CS350 Safehome Project

Software Design Specification (SDS)

Project Team #8

Members:

20256450	Jamal Alibalayev
20256436	Yonas Alexander Grossard Amin
20256444	Alan Pak To Cheung
20256439	Jongyoon Baek

Korea Advanced Institute of Science and Technology

Table of Contents

I. Overview **5**
1. Introduction 5
2. Goal 5
3. How the design work proceeded 5
4. Assumptions 6

II. Architectural structure **8**
0. Overall Architecture 8
1. Intelligent Security 9
1.1 Overview of Intelligent Security 9
1.2 Sensor Monitoring 9
1.3 Incident Management 10
1.4 Security Mode & Device Control 11
2. Live Surveillance 12
3. System and User Management 13
4. Remote Access and Account 14
5. Indoor Monitoring and Sound Control 15

III. Class Diagram **16**
1. Intelligent Security 16
1.1 Security Domain Model (Core Entities) 16
1.2 Sensor Monitoring & Event Processing 17
1.2.a Sensor Monitoring (Actor/Manager/Repos/Gateway) 17
1.2.b Sensors, Camera, and Event Data (UC 1.1) 18
1.2.c Actuators · Services · Repositories (UC 1.1) 19
1.3 Incident Management & Response 20
1.4 Security Policy & Control 21
2. Live Surveillance 22
3. System and User Management 22
4. Remote Access and Account 23
5. Indoor Monitoring and Sound Control 23

IV. CRC Cards **24**
1. Intelligent Security 24
2. Live Surveillance 31
3. System and User Management 32
4. Remote Access and Account 33
5. Indoor Monitoring and Sound Control 35

V. State Diagram **36**
1. Intelligent Security 36
1.1 State diagram for SecurityPolicyManager - Security Mode & Arming 36
1.2 State diagram for IncidentManager 37

1.3 State diagram for SensorMonitoringManager - Physical Intrusion (UC 1.1.1) 38
1.4 State diagram for SensorMonitoringManager - Outdoor Motion (UC 1.1.3) 39
1.5 State diagram for SensorMonitoringManager - Dog Barking (UC 1.1.4) 40
1.6 State diagram for IncidentManager - Configure Alarm Conditions (UC 1.2.1) 41
1.7 State diagram for SecurityPolicyManager - Sensor Bypass (UC 1.3.2) 42
1.8 State diagram for SecurityPolicyManager - Sensor Activation & Deactivation (UC 1.3.3) 43
2. Live surveillance 44
2.1 State diagram for SurveillanceManager 44
2.2 State diagram for User 44
2.3 State diagram for Recording 45
2.4 State diagram for RecordingSettings 46
2.5 State diagram for Storage Repository 47
2.6 State diagram for Camera 47
3. System and User Management 48
3.1 State diagram for SystemDeviceController 48
3.2 State diagram for SystemStatusController 49
3.3 State diagram for SystemLogController 49
3.4 State diagram for UserPermissionController 50
4. Remote Access and Account 51
4.1 State diagram for UserAccount 51
4.2 State diagram for Authentication & Session 51
5. Indoor Monitoring and Sound Control 52
5.1 State diagram for IndoorLightController 52
5.2 State diagram for IndoorAirQualityController 53
5.3 State diagram for IndoorPowerMonitoringController 53
5.4 State diagram for IndoorLightService 54
VI. Design Evaluation 55
1. Architectural Design Matrix 55
1.1 Intelligent Security Subsystem 55
1.1.a Fenton's simple morphology metrics 55
1.2 Live Surveillance Subsystem 55
1.2.a Fenton's simple morphology metrics 55
1.3 System & User Management Subsystem 55
1.3.a Fenton's simple morphology metrics 55
1.4 Remote Access & Account Subsystem 56
1.4.a Fenton's simple morphology metrics 56
1.5 Indoor Monitoring & Device Control Subsystem 56
1.5.a Fenton's simple morphology metrics 56
1.6 Overall 56
1.6.a Fenton's simple morphology metrics 56

1.7 Conclusion of Architectural Design Matrix 56
2. CK Metrics 57
2.1 Intelligent Security - CK Metrics 57
2.2 Live Surveillance- CK Metrics 61
2.3 System & User Management- CK Metrics 62
2.4 Remote Access & Account- CK Metrics 64
2.5 Indoor Monitoring & Device Control- CK Metrics 65
2.6 CK Metrics Conclusion 67
3. MOOD metric 68
a. AHF, AIF, MIF 68
2. Live Surveillance (UC 2.x) 80
3. System & User Management (UC 3.x) 81
4. Remote Access & Account (UC 4.x) 82
6. Coupling Factor Conclusion 85
VII. WHO DID WHAT **85**
VIII. Meeting Log **86**
Meeting 1 86
Meeting 2 87
Meeting 3 88
Meeting 4(final meeting) 90
IX. Glossary 91
1. General Design Concepts 91
2. Primary Devices 91
3. Software Subsystems 91
4. Manager, Controller, Repository, and Service Classes 92
a. Manager Classes 92
b. Domain Entities & State Classes 92
c. Repositories 92
d. Controllers 92
e. Authentication & Session Components 93
5. Security Policies & Rules 93
6. Camera & Recording Operations 93
7. User Roles 94
8. General Technical Concepts 94

I. Overview

1. Introduction

This document, the Software Design Specification (SDS), describes the design model for the 'SafeHome' system. It is a direct translation of the requirements and analysis models presented in the previous Software Requirements Specification **(SRS, CS350 Safehome Project, Team 1).**

Since the design phase is directly connected to the implementation phase, this document focuses on providing a well-formed and concrete design. To portray the system's design, this report presents the overall architectural structure, detailed class diagrams, CRC (Class-Responsibility-Collaboration) cards and state diagrams.

2. Goal

- The design of the SafeHome system is guided by the following core principles:
- Completely follow the requirements and the analysis model as defined in the SRS.
- Achieve the core design principles of low coupling, high cohesion, and modularity.
- Pursue key quality attributes, including testability, integrity, efficiency, maintainability, and reliability.
- Minimize complexity while considering reusability and flexibility for future increments.

3. How the design work proceeded

Our team followed an iterative process to develop the design model:

- Extraction of classes: To ensure the correctness of the design model, we reviewed the nouns and verbs from the use case scenarios in the SRS (as per the method in SEPA, Chapter 8.7) to extract initial classes.
- Creation of **architectural structure**: Based on the extracted classes and the use case scenarios, we created the high-level architectural structure of the SafeHome system.
- Creation of **class diagram**: Using the extracted classes and architecture, we created an initial class diagram with implementation details in mind.
- Creation of **CRC cards**: We developed CRC cards for key classes to define their responsibilities and collaborations.
- Refinement of class diagram: By testing the design with the CRC cards and reviewing the SRS, we refined the class diagram. This step focused on achieving low coupling and high cohesion. Implementation-specific classes (e.g., for database access) were also added.

- Creation of **state diagrams**: We created state diagrams for classes with complex, state-dependent behavior.
- Refinement of class diagram: We reviewed the diagram again, adding missing attributes and methods identified during state modeling.

- Final Refinement of class diagram: We performed a final review of the class diagram from an implementation viewpoint, adding any remaining missing functions or attributes.

4. Assumptions

This design is based on the following assumptions, which are carried over from the SRS:

1. Incremental Development: This design model is for the first increment of the SafeHome system. Features noted in the SRS as for future iterations (e.g., OTA Firmware Updates, advanced data privacy policies, Secure Onboarding) are not included in this design.

2. Platform Focus: The primary user interface for this increment is the mobile application. The design of controllers and views will prioritize this platform. The system is designed around smartphone access – monitoring, controls, and notifications occur via the mobile app.

3. All physical hardware installation and floor plan configuration are assumed to be completed prior to this design.

4. The SDS focuses on software behavior with already-deployed sensors and devices; physical device placement, wiring, and setup are outside the project scope. In other words, the system design assumes a pre-configured environment of sensors/cameras, and does not cover the process of installing or positioning hardware in the home.

5. The security design supports grouping sensors into multiple safety zones (e.g. “room1”, “hallway”) for flexible arming scenarios. A single motion sensor can belong to more than one zone, and if different zone settings conflict, the most recent change takes precedence (**Last-Come, First-Served** policy). In practice, this means that when overlapping zones issue configuration updates to a shared sensor (such as one zone bypassing a sensor while another arms it), the sensor’s state will reflect whichever command was applied last. This assumption ensures a deterministic resolution if zone configurations overlap.

6. System Actor: The 'System administrator' actor referenced in the analysis model is not a human user. It is an abstraction for the SafeHome system itself, which facilitates automated actions and management.

7. The design assumes that administrative actions (e.g. routine maintenance tasks, enforcing security policies) are handled automatically by the software. There is no dedicated human admin actor; the system’s back-end services act as the administrator for managing

devices, users, and configurations.

8. All sensors and cameras communicate with the central SafeHome hub via secure, wireless protocols (e.g. Zigbee, Z-Wave, or Wi-Fi). The design assumes reliable two-way communication within a typical home environment and that these protocols provide encryption and authentication by default.

9. Network-level details (signal range, interference) are handled by the underlying technology and the SDS focuses on higher-level message handling.

10. It is assumed that devices send timely status updates or events to the hub whenever their state changes, enabling the software to react in near real-time.

11. It is assumed that cameras can be remotely enabled/disabled through the hub. The SDS does not implement video/image processing algorithms and it treats the camera as a black box that provides a stream, focusing on forwarding the feed to the mobile app and storing clips when required.

12. The SafeHome architecture assumes an always-on internet connection for cloud services (remote alerts, mobile app access, video streaming), but is designed to tolerate outages gracefully.

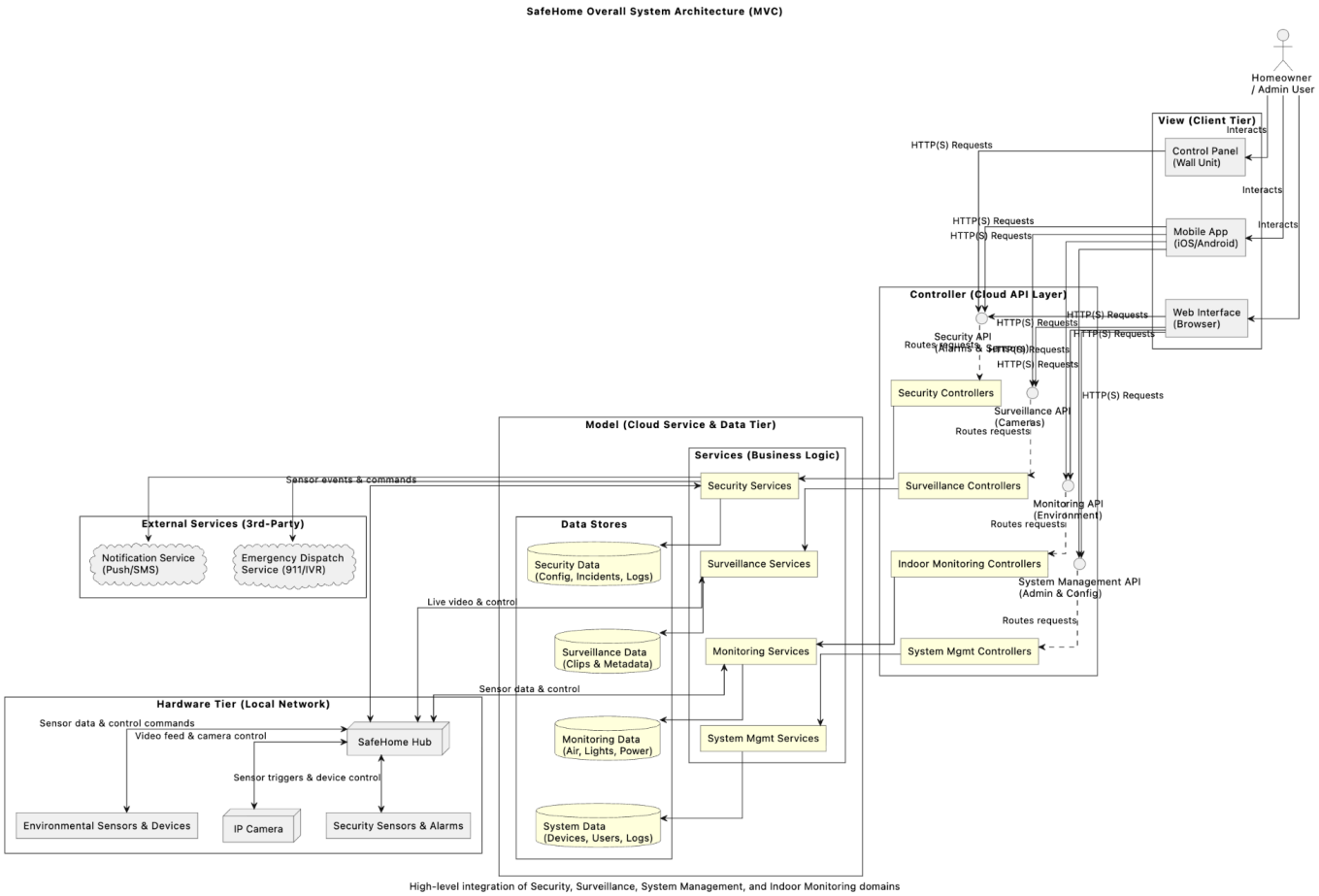
13. We assume power supply is stable (or backed by battery) for the hub and sensors during normal operation; handling of prolonged power failures (beyond noting the event) is outside the first-release scope.

14. SafeHome manages their own user accounts internally for authentication and authorization, which means that we assume no integration with third-party identity providers at this stage; users must sign up and log in through the SafeHome app, and credentials are stored securely within the system.

15. All 'Instant Notifications' (SRS 1.2.1) are delivered as **push notifications** to the user's registered mobile device(s). This relies on cloud-based notification services (e.g., Apple Push Notification Service, Firebase Cloud Messaging) and the mobile app's ability to receive them.

