermaid editing, many diagrams were created using Mermaid. b. Prefer expressing diagrams in PlantUML c. Use other tools when class diagrams become complex and require a cleaner layout. 2. Interface simplification a. In the class diagrams for WebInterface and Control Panel, user calls are the main interaction, and since detailed interfaces are not defined, attributes and methods were omitted. 3. Definition of the logging system (Activity Log / Audit Log). a. The Activity Log records all system events chronologically (sensor activations, user actions, device state changes), while the Audit Log captures detailed administrative and security-sensitive operations (who, when, from which device) for verification, incident analysis, and compliance. 4. Organizing the system architecture document structure. a. Reordered subsystem sections from 3.1 Overview to Intelligent Security, Live Surveillance, User & Configuration, Device & Automation, and Infrastructure for clarity and consistency. 5. Discussion on the composition of the Infrastructure diagram. a. Decided to place StorageManager and LogManager in Infrastructure but also include LogManager in related subsystem diagrams for clarity and easier reference. 6. Definition of StorageManager / FileManager roles. a. StorageManager handles DB connections, and FileManager handles creating, storing, and exporting recordings. b. Separation was needed because the old StorageRepository had a different purpose. 7. Discussion regarding SecurityZone and ModeManager. a. Discussion on whether to merge SafeHomeMode and SecurityMode. b. Decided to standardize on SecurityMode because SafeHomeMode does not appear in the SRS. 8. Discussion on where to place ModeManager. a. Consider whether it belongs in Intelligent Security or Infrastructure. b. Decided to keep it in its current location since it is not a common module. 9. Standardization of relationship notation in class diagrams.

	a. Since the overall diagram adopts the 1-to-X relationship notation, detailed diagrams must also follow the same convention. b. Ensuring consistency and accuracy in relationship notation across the document. 10. Coordination of schedule and task assignments. **Decisions Made** ● Draw diagram using PlantUML ● Define Activity Log and Audit Log separately according to security and compliance needs. ● Rearrange subsystem documentation sections from 3.1 to 3.6. ● Include LogManager not only in Infrastructure but also in related subsystems. ● Separate StorageManager (DB) and FileManager (recording storage) as distinct classes. ● Remove SafeHomeMode and unify with SecurityMode. ● Do not support user-defined custom modes since it is an open issue in the SRS. ● Keep ModeManager in its current location (not moved to Infrastructure). ● Apply the 1-to-X relationship notation to all detailed diagrams. **Action Items** ● **All members:** final diagram review, refine the report and compute metrics. ● **Sumin Lee:** add 1-to-X relationships to the detailed class diagrams. ● **Gyeongyeon Kim:** add 1-to-X relationships to the detailed class diagrams. ● **Next meeting:** November 14 (Fri), 5:00–9:00 PM – Final refinement meeting.

Date	2025/11/13 19:00 - 22:00
Location	E9 room 4D, Library
Attendees	Gyeongyeon Kim, Wonjoon Lee, Sumin Lee, Yeongjun Joo
Summary	**Meeting Objective** To improve the ambiguous parts in the overall structure **Key Discussion Points** **1. Architecture Structure and Template Issues** a. Simplicity of Sub-architecture: A decision was made to omit detailed functions from the sub-architecture diagrams to maintain simplicity and clarity. b. Ambiguity in Architectural Diagram: The team is unclear on what the architectural structure diagram should depict. Specifically, it is unknown what exists under "External Interface" (besides sensors), which is positioned alongside "Sensor Management" and "Response Management" in the example tree structure.

c. Need for Template Modification: The provided DDS example is recognized as unsuitable for our project and must be entirely modified. The team is considering changing the architecture name in the diagram (e.g., from "Overall" to "Intelligent Security") and shifting from the template's vertical tree structure to a horizontal flow.

2. Component Placement and Dependencies

a. Location of Log Management: Log Management interferes with all components. Its final placement must be decided after the class diagram is reviewed.
b. Role of the Control Panel: The control panel is an interface that queries information from the system *upstream* of the security module. It was clarified that the security module is *not* dependent on the control panel.
c. Relationship between "Response" and "Incident":
    i. An overlap between "Incident Management" and "Response" was identified.
    ii. If "Incident Management" is definitively positioned as a higher-level module, its functions must be divided and moved down into sub-modules.
    iii. Logging is required for individual sensor activations (e.g., a door opening), not just for "Responses." This implies that logging functions are needed outside of just the "Response" module.