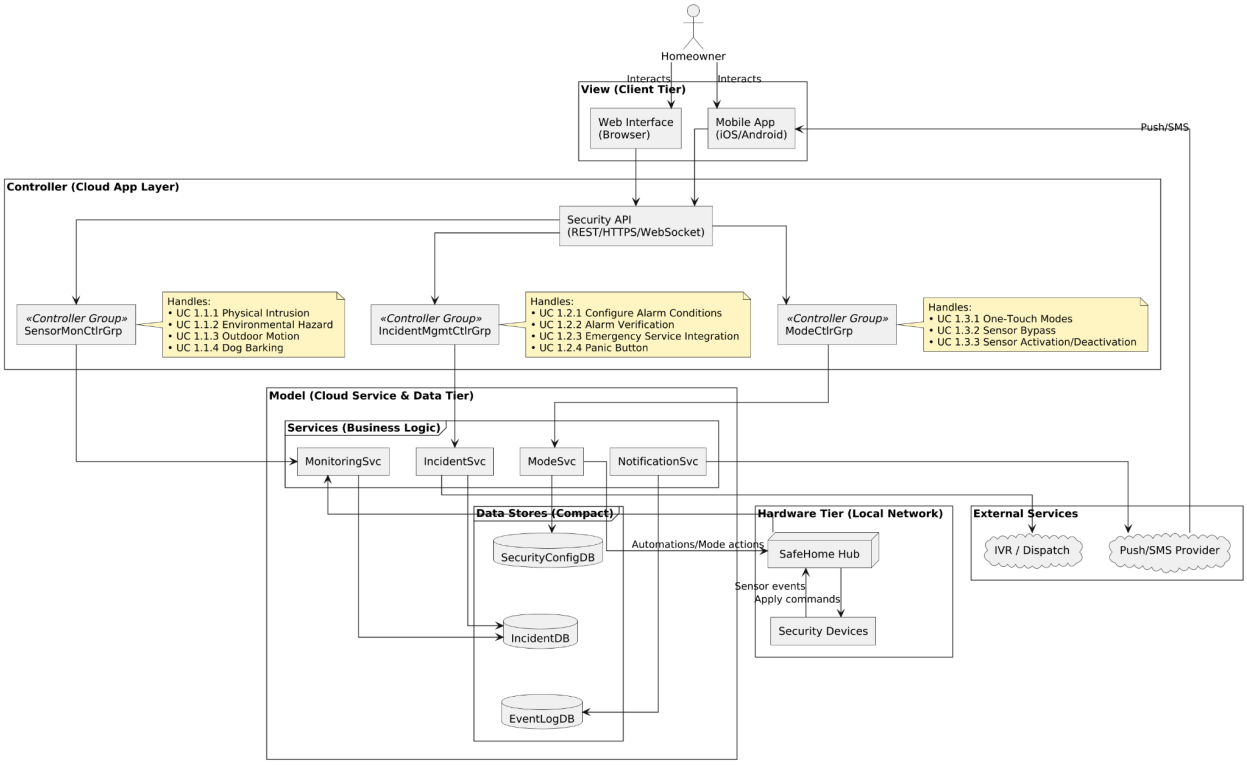
II. Architectural structure

0. Overall Architecture

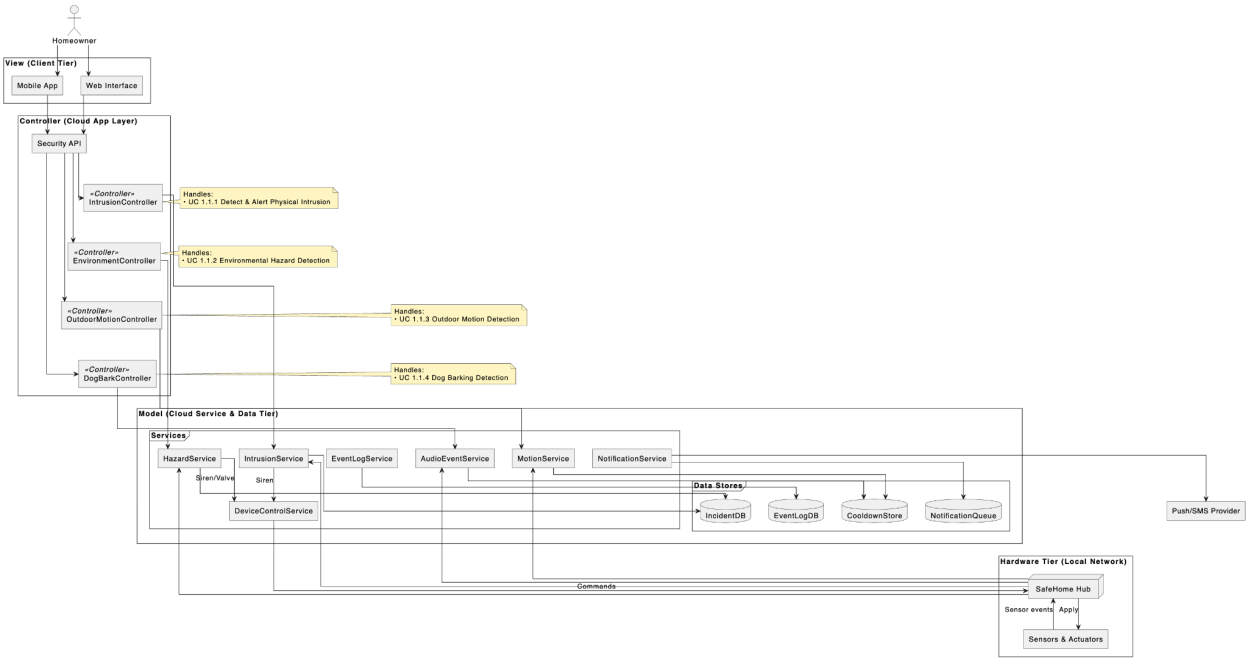


1. Intelligent Security

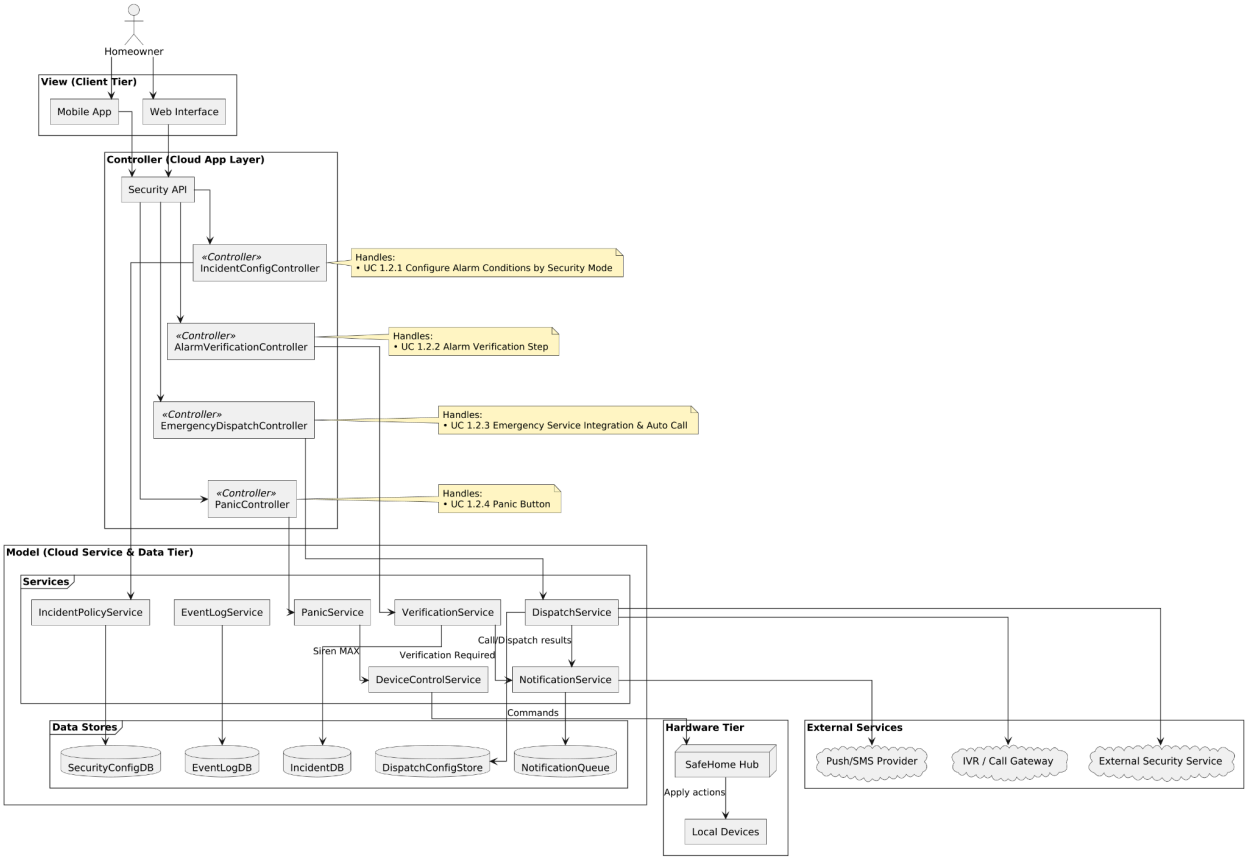
1.1 Overview of Intelligent Security



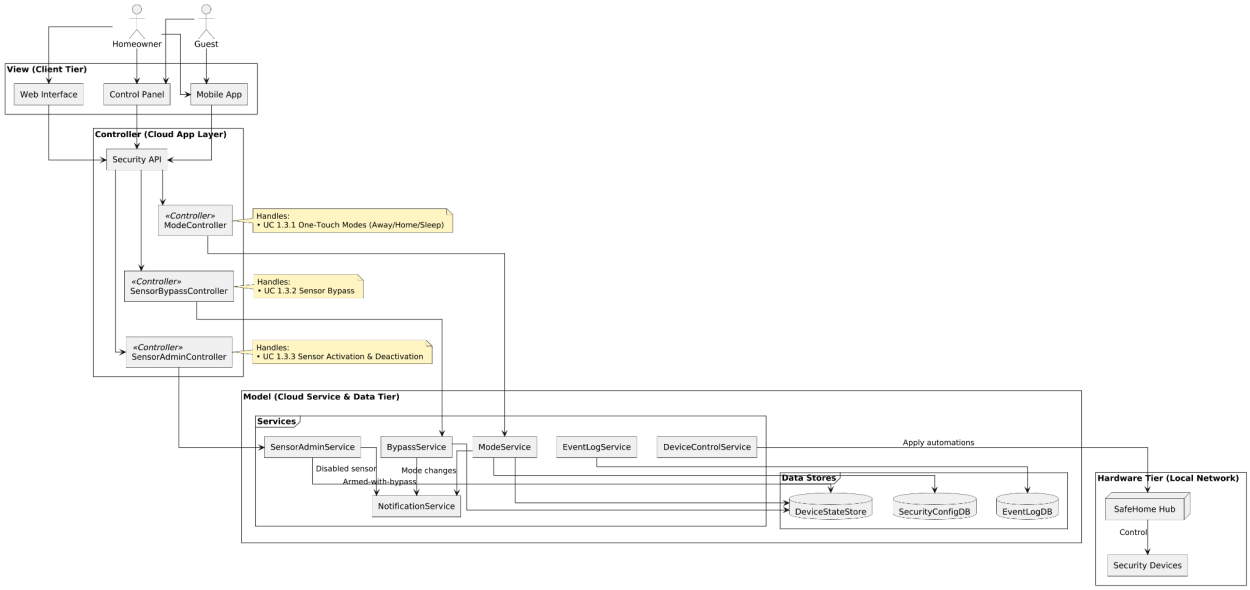
1.2 Sensor Monitoring



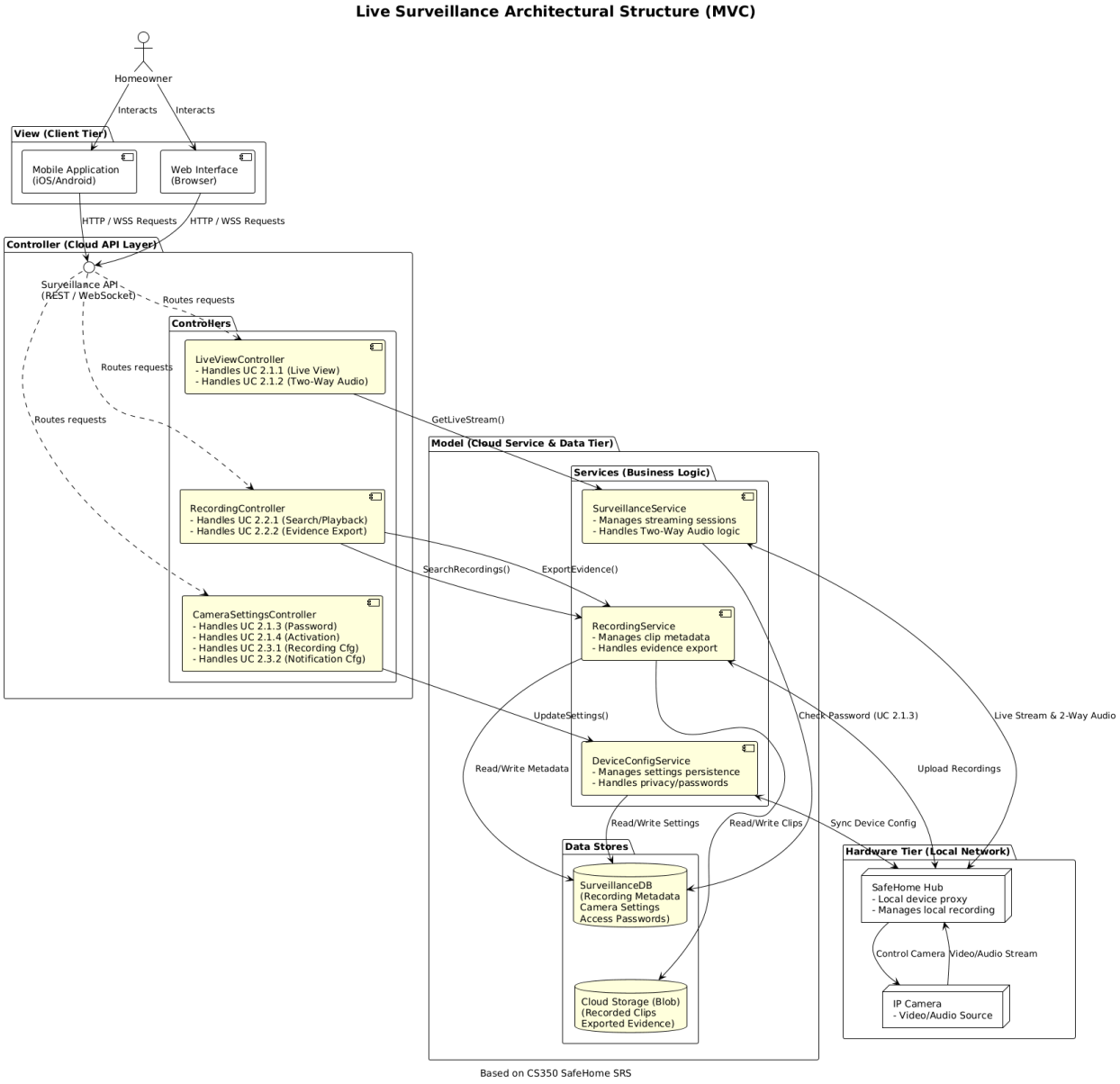
1.3 Incident Management



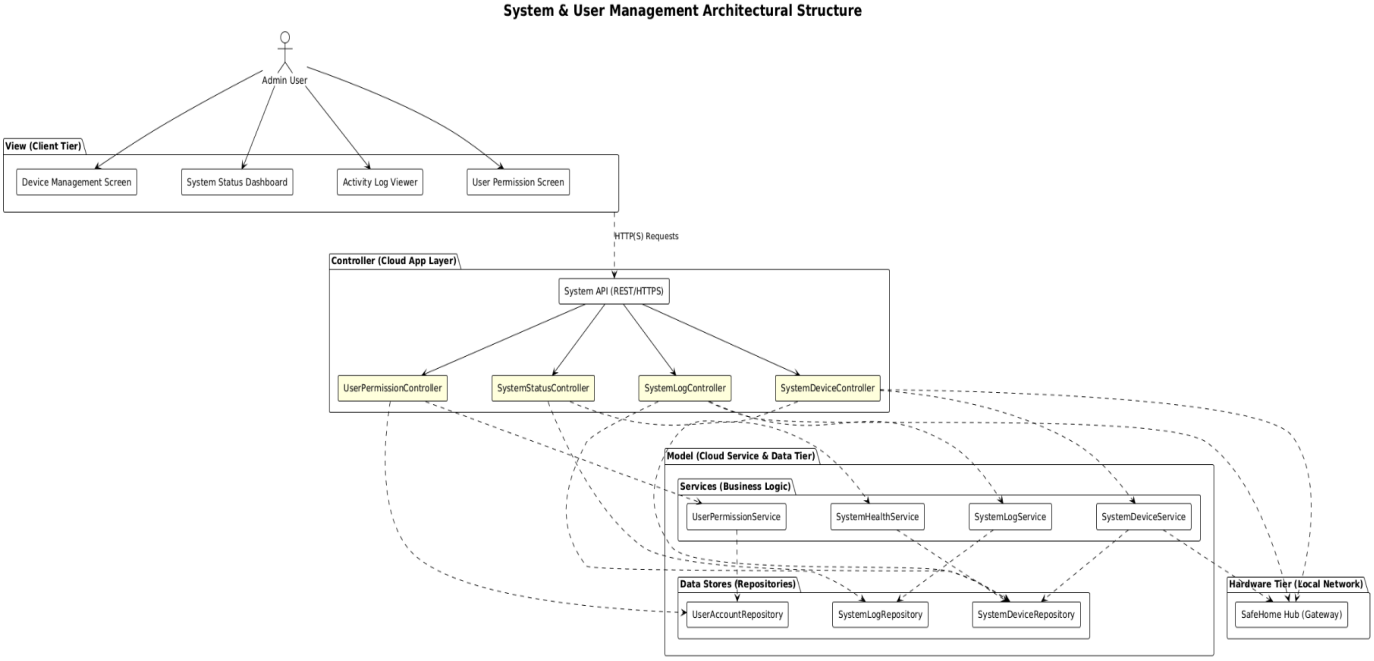
1.4 Security Mode & Device Control



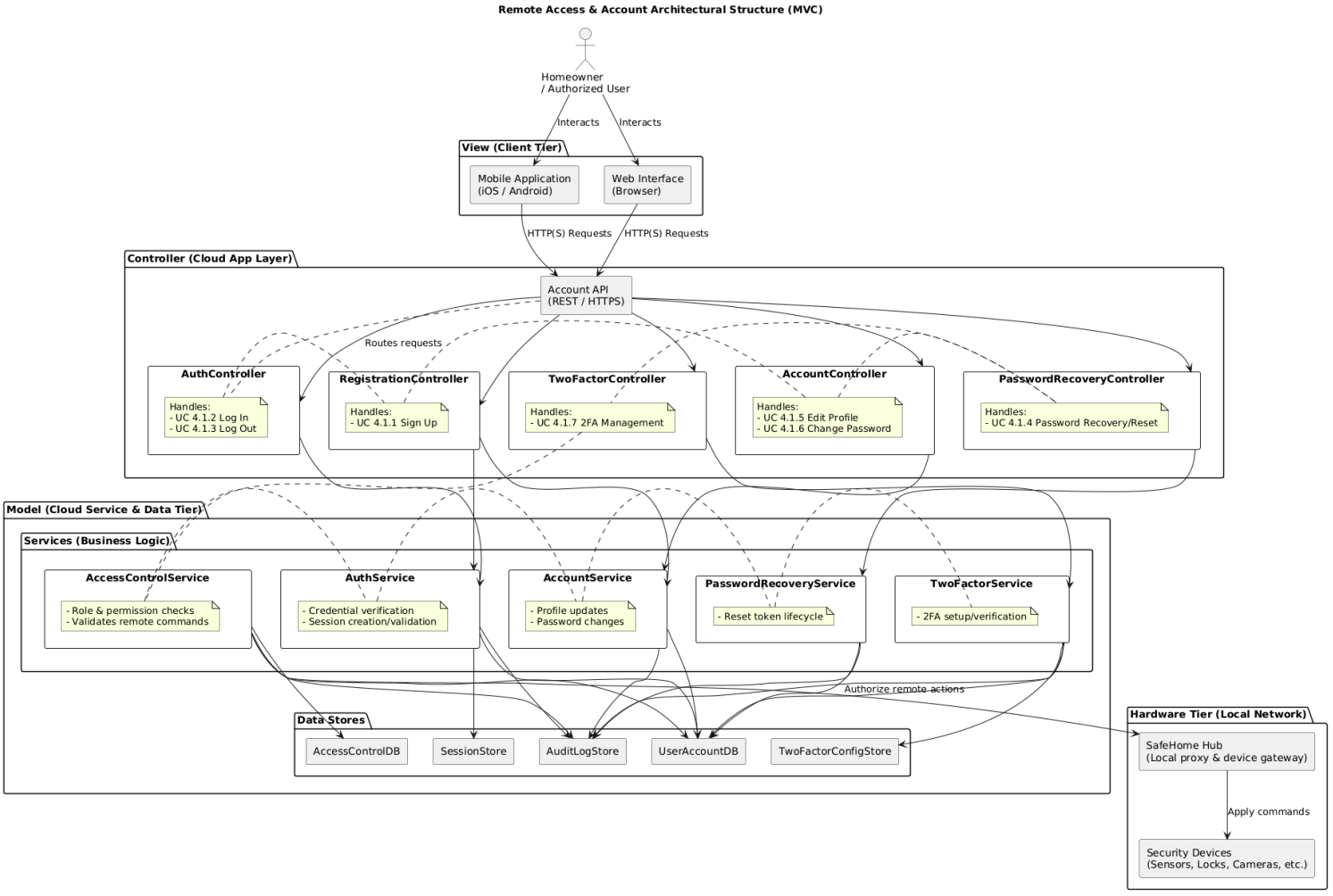
2. Live Surveillance



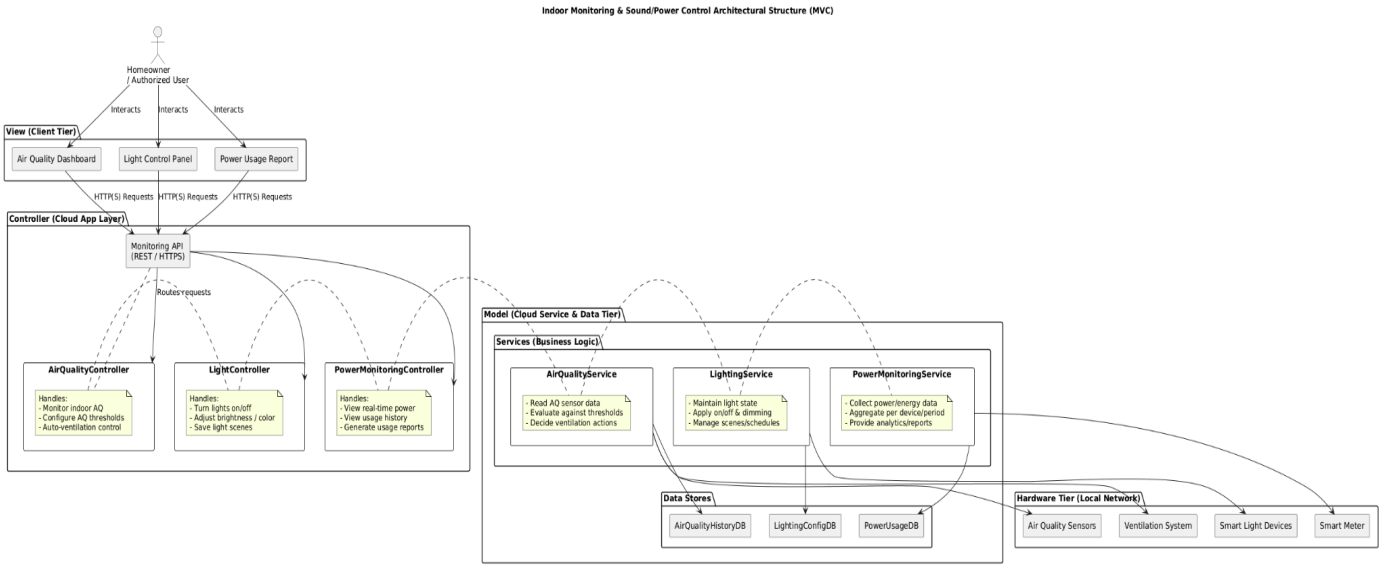
3. System and User Management



4. Remote Access and Account



5. Indoor Monitoring and Sound Control

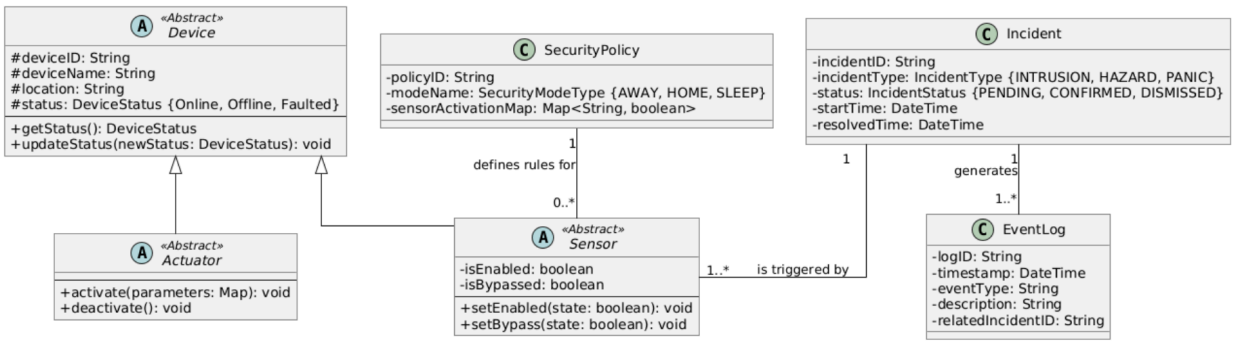


III. Class Diagram

1. Intelligent Security

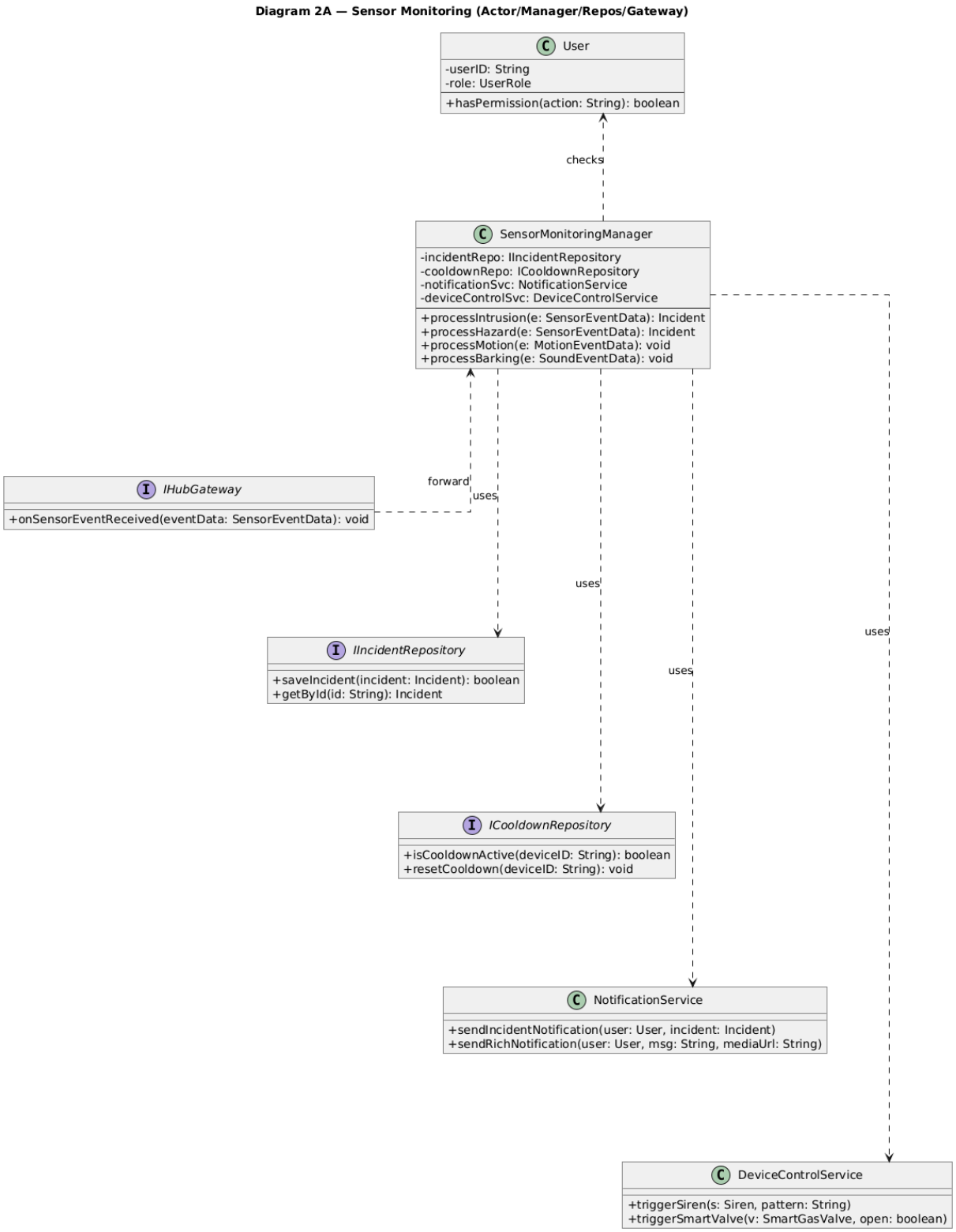
1.1 Security Domain Model (Core Entities)

Diagram 1: Security Domain Model (Core Entities)

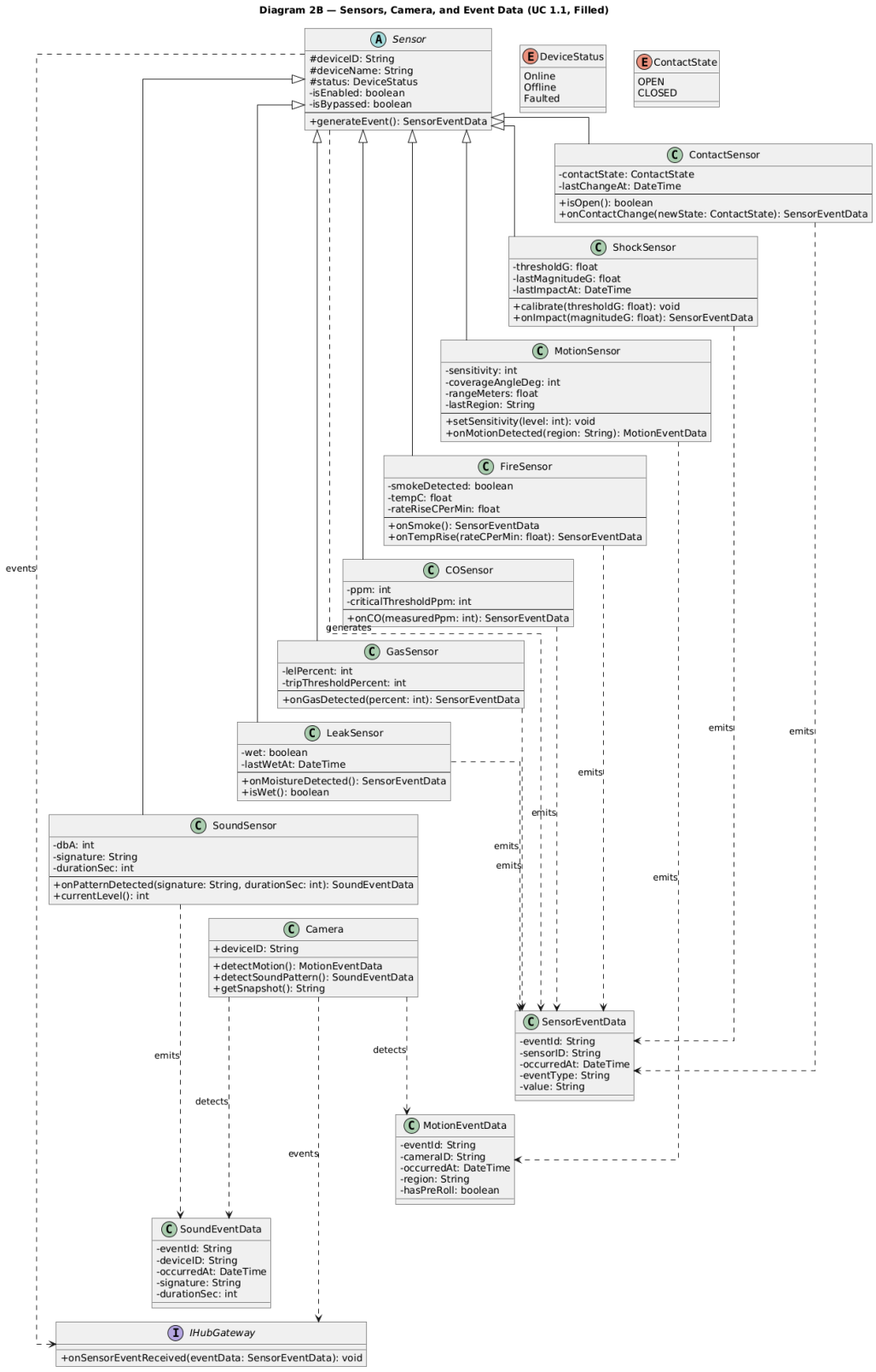


1.2 Sensor Monitoring & Event Processing

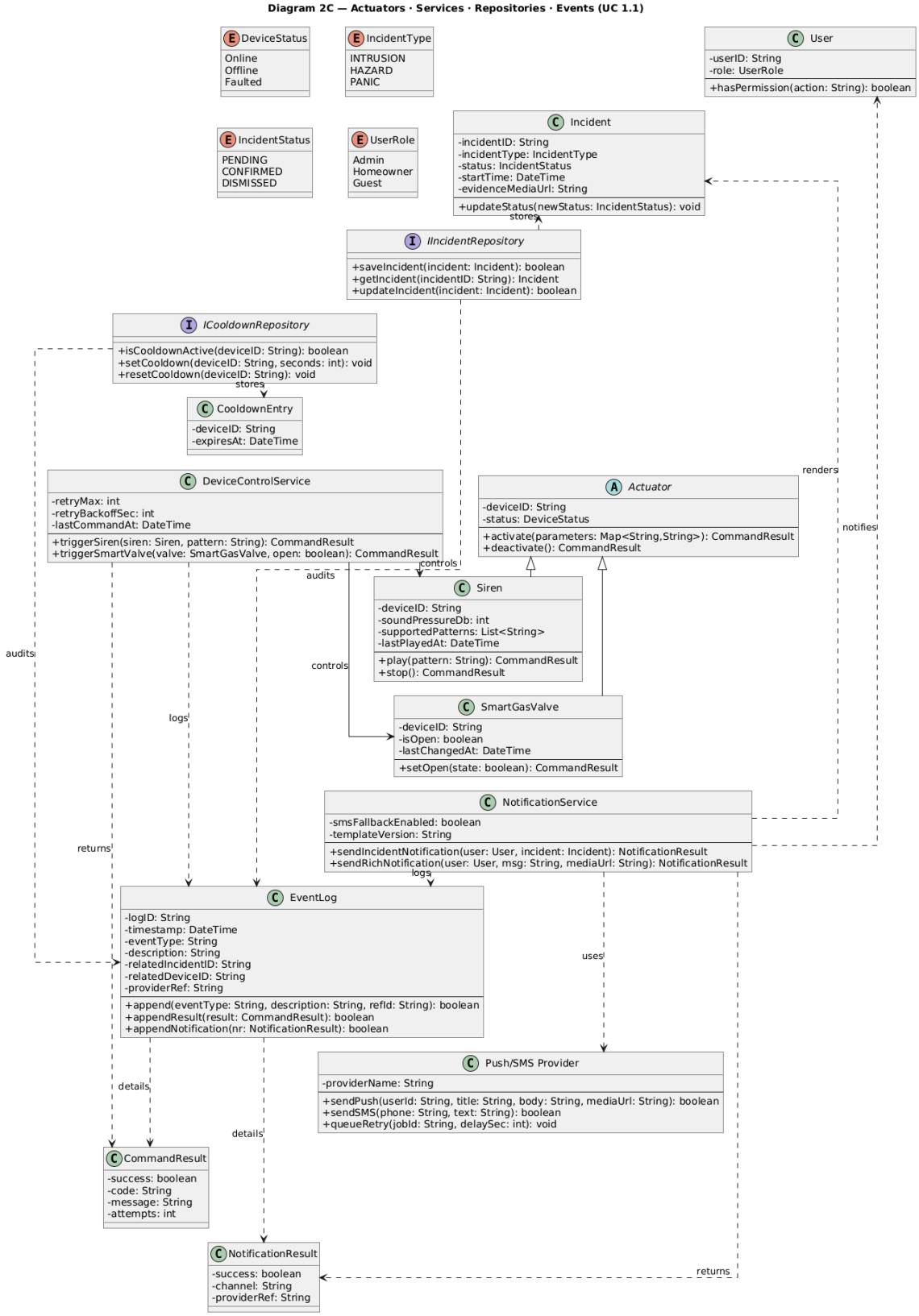
1.2.a Sensor Monitoring (Actor/Manager/Repos/Gateway)



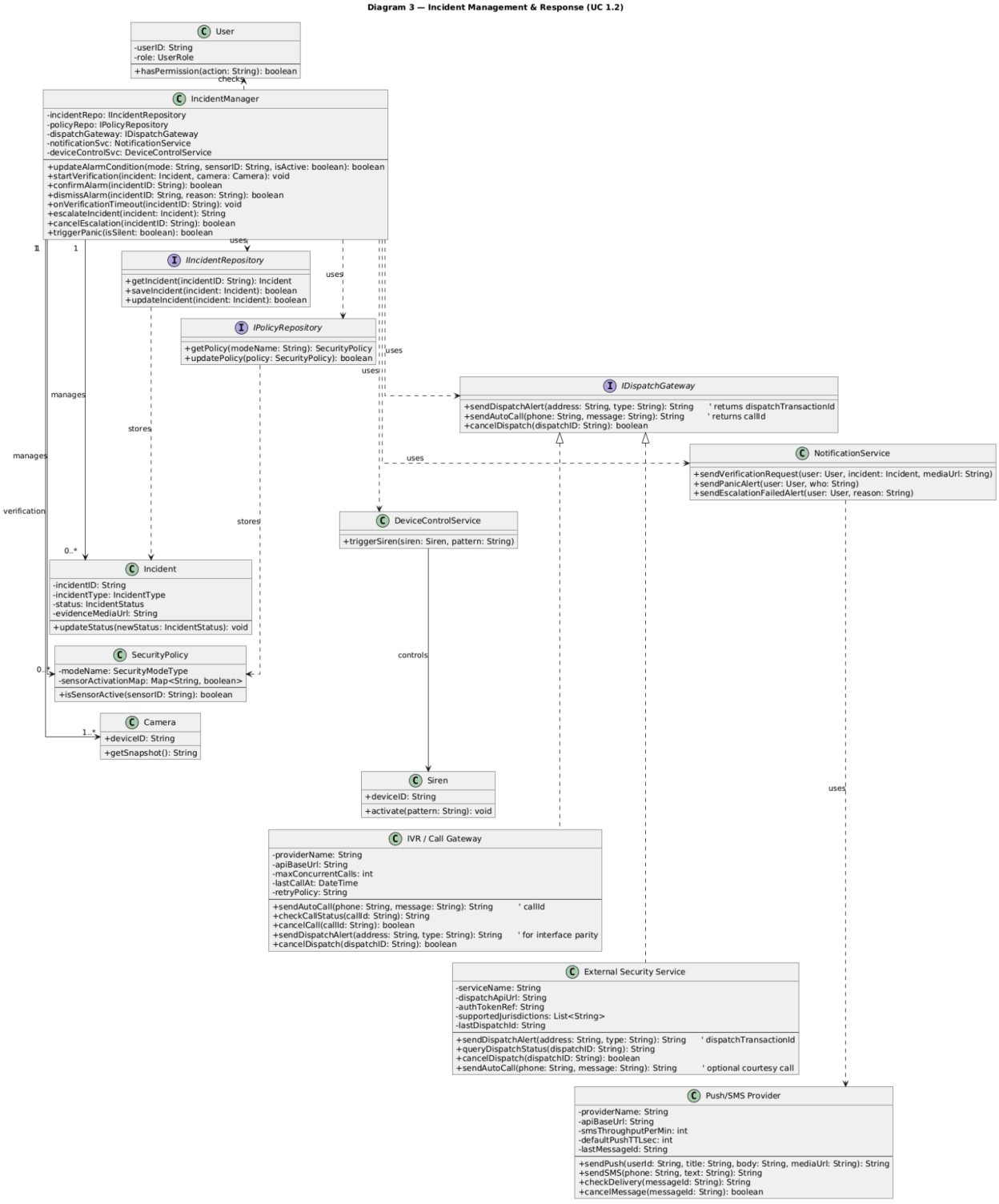
1.2.b Sensors, Camera, and Event Data (UC 1.1)



1.2.c Actuators · Services · Repositories (UC 1.1)