3. Function Scope and Completeness

a. Complexity of Security Functions: While Intelligent Security is complex, this is not necessarily due to the complexity of its sub-functions. The team agreed that having only three main sub-components is acceptable.
b. Omission of "Indoor Monitoring and Device Control": This function was missing from the original plan. It was agreed that this is essential for both the mobile and web applications and must be added to the design.

4. Specific Module Review

a. Intelligent Security (mistakenly 'Intelligent Bond'): The connection flow appears correct: proceeding first to "Intelligent Security," then to "Device Manager," which in turn connects back to the "System." This part is considered updated.

Decisions Made

1. Simplify Architecture Diagrams: Sub-architecture diagrams will be kept simple, focusing on top-level functions to avoid unnecessary complexity.
2. Revise Template: The DDS example template will be discarded. A new, horizontal architectural structure diagram will be designed to align with our project's SRS.
3. Clarify Dependencies: The Control Panel is defined as an actor/interface that *uses* the security module, not as a dependency *of* the module.
4. Redefine Module Hierarchy: "Incident Management" is confirmed as the high-level module. "Response" functions will be organized as sub-modules within it.
5. Expand Functional Scope: The "Indoor Monitoring and Device Control" function is now officially included in the project scope and must be added to the architecture.

	Action Items 1. Re-evaluate Log Management: After the class diagrams are finalized, the team will revisit the placement of Log Management to find a solution that minimizes interference with other components. 2. Redesign Architecture Diagram: Create the new architectural diagram reflecting the decisions made (e.g., horizontal flow, inclusion of all modules). 3. Integrate "Indoor Monitoring": Update all relevant design documents, including the new architecture diagram, to reflect the addition of the "Indoor Monitoring and Device Control" component. 4. Refine "Incident Management": Reorganize the sub-functions under "Incident Management" and clearly define their relationship with the 'Response' module.

APPENDIX A. GLOSSARY

System

The SafeHome system is designed as an integrated home automation and security platform that unifies five major functional domains: Intelligent Security, Live Surveillance, System & User Management, Remote Access & Account, and Indoor Monitoring & Device Control.

The HW#2_SDS_draft defines the architectural structure of this system, including how the subsystems governs global system states and how the ConfigurationManager maintains persistent configuration data.

These functionalities are based directly on the use cases defined in Team 1's SRS and form the foundation for the SDS design.

- Related Use Cases: UC 1.x, UC 2.x, UC 3.x, UC 4.x, UC 5.x
- Reference: I. OVERVIEW, II. ARCHITECTURAL STRUCTURE, III. CLASS DIAGRAMS, V. STATE DIAGRAMS

SafeHome Executive

The term *SafeHome Executive* refers to the architectural role fulfilled by the System within the overall design.

In Chapter II, the SDS describes each module based on its operational responsibility (typically using a *-management* naming convention).

In contrast, Chapter III lists the concrete classes that implement these responsibilities (typically expressed using *-manager*), thereby clarifying the relationship between conceptual components and actual class definitions.

- Reference: II. ARCHITECTURAL STRUCTURE

Web Interface

The Web Interface is a browser-based client that provides features equivalent to the mobile application, primarily intended for configuration tasks and administrative controls from desktop environments.

In the SDS, the Web Interface is defined as a class; however, detailed GUI components were intentionally omitted because the interface design is outside the scope of this document.

During class definition, the team clarified that the Web Interface should *not* be interpreted as maintaining a strict 1:1 connection with the system, and that server–client relationships must be explicitly modeled.

- Related Use Cases: All user-facing UCs, including UC 1.2.1 (Configure Conditions by Mode), UC 1.3.1 (One-Touch Modes), UC 4.1.2 (Log In)
- Reference: UC 4.1.2, III. CLASS DIAGRAMS, Meeting Log 11/13

Control Panel

The Control Panel is a physical touchscreen device installed inside the home that offers direct access to essential system functions, even without an internet connection.

It supports arming/disarming the system, accessing camera feeds, and managing core local settings.

Like the Web Interface, only the class itself was defined in the SDS; specific GUI details were excluded because they fall outside the scope of this specification.

To prevent confusion, the SDS clarifies that the Control Panel does not imply a 1:1 server connection model. Additionally, in cases of conflicting commands, Control Panel inputs take priority over remote/mobile inputs.

- Related Use Cases: UC 1.3.1 (One-Touch Modes), UC 4.1.2 (Log In), UC 1.3.2 (Sensor Bypass)
- Reference: Fig. 1 (p.6), I. OVERVIEW (Assumption), V. STATE DIAGRAMS

BaseSensorAdapter

An abstract interface responsible for converting raw physical sensor signals into standardized SensorEvent objects.