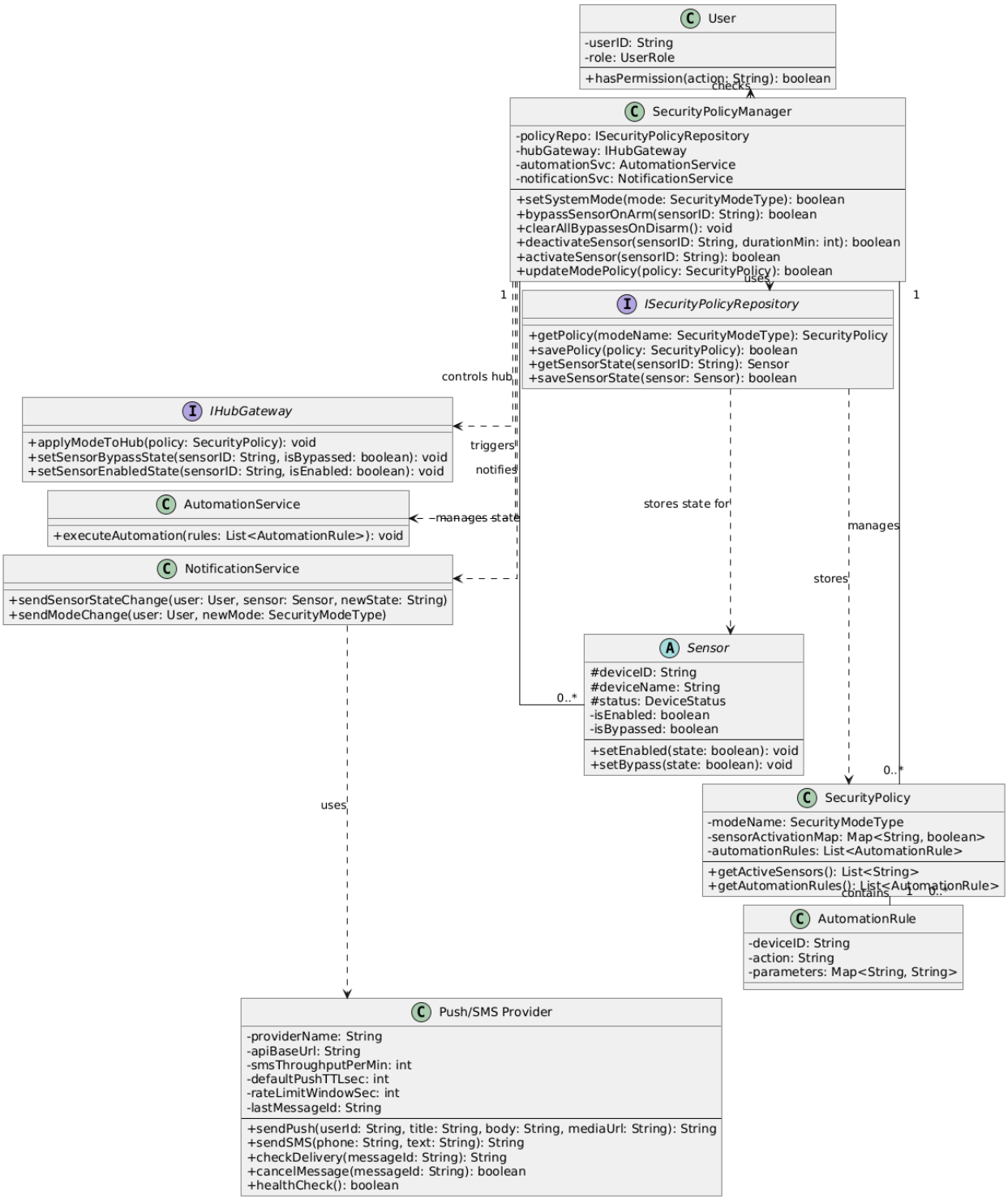


1.3 Incident Management & Response

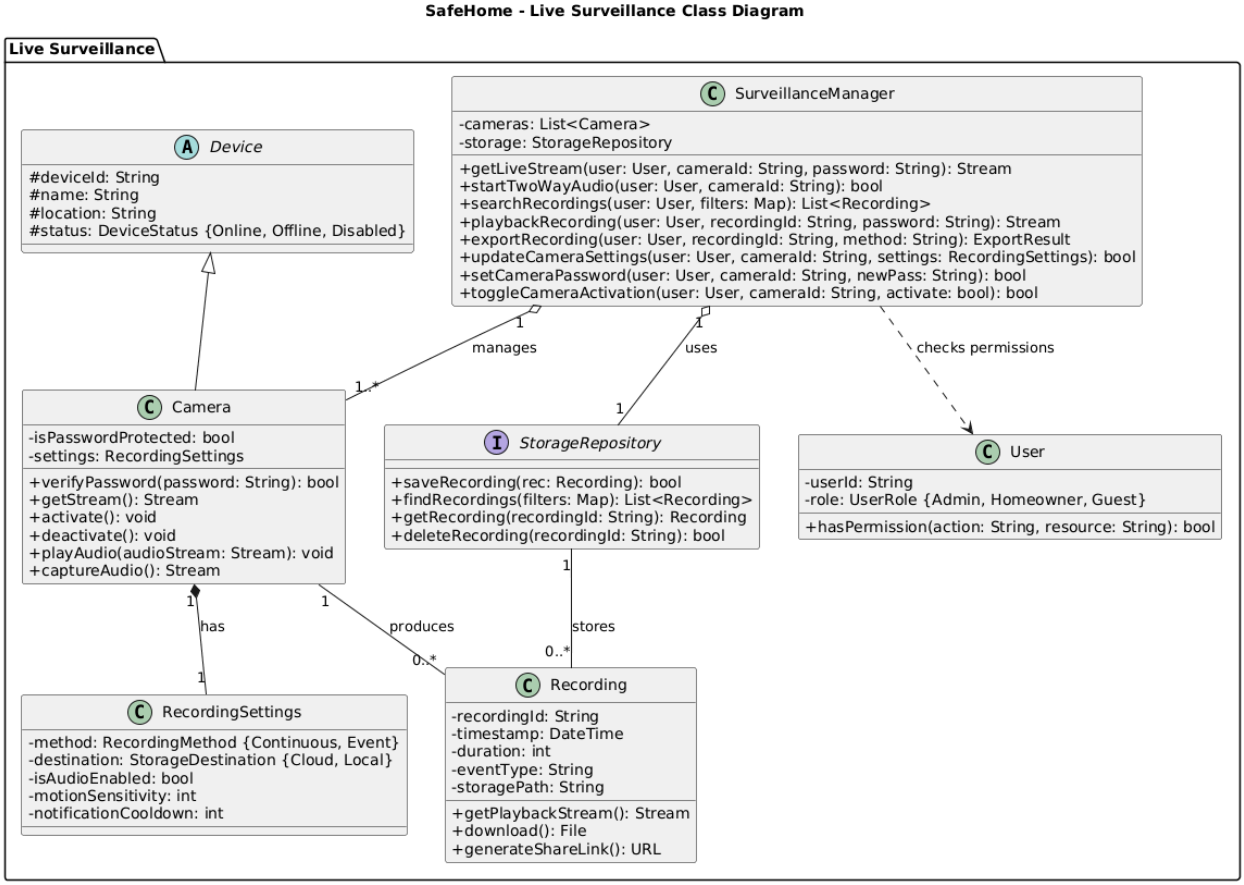


1.4 Security Policy & Control

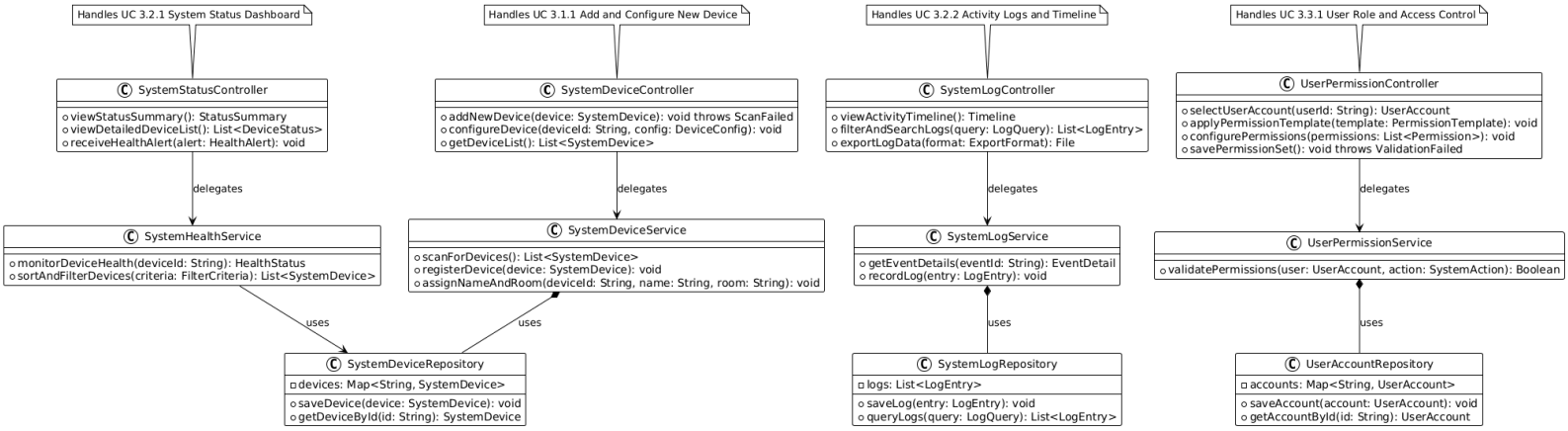
Diagram 4 — Security Policy & Control (UC 1.3)



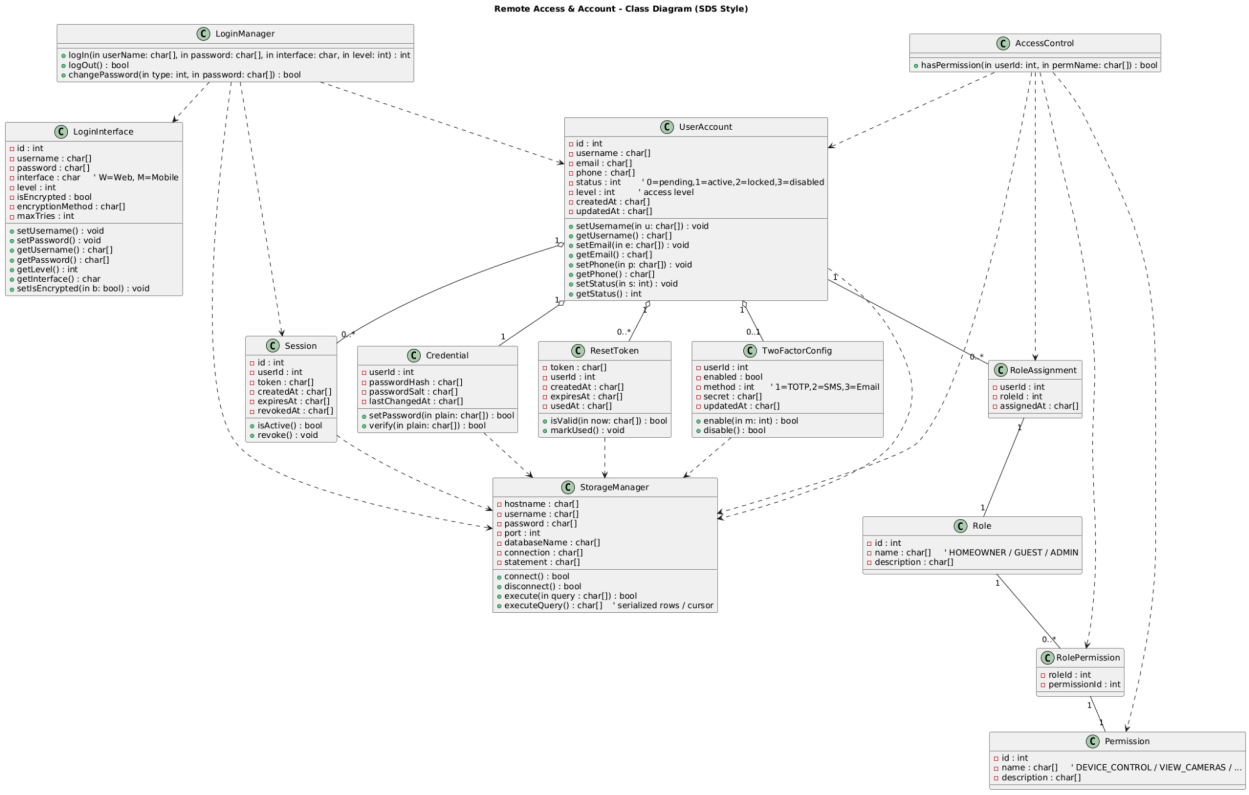
2. Live Surveillance



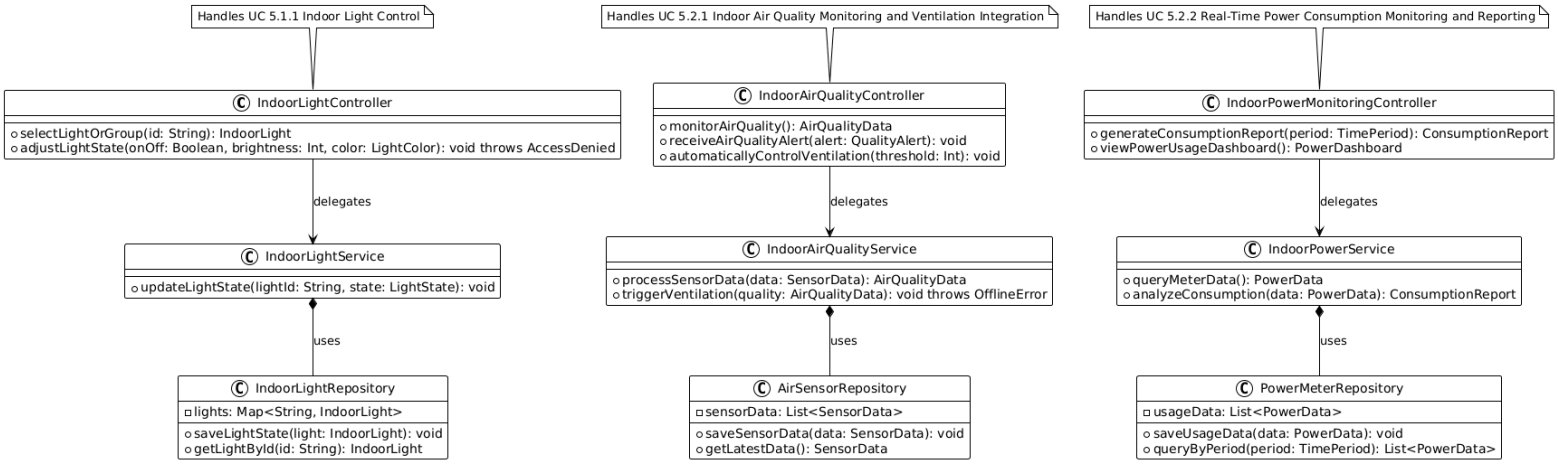
3. System and User Management



4. Remote Access and Account



5. Indoor Monitoring and Sound Control



IV. CRC Cards

1. Intelligent Security

Class	Responsibilities	Collaborators
SensorMonitoringManager	• Ingest sensor/camera events from the hub • Check security mode and apply a 60-second de-duplication window • Classify events (Intrusion, Hazard, Outdoor Motion, Barking) • Create/record Incidents; write activity logs • Trigger local siren patterns and safety actions (e.g., gas valve) • Send rich push/SMS notifications (thumbnail → full media) • Handle cloud/network fallbacks and per-device cooldowns • Route and normalize per-sensor events (Contact/Shock/Motion/Fire/CO/Gas/Leak/Sound) • Write detection/classification/cooldown decisions to EventLog	• User (permission checks) • Sensor, Camera (event sources) • Incident (creates/updates) • ICooldownRepository • IIncidentRepository • DeviceControlService • NotificationService • IHubGateway • EventLog
IncidentManager	• Start alarm verification with snapshot/clip • Handle Confirm / Dismiss and timeout escalation • Execute emergency escalation (IVR calls, external dispatch) • Cancel escalation if disarmed before commit; audit results • Update incident status lifecycle • Notify all authorized Homeowners	• User • Incident • Camera • IDispatchGateway • NotificationService • DeviceControlService • IIncidentRepository • IPolicyRepository
SecurityPolicyManager	• Set system mode (Away/Home/Sleep) with exit-delay handling • Sync mode profile to hub; show “Sync Pending” when offline • Execute Simulated Occupancy automation rules • Perform Sensor Bypass during arming (non–life-safety only)	• User • SecurityPolicy, AutomationRule • ISecurityPolicyRepository • IHubGateway • AutomationService • NotificationService

	• Activate/Deactivate sensors (timed or indefinite) with audit trail • Send notifications for mode/sensor state changes • Persist policy & device state; clear bypasses on Disarm	
User	• Represent a Homeowner/Admin/Guest account • Hold role used for authorization • Answer permission checks for managers	• SensorMonitoringManager • IncidentManager • SecurityPolicyManager
Device (Abstract)	• Define common attributes (deviceID, name, location, status) • Provide getStatus / updateStatus operations	• Sensor, Actuator, Camera (inherit)
Sensor (Abstract)	• Report events to the hub • Maintain enabled/disabled and bypassed flags • Allow administrative enable/disable and bypass toggles	• Device (inherits from) • SecurityPolicyManager (manages state) • SensorMonitoringManager (consumes events)
Camera	• Detect motion or audio patterns (when supported) • Provide snapshot/clip for verification & notifications • Offer live preview for incident review • Hold per-device sensitivity and cooldown settings • Detect sound patterns when supported (detectSoundPattern); provide snapshot for verification	• SensorMonitoringManager • IncidentManager • NotificationService
Actuator (Abstract)	• Provide activate/deactivate interface for hardware actions • Serve as base for sirens/valves and similar devices	• DeviceControlService

Siren	• Play distinct alarm patterns; report activation outcome • Fail safely and log when offline/unavailable	• DeviceControlService • IncidentManager, SensorMonitoringManager
SmartGasValve	• Shut off or restore gas flow on safety events • Report command success/failure	• DeviceControlService • SensorMonitoringManager
SecurityPolicy	• Store per-mode sensor activation map and automation rules • Answer “Is the sensor active in this mode?” • Track version/audit information for edits	• AutomationRule (contains) • SecurityPolicyManager • ISecurityPolicyRepository
AutomationRule	• Describe a device action for a given mode (deviceID/action/params) • Execute during mode transitions or schedules	• SecurityPolicy (owns) • AutomationService (executes)
Incident	• Represent a security event (type, status, timestamps, evidence) • Transition across Pending/Verified/Confirmed/Escalated/Closed • Link triggering sensors and related logs/media	• IncidentManager, SensorMonitoringManager • IIncidentRepository
EventLog	• Persist chronological records for events, commands, results • Store metadata (timestamp, type, relatedIncidentID) • Support audits and timeline queries • Append command results (appendResult(CommandResult)) and notification results (appendNotification(NotificationResult)) • Store external providerRef for audit	• SensorMonitoringManager • IncidentManager • SecurityPolicyManager • DeviceControlService • NotificationService • IIncidentRepository • ICooldownRepository • CommandResult • NotificationResult

IHubGateway (Interface)	• Abstract hub communication and command application • Forward sensor/camera events to managers • Apply mode, bypass, enable/disable to devices	• SensorMonitoringManager • SecurityPolicyManager
IIncidentRepository (Interface)	• Persist, retrieve, and update Incident objects atomically • Support idempotent writes and audit metadata • Provide getById(id)/getIncident(incidentID) and atomic updateIncident	• IncidentManager • SensorMonitoringManager • Incident • EventLog
ISecurityPolicyRepository (Interface)	• Load/save SecurityPolicy versions and sensor state • Provide helpers for edit leases/version checks	• SecurityPolicyManager
ICooldownRepository (Interface)	• Track per-device cooldown windows • Answer active/expired cooldown queries; reset after notify • Provide setCooldown(deviceID, seconds) and manage CooldownEntry	• SensorMonitoringManager • CooldownEntry • EventLog
IDispatchGateway (Interface)	• Initiate automated phone calls (IVR) • Submit/cancel dispatch requests to external services • Return transaction IDs and outcomes • Dispatch request/cancel and automated call; status queries are available via concrete implementations (IVR/External Service)	• IncidentManager • IVR / Call Gateway, External Security Service (implementers)
IPolicyRepository (Interface)	• Retrieve the security mode profile with getPolicy(modeName). • Update the policy with updatePolicy(policy) (atomically stored).	• IncidentManager (reads/modifies mode policies during alarm validation) • SecurityPolicy

NotificationService (Service)	• Send push/SMS notifications (verification, alerts, results) • Provide rich notifications with thumbnails and deep links • Apply SMS/queue fallbacks on failure • Transmission status inquiry/retry policy (based on provider SLA) • Return NotificationResult for verification/alert messages • Support provider status checks and cancellation via external provider • Log notification outcomes	• SensorMonitoringManager • IncidentManager • SecurityPolicyManager • External Push/SMS Provider • NotificationResult • EventLog
DeviceControlService (Service)	• Trigger siren patterns; control smart valves/actuators • Report success/failure for logging and retries • Coordinate with hub via gateway • Return CommandResult for all device commands • Apply retry/back-off • Log command outcomes to EventLog	• Siren, SmartGasValve • SensorMonitoringManager • IncidentManager • IHubGateway • CommandResult • EventLog
AutomationService (Service)	• Execute AutomationRules for the selected mode • Sequence and retry device commands; report outcomes	• SecurityPolicyManager • IHubGateway • NotificationService (optional feedback)
Push/SMS Provider (External)	• Send sendPush() and sendSMS(). • Check delivery status with getDeliveryStatus() and monitor SLA. • Cancel messages, check delivery status, expose health checks; operate within throughput/rate-limit constraints	• NotificationService
IVR / Call Gateway (External)	• Automated voice call initiation, interruption (cancellation), and result reporting.	• IncidentManager (called via IDispatchGateway on behalf of)

	• Initiate/cancel automated calls; check call status; (for parity) send/cancel dispatch when supported	
External Security Service (External)	• Receive dispatch requests, return transaction ID, process dispatch cancellations, and report results. • Send/cancel dispatch and query dispatch status; optionally place courtesy auto-calls	• IncidentManager
ContactSensor	• Generate contact-change events (onContactChange) • answer current open/closed state (isOpen)	• IHubGateway • SensorMonitoringManager
ShockSensor	• Calibrate vibration threshold (calibrate) • emit impact events with measured G-force (onImpact)	• IHubGateway • SensorMonitoringManager
MotionSensor	• Configure sensitivity (setSensitivity) • emit region-aware motion events (onMotionDetected)	• IHubGateway • SensorMonitoringManager
FireSensor	• Emit smoke and rapid temperature-rise events (onSmoke, onTempRise)	• IHubGateway • SensorMonitoringManager
COSensor	• Measure CO ppm and emit threshold events (onCO)	• IHubGateway • SensorMonitoringManager
GasSensor	• Detect combustible gas (LEL %) and emit alerts (onGasDetected)	• IHubGateway

		• SensorMonitoringManager
LeakSensor	• Detect moisture and emit water-leak events (onMoistureDetected) • report wet/dry state (isWet)	• IHubGateway • SensorMonitoringManager
SoundSensor	• Detect pattern/duration-based acoustic events (onPatternDetected) • report current dB level (currentLevel)	• IHubGateway • SensorMonitoringManager
CommandResult	• Encapsulate device command outcome (success/code/message/attempts)	• DeviceControlService • EventLog
NotificationResult	• Encapsulate notification outcome (channel/providerRef/success)	• NotificationService • EventLog
CooldownEntry	• Persist per-device cooldown expiration	• ICooldownRepository

2. Live Surveillance

Class	Responsibilities	Collaborators
Surveillance Manager	● Coordinate access to live video streams ● Verify user permissions for specific functions ● Initiate and manage two-way audio sessions ● Search for recordings based on filters; ● Coordinate playback of recordings; ● Handle recording export and sharing requests ● Update camera settings (e.g., recording, notifications) ● Manage camera privacy (set password, activate/deactivate)	● User (to check permissions) ● Camera (to get streams, update settings) ● StorageRepository (to search/retrieve recordings) ● Recording (to initiate playback/export)
User	● Represent the system's Homeowner or Guest ● Hold the user's role (e.g., Admin, Homeowner); ● Determine if the user has permission to perform an action (e.g., change settings, view a password-protected camera)	● SurveillanceManager (provides context for permission checks)
Device (Abstract)	● Define common attributes for all hardware (ID, Name, Location) ● Maintain operational status (Online, Offline, Disabled)	● Camera (inherits from this class)
Camera	● Provide the live video stream ● Verify if an access password is correct ● Start/stop video streaming and recording on command ● Activate or deactivate (Privacy Mode) ● Play incoming audio (for two-way talk) ● Capture local audio (for two-way talk) ● Hold its specific recording and notification settings	● Device (inherits from) ● SurveillanceManager (is managed by) ● RecordingSettings (holds) ● Recording (is the source of)

Recording Settings	• Store the recording method (Continuous, Event) • Store the recording destination (Cloud, Local) • Store whether audio is enabled • Store device sensitivity and notification cooldown	• Camera (is an attribute of)
Recording	• Represent a recorded video clip • Store metadata (timestamp, duration, event type) • Provide a video stream for playback • Provide a file for download • Generate a secure link for sharing	• StorageRepository (is stored by) • SurveillanceManager (is managed by)
Storage Repository (Interface)	• Abstract the storage location (local or cloud) • Save new recording clips • Find and retrieve recordings based on filters (date, camera, event) • Delete recordings (e.g., per retention policy)	• SurveillanceManager (uses this interface) • Recording (manages this object)

3. System and User Management

Class	Responsibilities	Collaborators
System Device Controller	• Handle device scan/addition/configuration • Manage exceptions like scan failure	• SystemDeviceService, • SystemDeviceRepository • SafeHome Hub (Gateway)
System Status Controller	• Display summaries/lists; Handle health alerts	• SystemHealthService • SystemDeviceRepository
System Log Controller	• Timeline view/filter/search/export	• SystemLogService • SystemLogRepository • SafeHome Hub (for sync)
User Permission Controller	• Manage account selection/permissions/templates • Validate roles	• UserPermissionService • UserAccount Repository

System Device Service	• Execute scanning/registration/assignment logic	• SystemDeviceRepository • SafeHome Hub
System Health Service	• Monitor/sort/filter health/status	• SystemDeviceRepository
System Log Service	• Query/record logs	• SystemLogRepository
User Permission Service	• Configure/validate permissions	• UserAccountRepository
System Device Repository	• Persist/retrieve device data	• N/A (data layer)
User Account Repository	• Persist/retrieve accounts/roles	• N/A (data layer)
System Log Repository	• Persist/query logs	• N/A (data layer)