This class abstracts hardware variability and ensures that all downstream components receive events in a unified format.

- Reference: Intelligent Security – Sensor Layer
- SRS Status: Not defined; derived from UC 1.1.x (Sensor Monitoring)

IncidentController

The primary coordinator of Intelligent Security workflows.

It receives normalized sensor events, checks the active security mode, evaluates bypass rules, triggers the verification process, and escalates incidents when required.

It also creates and updates IncidentRecord objects.

- Reference: Intelligent Security Core Flow
- SRS Status: Not explicitly named; derived from UC 1.2.x (Incident Management)

VerificationManager

A subsystem component that implements the Alarm Verification Step workflow.

It sends verification prompts to users, handles confirmation/dismissal inputs, and logs all verification-related actions.

- Reference: UC 1.2.2 (Alarm Verification Step)
- SRS Status: Not named in SRS; logic implied in the workflow

EscalationManager

Handles the full escalation procedure after verification or timeout.

Controls sirens via local actuators, sends alerts, and triggers emergency dispatch requests.

- Reference: UC 1.2.3 (Emergency Service Integration)
- SRS Status: Not named in SRS; functionality derived from escalation steps

BypassManager

Manages sensor bypass states.

Based on active security mode and bypass rules, it determines whether a sensor event should be ignored.

- Reference: UC 1.3.2 (Sensor Bypass)
- SRS Status: Not present; introduced in SDS to model bypass logic explicitly

IncidentRecord

A dedicated data entity that tracks the lifecycle of a single incident, from creation through verification, escalation, and resolution.

Used for audit, evidence tracking, and system history management.

- Reference: Incident Logging & Review
- SRS Status: Not listed in SRS; added in SDS for persistent incident tracking

NotificationService
An internal communication service responsible for sending push notifications and SMS alerts to users.
The SRS describes notification behavior, but does not define a dedicated service component, so this class was introduced to encapsulate notification delivery logic.
- Reference: UC 1.2.2, UC 2.3.2
- SRS Status: Not explicitly named; SDS-specific service extraction

DispatchService
A service interface for interacting with external emergency responders (e.g., police or IVR systems).
Provides request and cancellation mechanisms for alarms requiring dispatch.
- Reference: UC 1.2.3 (Auto Call / Emergency Integration)
- SRS Status: Not defined; SDS introduces concrete service class

LocalActuator
Controls local hardware devices such as sirens or indoor alarms that must respond instantly during escalation events.
- Reference: UC 1.2.x escalation behavior
- SRS Status: Not explicitly defined; SDS formalizes this hardware abstraction

CameraController
Handles low-level camera operations including live streaming, camera activation/deactivation, and privacy protection functions.
The design is based on SRS camera-related use cases, though the SRS does not specify a controller-level class.
- Reference: UC 2.1.x (Camera Viewing & Control)
- SRS Status: Functional description exists, but class name is SDS-specific

RecordingManager
Manages recording storage, playback, export, and evidence sharing.
Implements the workflow for locating recorded clips, preparing them for download, and generating secure shareable links.
- Reference: UC 2.2.1, UC 2.2.2 (Search / Playback / Evidence Export)
- SRS Status: Not defined by name; SDS adds explicit manager

IndoorDeviceRequestManager
Processes user commands for indoor devices such as lights, smart meters, and air sensors.
Validates permissions, writes logs, and sends commands to the SmartDevice.
- Reference: UC 5.1.1 (Indoor Device Control)
- SRS Status: Not named; introduced to support modular command handling

AutomationRequestManager
Executes ventilation automation routines based on IAQ sensor data and system rules.

Coordinates with SmartDevice and maintains automation states.

- Reference: UC 5.2.1 (IAQ Monitoring & Automation)
- SRS Status: Not present; SDS explicitly defines automation flow

Automation

Represents a rule-driven automation scenario, such as air-quality-based ventilation. Encapsulates conditions and actions associated with automated flows.

- Reference: UC 5.2.x
- SRS Status: SDS-specific representation of automation rules

SystemSettings

An internal configuration entity that stores persistent system-wide settings such as mode configuration, thresholds, and connectivity parameters.

- Reference: UC 1.2.1 (Configure Alarm Conditions), UC 3.2.1 (System Dashboard)
- SRS Status: Not defined as a class; SDS introduces dedicated data model

Indoor Air Quality (IAQ)

A measure of air cleanliness based on pollutant levels (CO_2, VOCs, particulates). SafeHome can monitor IAQ and automatically activate ventilation when thresholds are exceeded.

- Reference: SRS Glossary: Indoor Air Quality (IAQ)