4. Remote Access and Account

Class	Responsibilities	Collaborators
UserAccount	• Maintain user identity and contact information • Track timestamps, account status, and level • Manage linked two-factor configuration • Handle password-reset tokens • Maintain role and permission links • Persist data to the storage layer	• Credential • Session • TwoFactorConfig • ResetToken • RoleAssignment • StorageManager
Credential	• Store password hash and salt • Verify password correctness • Enforce password policy	• UserAccount • StorageManager
Session	• Manage token, expiry, and revocation • Determine active / expired state • Terminate user session on logout	• UserAccount • StorageManager
TwoFactorConfig	• Enable / disable two-factor authentication • Store method and secret key • Validate verification codes	• UserAccount • StorageManager

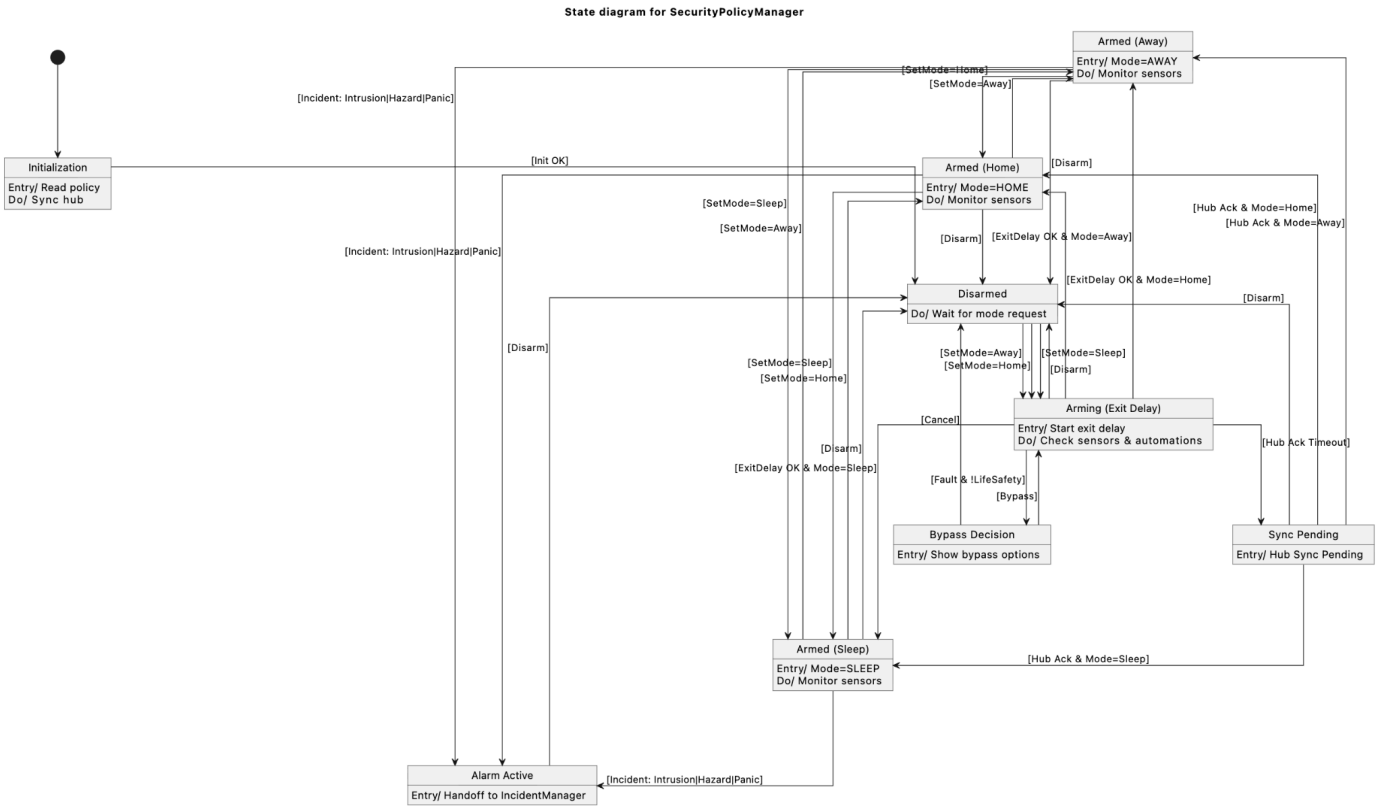
ResetToken	• Create new reset token • Check validity / expiry • Mark token as used	• UserAccount • StorageManager
Role	• Create and describe user roles • Manage mapping between roles and permissions • Persist role definitions	• RoleAssignment • RolePermission • StorageManager
Permission	• Define and document permission scope • Associate with roles • Persist permission data	• RolePermission • Role • StorageManager
RoleAssignment	• Assign / remove role to / from user • Track assignment timestamps • Query user's roles	• UserAccount • Role • StorageManager
RolePermission	• Attach / detach permissions to roles • Manage permission hierarchy • Persist mapping data	• Role • Permission • StorageManager
LoginInterface	• Accept username, password, and interface type • Limit login attempts • Encrypt transmitted credentials	• LoginManager
LoginManager	• Authenticate users • Manage sessions and credentials • Change passwords and enforce policy • Delegate access checks	• LoginInterface • UserAccount, Credential, Session • StorageManager • AccessControl
AccessControl	• Verify if user has permission • Validate role and permission combinations • Authorize or reject remote commands	• UserAccount • RoleAssignment, RolePermission • StorageManager
StorageManager	• Connect / disconnect to data store • Execute queries and updates • Manage serialization of objects	• All domain classes

5. Indoor Monitoring and Sound Control

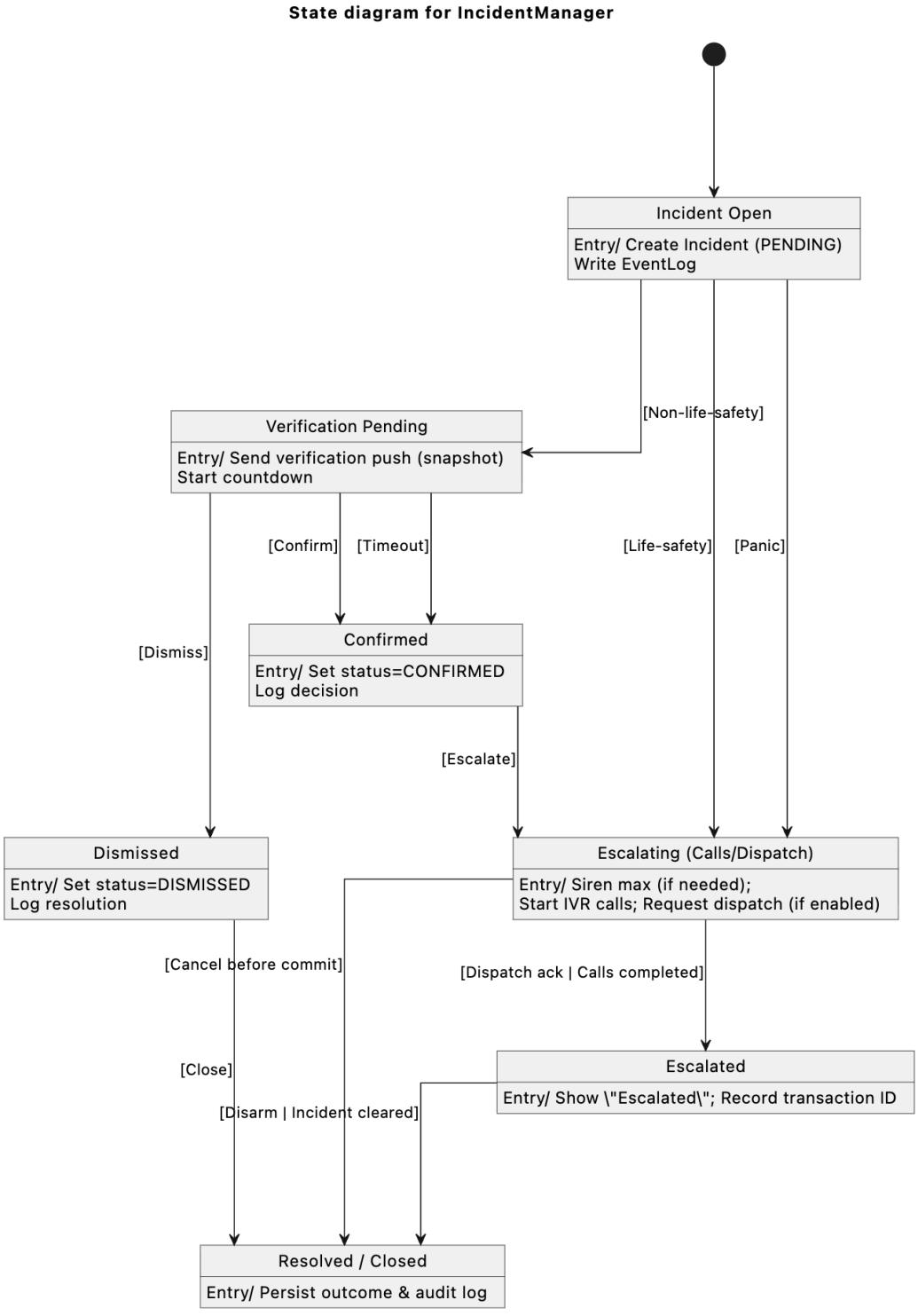
Class	Responsibilities	Collaborators
Indoor Light Controller	● Select/adjust light/group states (UC 5.1.1) ● Role-based access checks	● IndoorLightService ● IndoorLightRepository ● SafeHome Hub (Gateway)
Indoor AirQuality Controller	● Monitor quality/alerts/auto-ventilation (UC 5.2.1)	● IndoorAirQualityService ● AirSensorRepository ● SafeHome Hub
Indoor PowerMonitoring Controller	● Report generation/dashboard views (UC 5.2.2) ● Threshold monitoring	● IndoorPowerService ● PowerMeterRepository ● SafeHome Hub
Indoor Light Service	● Process/update light control logic	● IndoorLightRepository ● SafeHome Hub
Indoor AirQuality Service	● Analyze data/trigger ventilation ● Handle offline	● AirSensorRepository ● SafeHome Hub
Indoor Power Service	● Query/analyze power data	● PowerMeterRepository ● SafeHome Hub
Indoor Light Repository	● Persist/retrieve light states	● N/A (data layer)
AirSensor Repository	● Persist/retrieve sensor data	● N/A (data layer)
Power Meter Repository	● Persist/query usage data	● N/A (data layer)

V. State Diagram
1. Intelligent Security

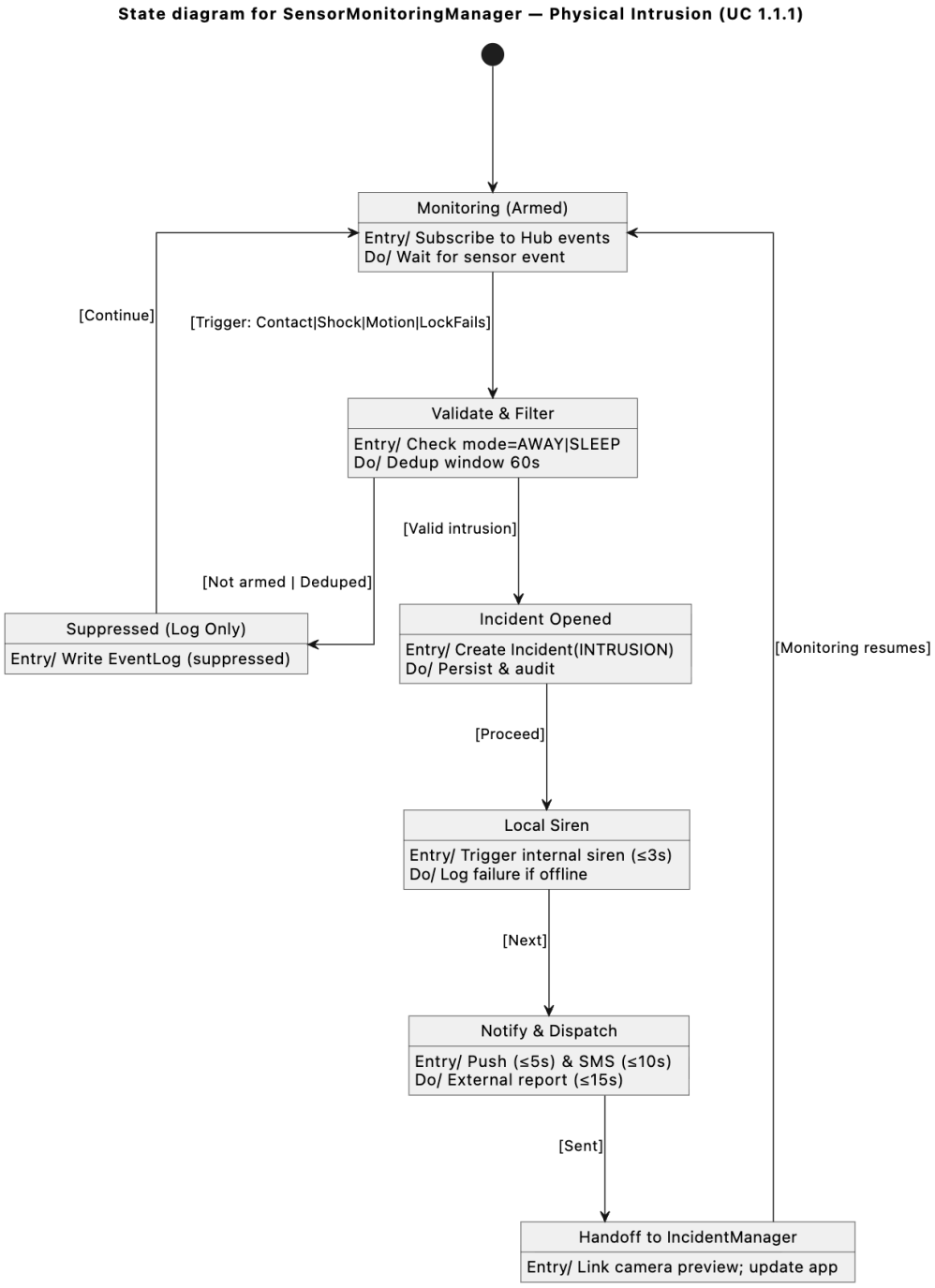
1.1 State diagram for SecurityPolicyManager - Security Mode & Arming



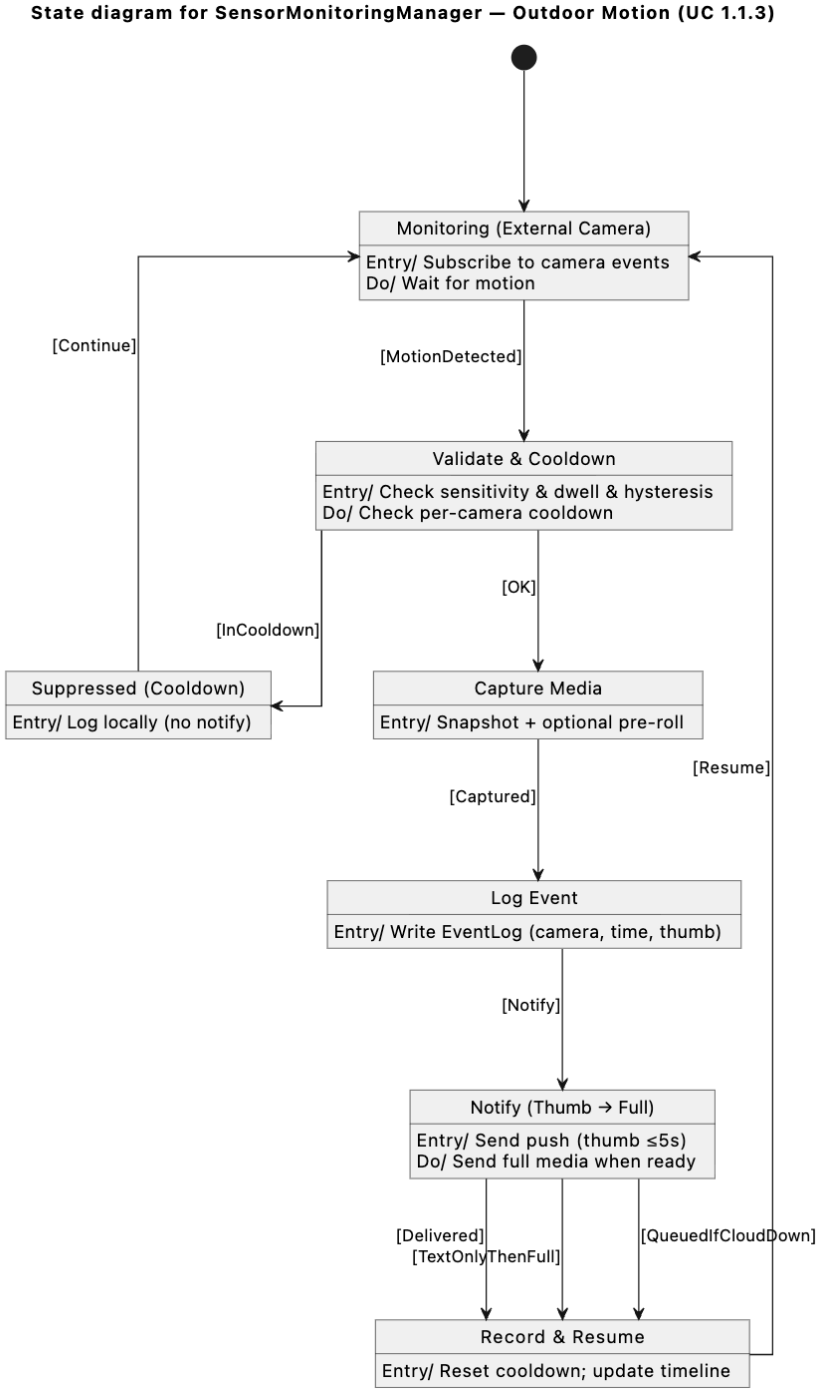
1.2 State diagram for IncidentManager



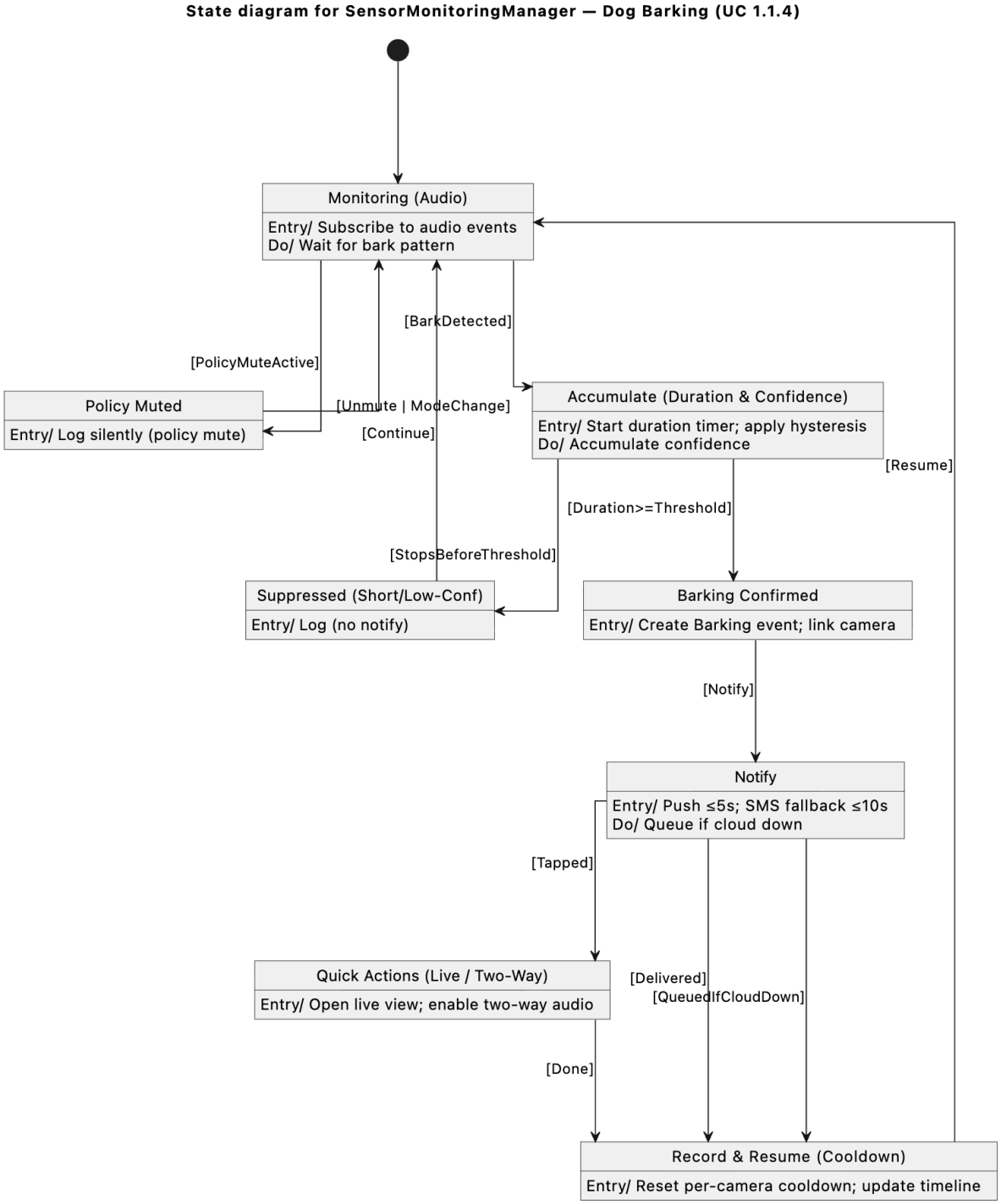
1.3 State diagram for SensorMonitoringManager - Physical Intrusion (UC 1.1.1)



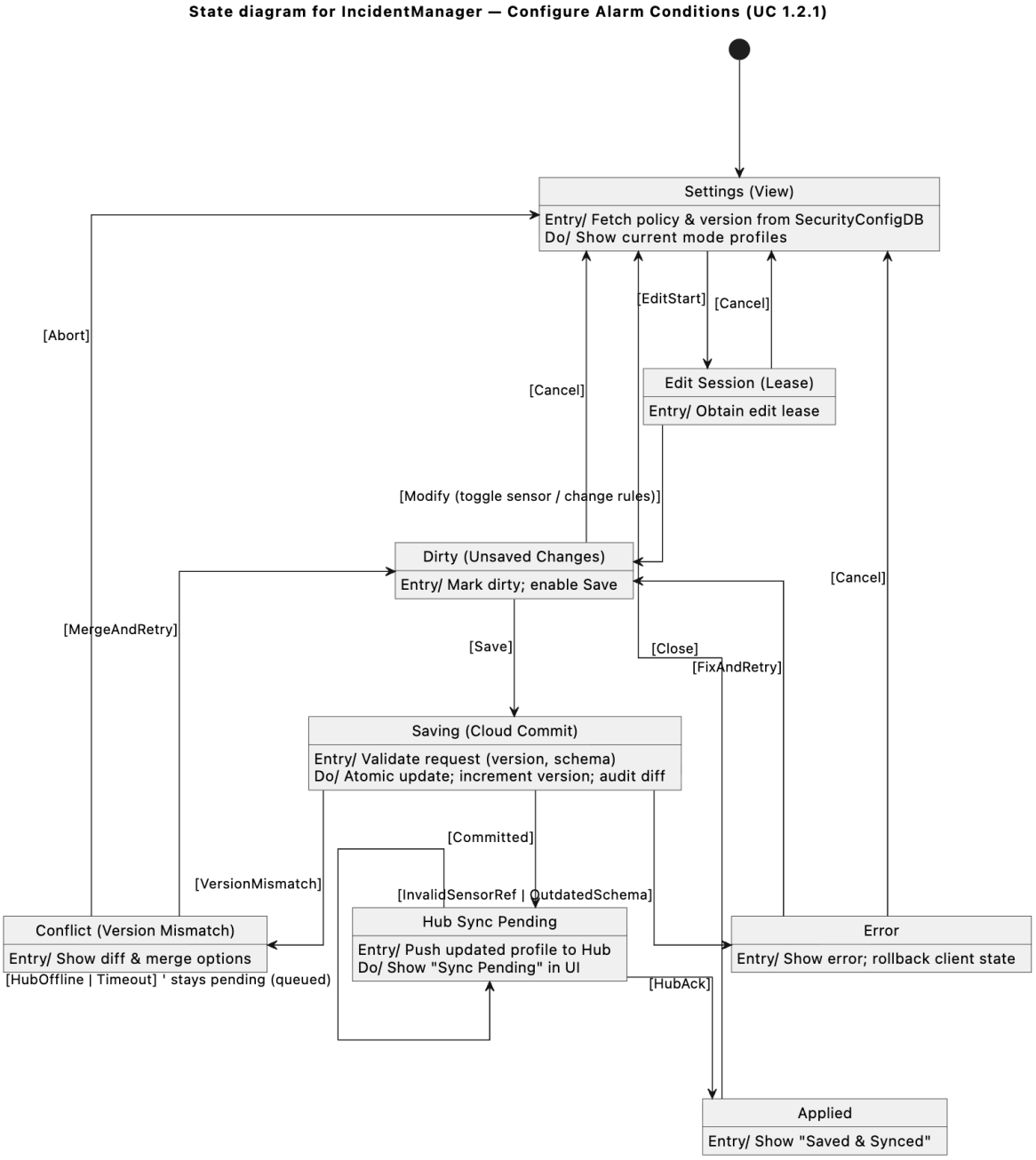
1.4 State diagram for SensorMonitoringManager - Outdoor Motion (UC 1.1.3)



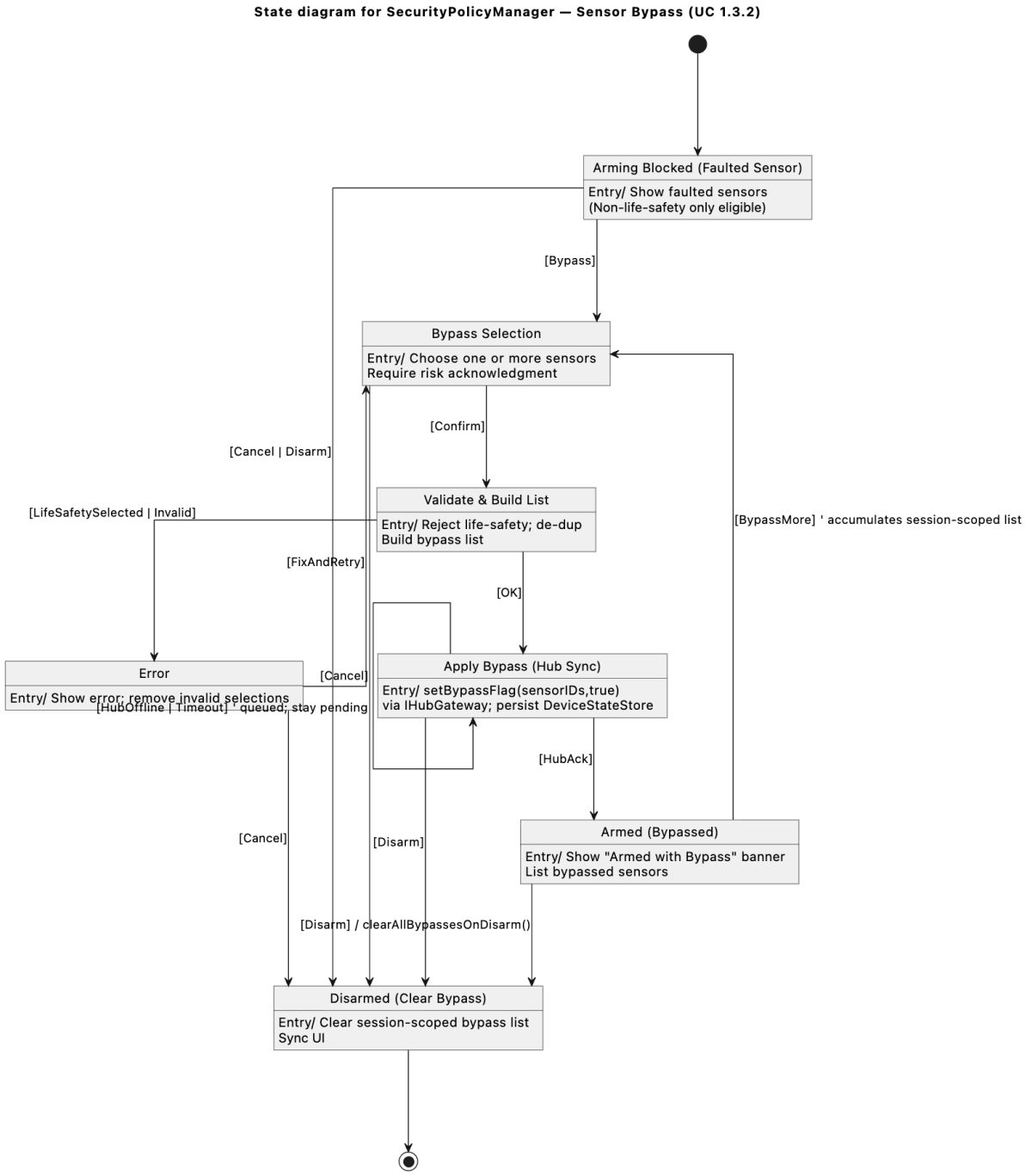
1.5 State diagram for SensorMonitoringManager - Dog Barking (UC 1.1.4)



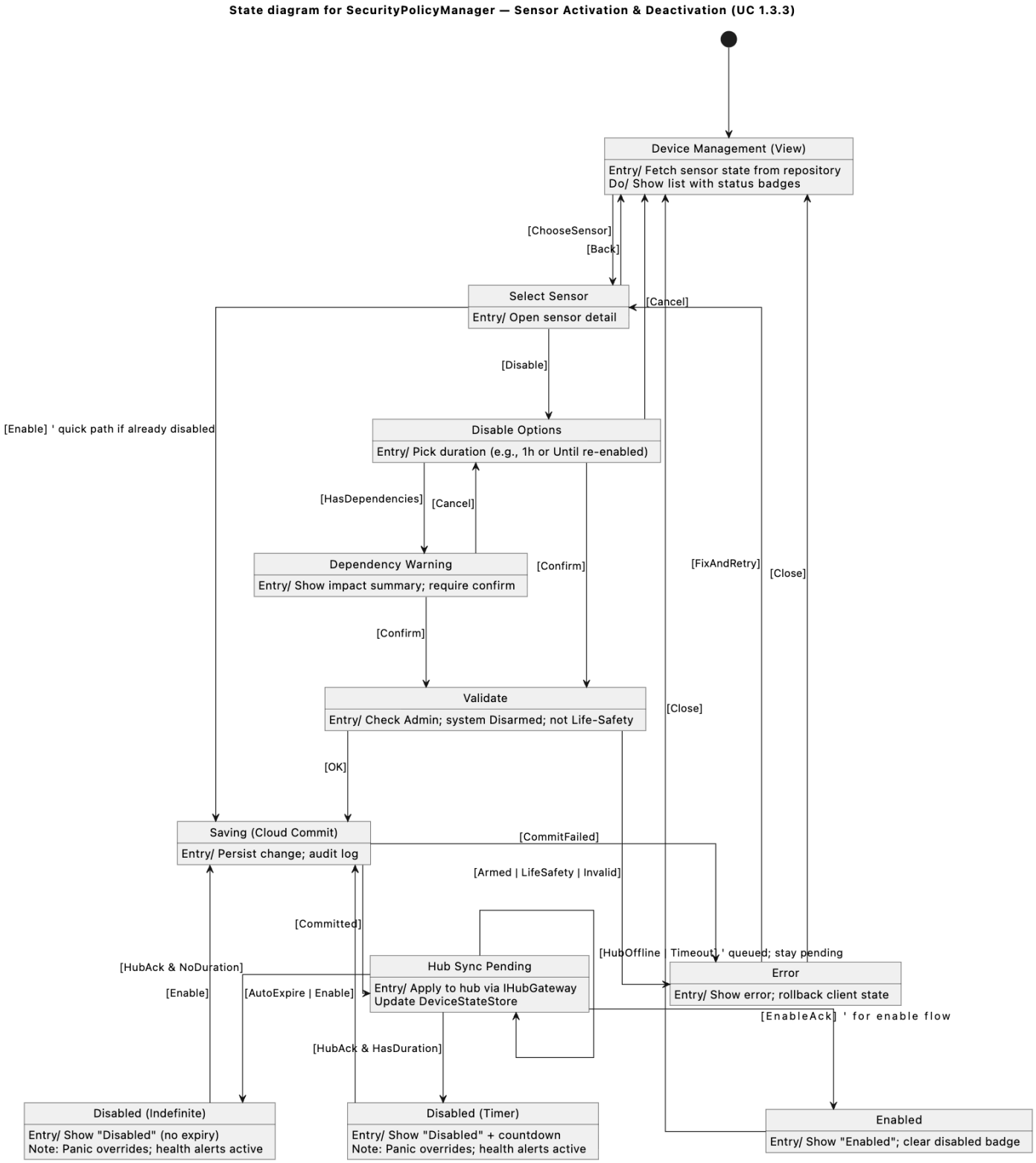
1.6 State diagram for IncidentManager - Configure Alarm Conditions (UC 1.2.1)



1.7 State diagram for SecurityPolicyManager - Sensor Bypass (UC 1.3.2)

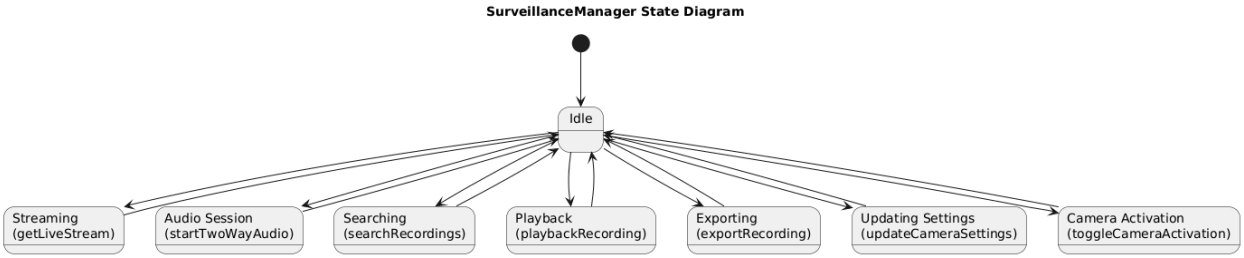


1.8 State diagram for SecurityPolicyManager - Sensor Activation & Deactivation (UC 1.3.3)

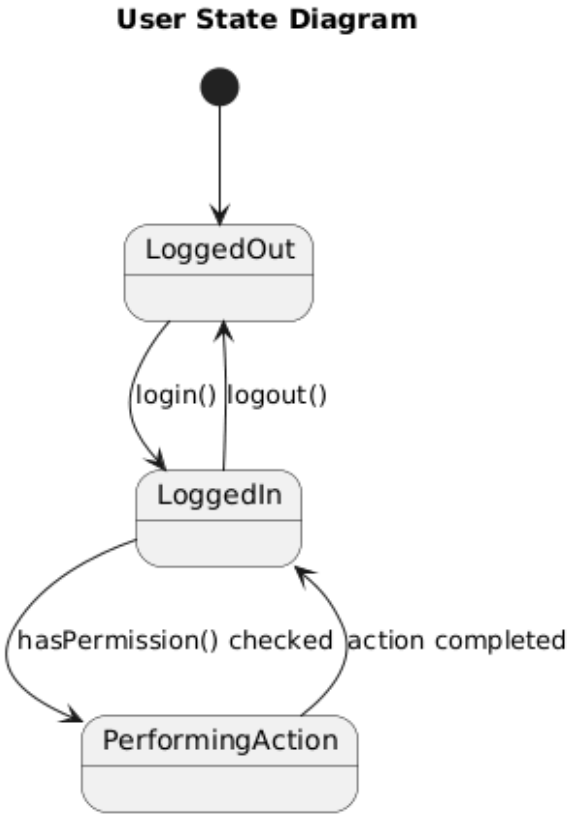


2. Live surveillance

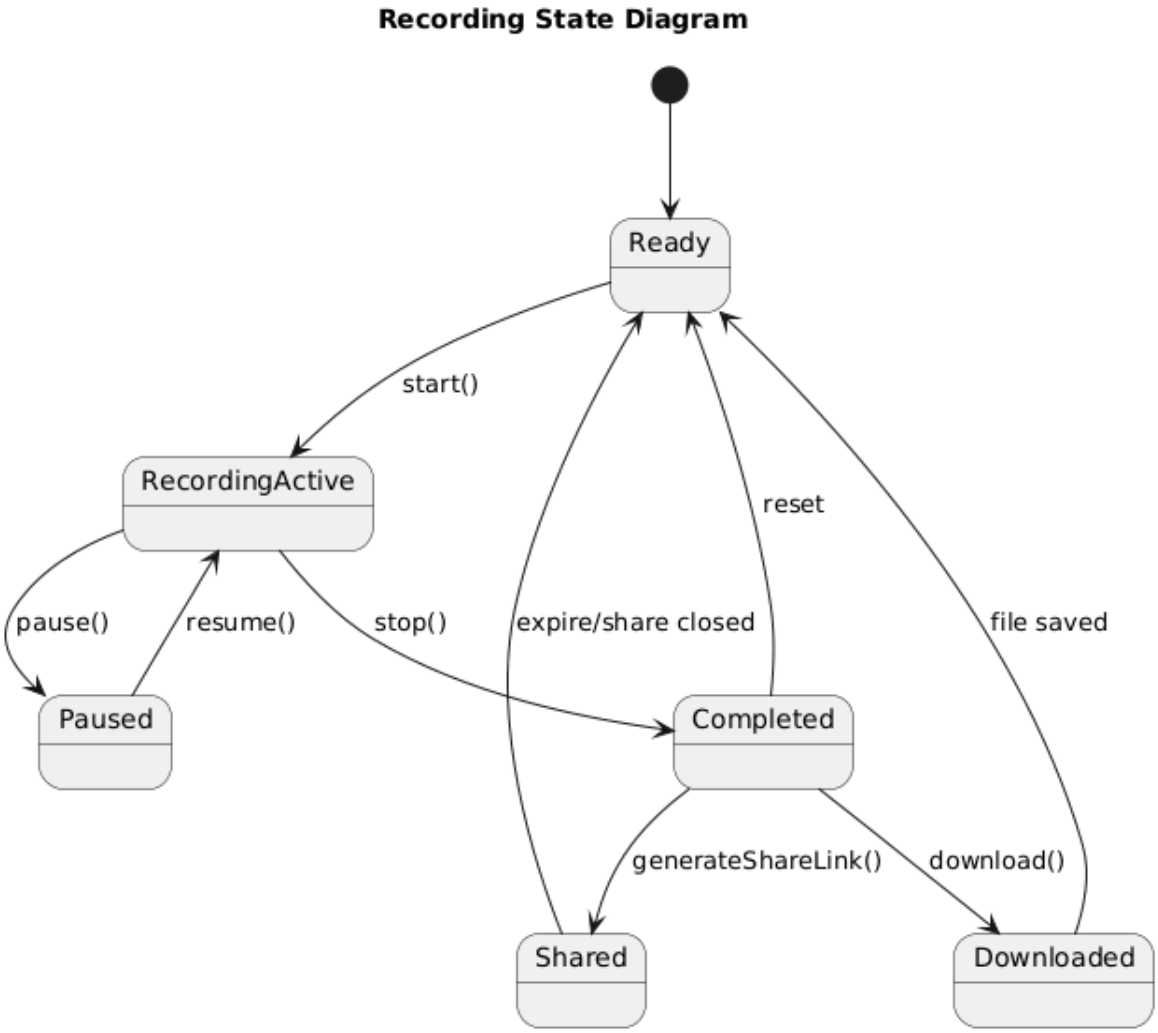
2.1 State diagram for SurveillanceManager



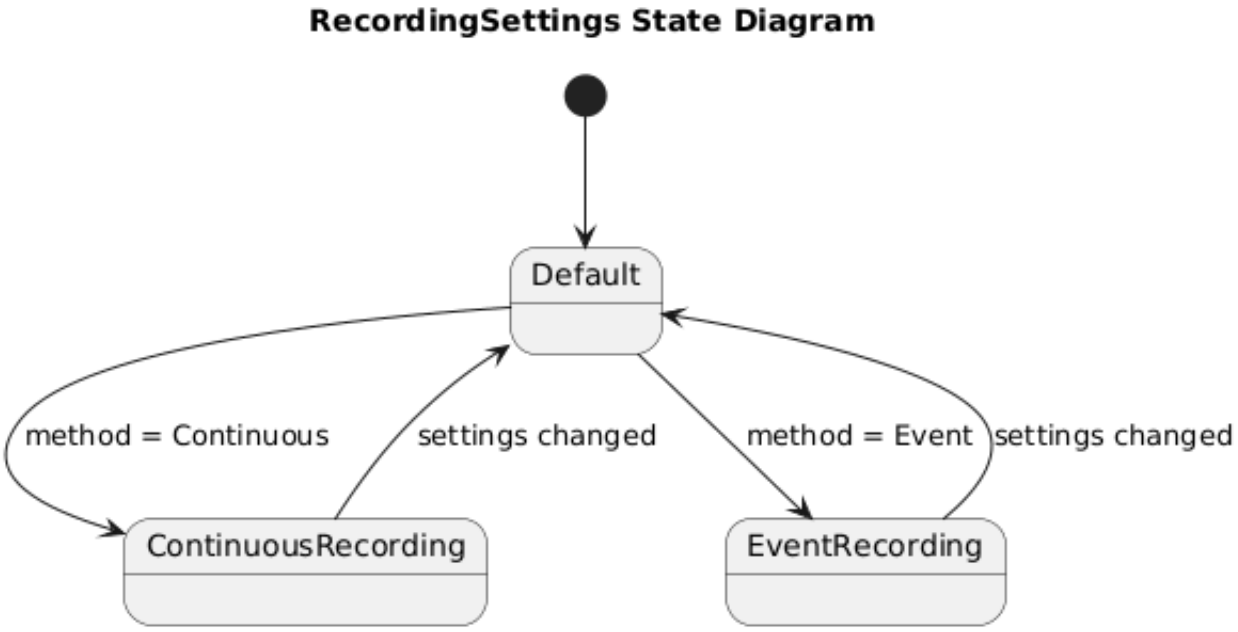
2.2 State diagram for User



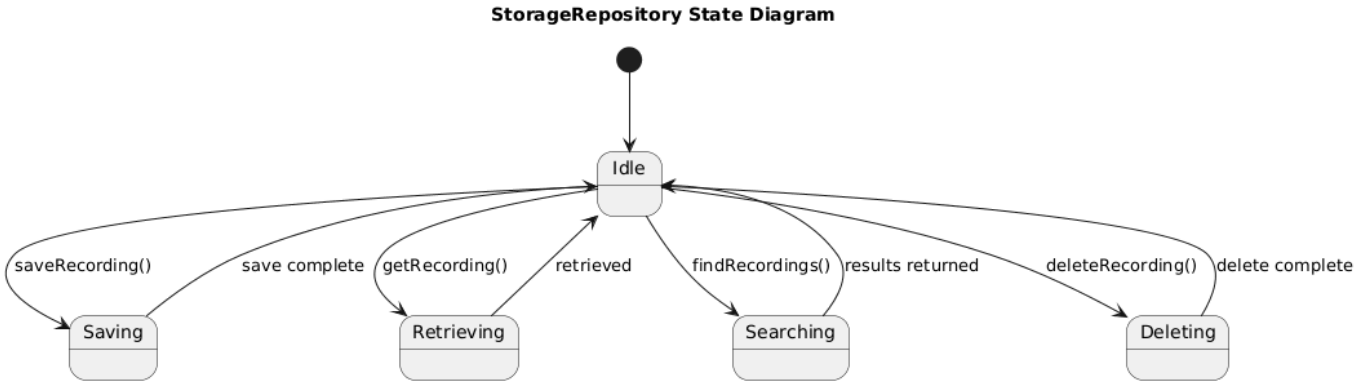
2.3 State diagram for Recording



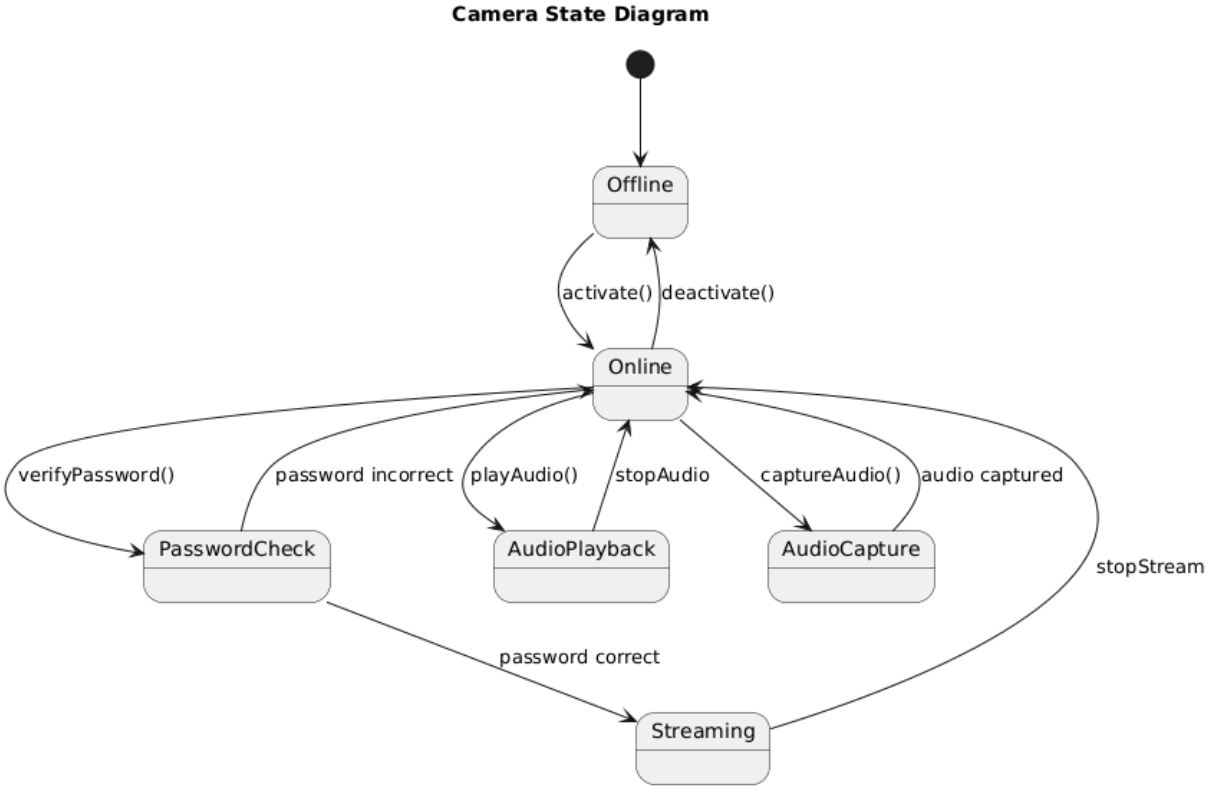
2.4 State diagram for RecordingSettings



2.5 State diagram for Storage Repository

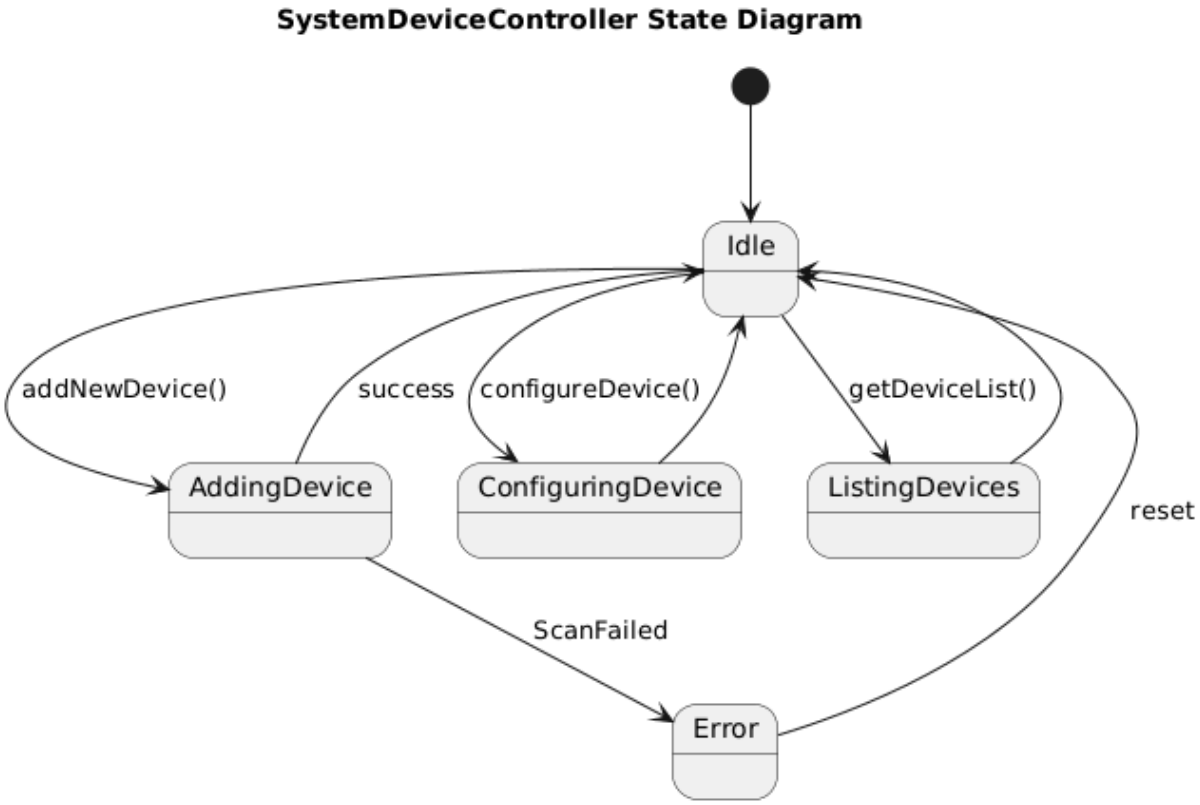


2.6 State diagram for Camera

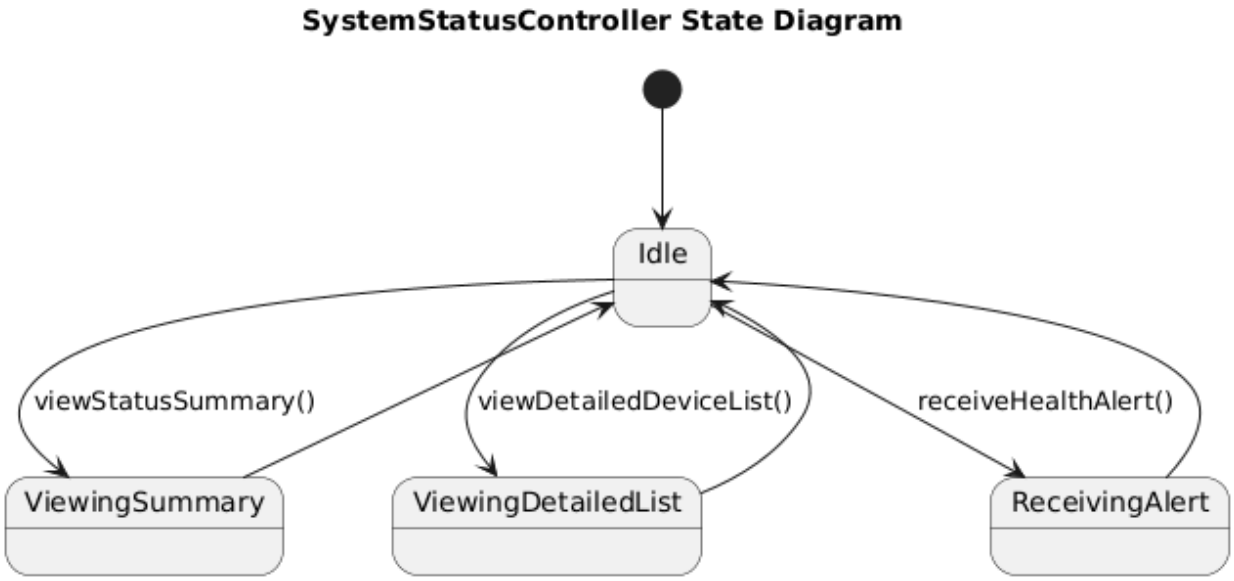


3. System and User Management

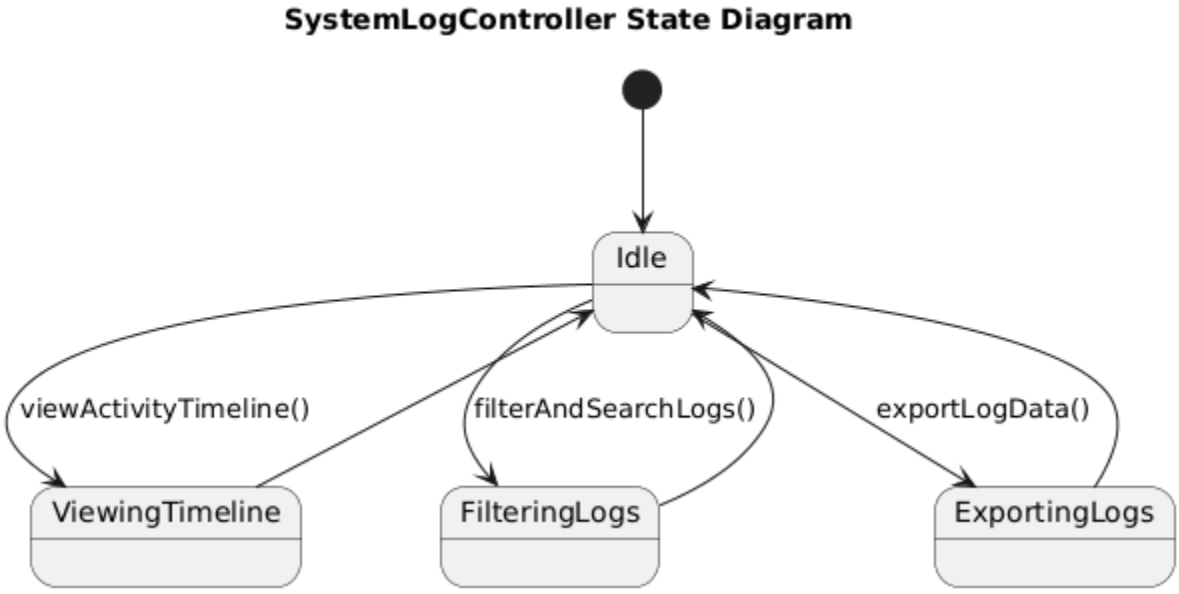
3.1 State diagram for SystemDeviceController



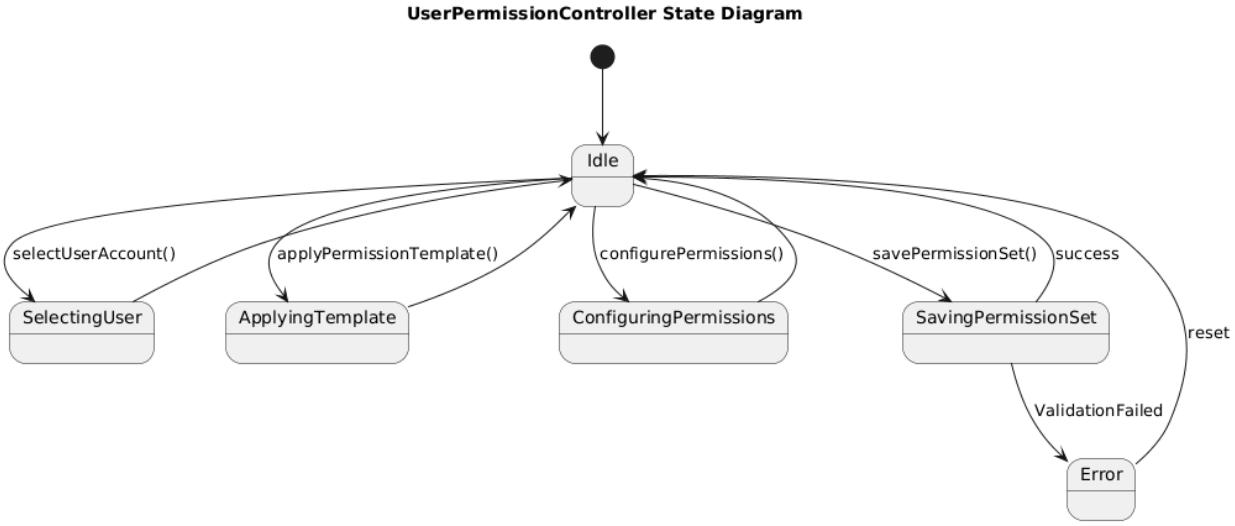
3.2 State diagram for SystemStatusController



3.3 State diagram for SystemLogController

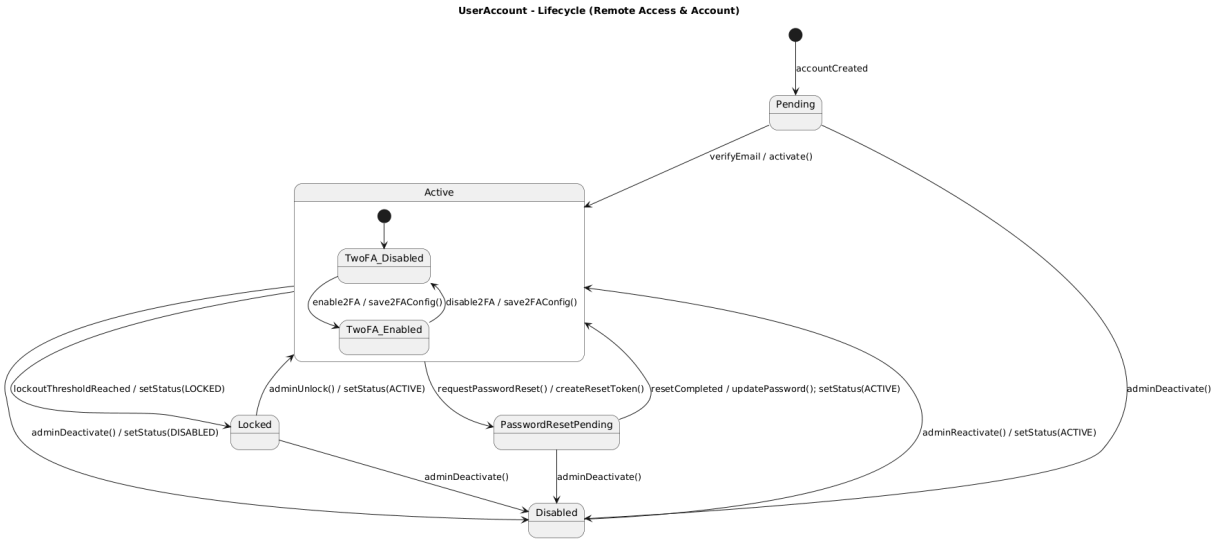


3.4 State diagram for UserPermissionController

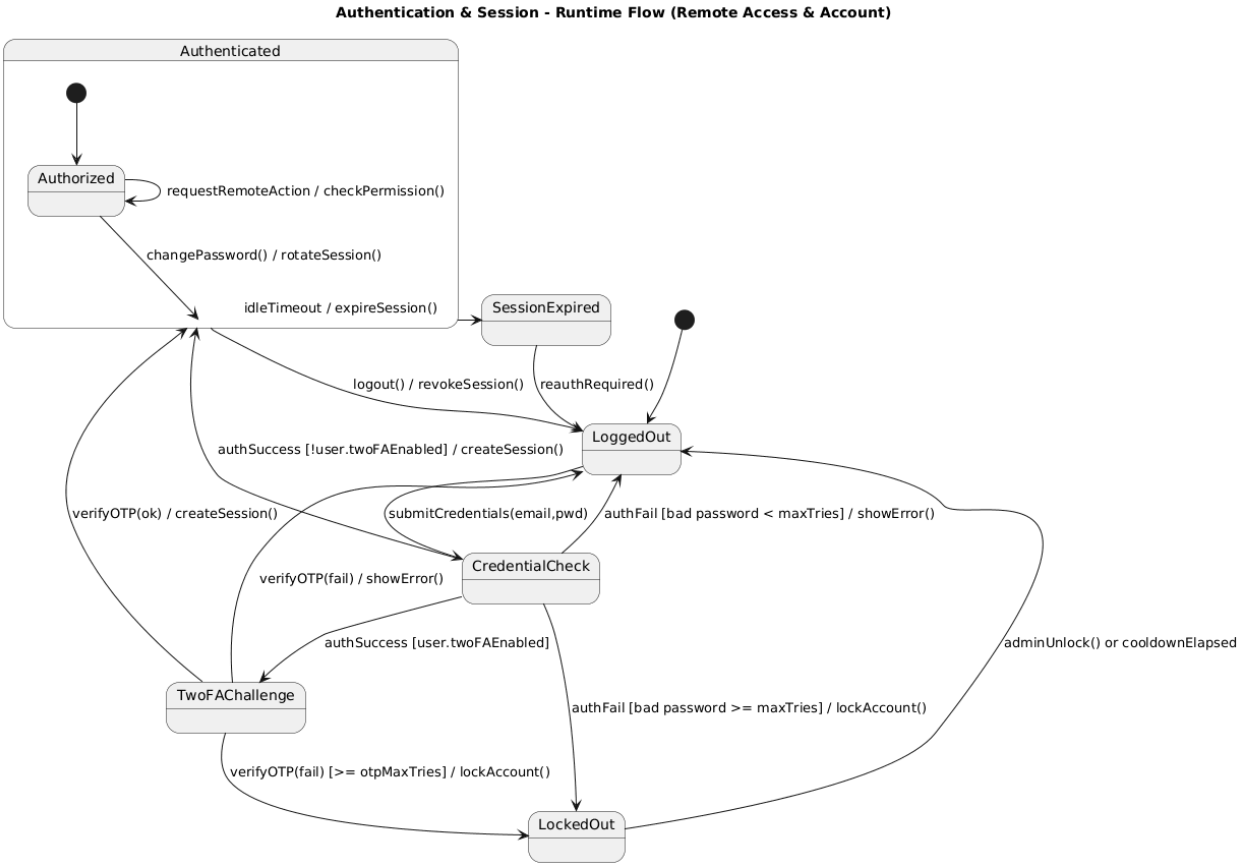


4. Remote Access and Account

4.1 State diagram for UserAccount

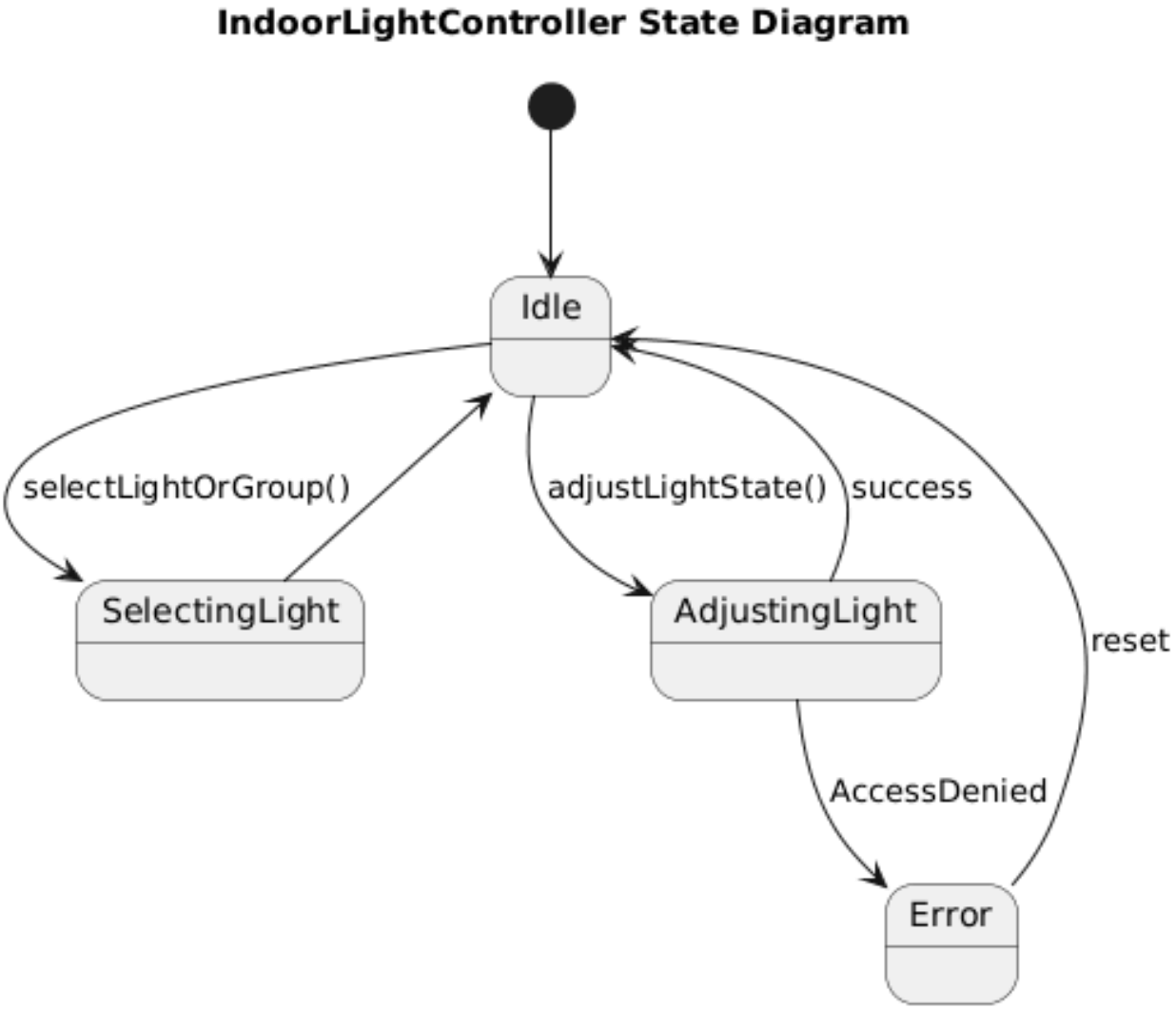


4.2 State diagram for Authentication & Session

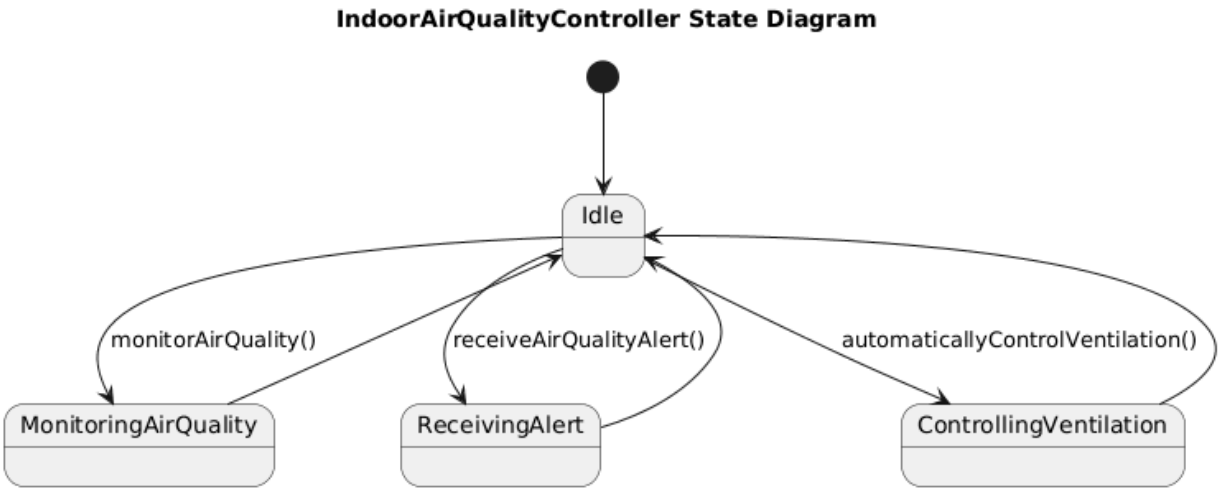


5. Indoor Monitoring and Sound Control

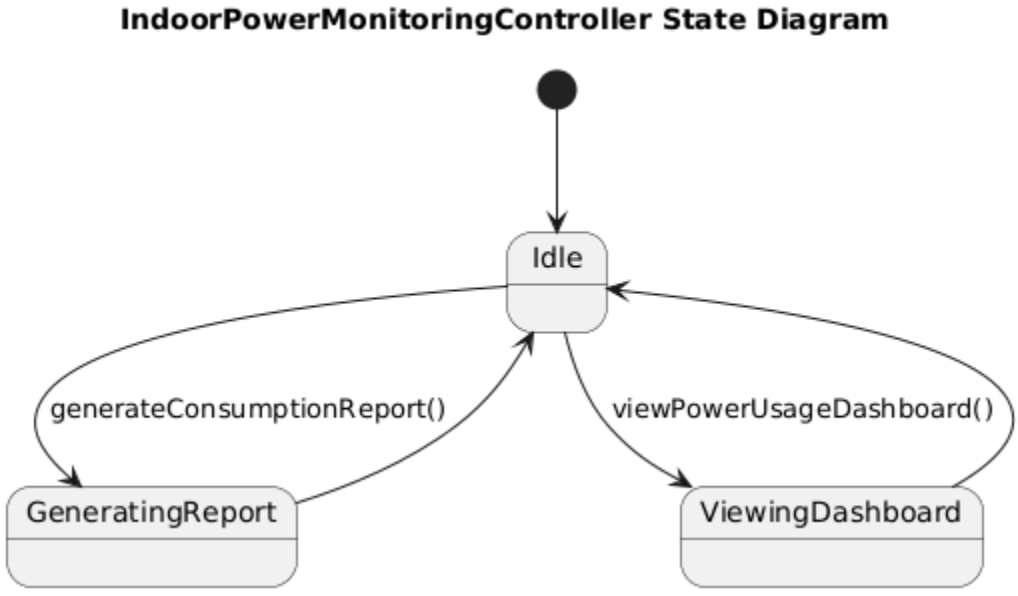
5.1 State diagram for IndoorLightController



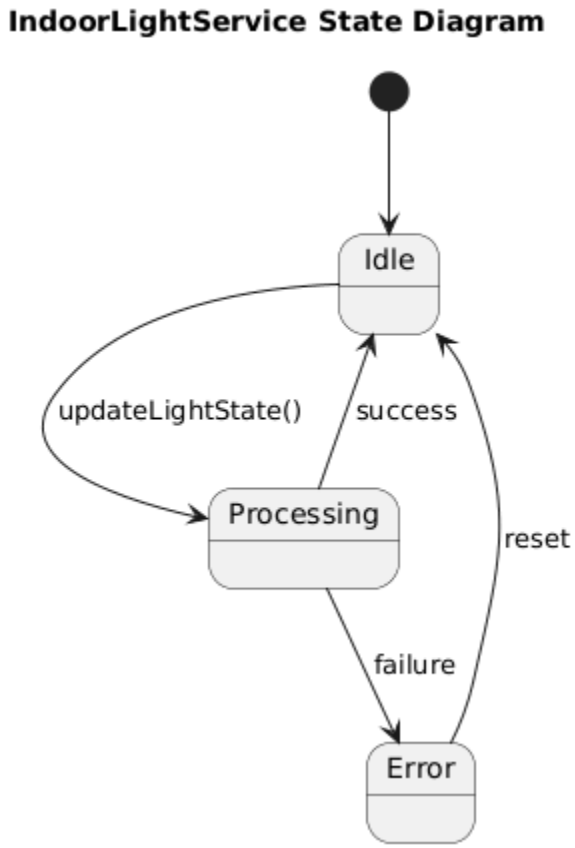
5.2 State diagram for IndoorAirQualityController



5.3 State diagram for IndoorPowerMonitoringController



5.4 State diagram for IndoorLightService



VI. Design Evaluation

1. Architectural Design Matrix

1.1 Intelligent Security Subsystem

1.1.a Fenton's simple morphology metrics

- Nodes(module) = 17
- Arc(connection) = 16
- Size = 33
- Depth = 5
- Width = 6
- Arc-to-node ratio = 16/17 ≈ 0.94

1.2 Live Surveillance Subsystem

1.2.a Fenton's simple morphology metrics

- Nodes(module) = 13
- Arc(connection) = 16
- Size = 29
- Depth = 5
- Width = 3
- Arc-to-node ratio = 16/13 ≈ 1.23

1.3 System & User Management Subsystem

1.3.a Fenton's simple morphology metrics

- Nodes(module) = 22
- Arc(connection) = 19
- Size = 41
- Depth = 5
- Width = 6
- Arc-to-node ratio = 19/22 ≈ 0.86

1.4 Remote Access & Account Subsystem

1.4.a Fenton's simple morphology metrics

- Nodes(module) = 20
- Arc(connection) = 28
- Size = 48
- Depth = 4
- Width = 6
- Arc-to-node ratio = 28/20 ≈ 1.4

1.5 Indoor Monitoring & Device Control Subsystem

1.5.a Fenton's simple morphology metrics

- Nodes(module) = 19
- Arc(connection) = 15
- Size = 34
- Depth = 5
- Width = 6
- Arc-to-node ratio = 15/19 ≈ 0.79

1.6 Overall

1.6.a Fenton's simple morphology metrics

- Nodes(module) = 112
- Arc(connection) = 144
- Size = 256
- Depth = 9
- Width = 38
- Arc-to-node ratio = 144/112 ≈ 1.286

1.7 Conclusion of Architectural Design Matrix

- **Intelligent Security:** With an arc-to-node ratio ≈ **0.94** and **Depth 5 / Width 6**, the structure is close to tree-like and moderately layered—coupling is controlled and breadth is balanced; maintainability should be solid with attention to the orchestration hubs.

- **Live Surveillance:** A higher density (**1.23**) on a small graph (13 nodes) and **narrow width (3)** suggest coupling concentrates around a few components (e.g., API/manager); complexity is acceptable but warrants careful responsibility boundaries.

- **System & User Management:** The ratio **0.86** with **Depth 5 / Width 6** indicates relatively modular, evenly layered design; coupling is low-to-moderate and should be easy to evolve as long as controllers remain thin.

- **Remote Access & Account:** The densest subsystem (**1.40**) with **Depth 4 / Width 6** shows tight interconnections around auth/permissions; change ripple risk is highest here, so interfaces and adapters are key to stability.

- **Indoor Monitoring & Device Control:** Lowest density (**0.79**) and **Depth 5 / Width 6** point to the most modular and testable slice; coupling is light and layering is clear.

- **Overall System:** At **1.286** with **Depth 9 / Width 38**, the system is moderately dense and very broad—cross-subsystem integration drives connectivity; keeping boundaries strict and minimizing unnecessary cross-links will preserve maintainability at scale.

2. CK Metrics

*Remarks: **DIT** (Depth of Inheritance Tree), **NoC** (Number of Children), **CBO** (Coupling Between Object classes), **LCOM** (Lack of Cohesion in Methods), **WMC** (Weighted Methods per Class), **RFC** (Response For Class)

2.1 Intelligent Security - CK Metrics

Class	DIT	NoC	CBO	LCOM	WMC	RFC
SensorMonitoringManager	0	0	9	0	4	9
IncidentManager	0	0	9	7	8	14
SecurityPolicyManager	0	0	7	0	6	15

User	0	0	0	0	1	1
Device (abstract)	0	2	0	0	0	0
Sensor (abstract)	1	8	2	0	3	4
ContactSensor	2	0	2	0	1	2
ShockSensor	2	0	2	0	1	2
MotionSensor	2	0	2	0	1	2
FireSensor	2	0	2	0	2	3
COSensor	2	0	2	0	1	2
GasSensor	2	0	2	0	1	2
LeakSensor	2	0	2	0	1	2
SoundSensor	2	0	2	0	1	2
Camera	1	0	4	0	3	4

Actuator (abstract)	1	2	1	0	2	2
Siren	2	0	1	0	2	2
SmartGasValve	2	0	1	0	1	1
Incident	0	0	2	0	1	1
SecurityPolicy	0	0	2	0	3	3
AutomationRule	0	0	0	0	0	0
EventLog	0	0	2	0	3	3
NotificationService	0	0	5	0	7	9
DeviceControlService	0	0	4	0	2	5
AutomationService	0	0	0	0	1	1
IIncidentRepository	0	0	2	0	3	4
ICooldownRepository	0	0	2	0	3	4

ISecurityPolicyRepository	0	0	2	0	4	4
IPolicyRepository	0	0	1	0	2	2
IHubGateway	0	0	2	0	4	4
IDispatchGateway	0	2	0	0	3	3
Push/SMS Provider (external)	0	0	0	0	6	6
IVR / Call Gateway (external)	0	0	0	0	5	5
External Security Service (external)	0	0	0	0	4	4

Summaries - Intelligent Security(n = 34)

Metric	Min (class)	Max (class)	Avg
DIT	0 (SensorMonitoringManager)	2 (ContactSensor)	0.68
NoC	0 (SensorMonitoringManager)	8 (Sensor (abstract))	0.41
CBO	0 (User)	9 (SensorMonitoringManager)	2.18
LCOM	0 (SensorMonitoringManager)	7 (IncidentManager)	0.21

2.2 Live Surveillance- CK Metrics

Class	DIT	NoC	CBO	LCOM	WMC	RFC
SurveillanceManager	0	0	6	0	7	8
User	0	0	0	0	1	1
Device (abstract)	0	1	0	0	0	0
Camera	1	0	3	0	6	6
RecordingSettings	0	0	0	0	0	0
Recording	0	0	0	0	3	3
StorageRepository	0	0	1	0	4	4

Summaries - Live Surveillance(n = 7)

Metric	Min (class)	Max (class)	Avg
DIT	0 (SurveillanceManager)	1 (Camera)	0.14
NoC	0 (SurveillanceManager)	1 (Device (abstract))	0.14

CBO	0 (User)	6 (SurveillanceManager)	1.43
LCOM	0 (SurveillanceManager)	0 (SurveillanceManager)	0.00

2.3 System & User Management- CK Metrics

Class	DIT	NoC	CBO	LCOM	WMC	RFC
SystemDeviceController	0	0	3	3	3	5
SystemStatusController	0	0	4	3	3	4
SystemLogController	0	0	5	3	3	5
UserPermissionController	0	0	4	6	4	4
SystemDeviceService	0	0	2	2	3	4
SystemHealthService	0	0	4	0	2	3
SystemLogService	0	0	3	0	2	3
UserPermissionService	0	0	3	0	1	2

SystemDeviceRepository	0	0	1	0	2	2
UserAccountRepository	0	0	1	0	2	2
SystemLogRepository	0	0	2	0	2	2

Summaries - System & User Management(n = 11)

Metric	Min (class)	Max (class)	Avg
DIT	0 (SystemDeviceController)	0 (SystemDeviceController)	0.00
NoC	0 (SystemDeviceController)	0 (SystemDeviceController)	0.00
CBO	1 (SystemDeviceRepository)	5 (SystemLogController)	2.91
LCOM	0 (SystemHealthService)	6 (UserPermissionController)	1.55

2.4 Remote Access & Account- CK Metrics

Class	DIT	NoC	CBO	LCOM	WMC	RFC
UserAccount	0	0	5	0	0	0
Credential	0	0	1	0	2	2
Session	0	0	1	0	2	3
ResetToken	0	0	1	0	2	3
TwoFactorConfig	0	0	1	0	2	3
Role	0	0	0	0	0	0
Permission	0	0	0	0	0	0
RoleAssignment	0	0	0	0	0	0
RolePermission	0	0	0	0	0	0
LoginInterface	0	0	0	0	0	0
LoginManager	0	0	5	0	3	8

AccessControl	0	0	5	0	1	2
StorageManager	0	0	0	0	4	4

Summaries - Remote Access & Account(n = 13)

Metric	Min (class)	Max (class)	Avg
DIT	0 (UserAccount)	0 (UserAccount)	0.00
NoC	0 (UserAccount)	0 (UserAccount)	0.00
CBO	0 (Role)	5 (UserAccount)	1.46
LCOM	0 (UserAccount)	0 (UserAccount)	0.00

2.5 Indoor Monitoring & Device Control- CK Metrics

Class	DIT	NoC	CBO	LCOM	WMC	RFC
IndoorLightController	0	0	2	2	2	3
IndoorAirQualityController	0	0	3	2	3	4

IndoorPowerMonitoringController	0	0	3	1	2	3
IndoorLightService	0	0	2	0	1	3
IndoorAirQualityService	0	0	2	0	2	3
IndoorPowerService	0	0	3	0	2	3
IndoorLightRepository	0	0	0	0	2	2
AirSensorRepository	0	0	0	0	2	2
PowerMeterRepository	0	0	0	0	2	2

Summaries - Indoor Monitoring & Device Control(n = 9)

Metric	Min (class)	Max (class)	Avg
DIT	0 (IndoorLightController)	0 (IndoorLightController)	0.00
NoC	0 (IndoorLightController)	0 (IndoorLightController)	0.00
CBO	0 (IndoorLightRepository)	3 (IndoorAirQualityController)	1.67
LCOM	0 (IndoorLightService)	2 (IndoorLightController)	0.56

2.6 CK Metrics Conclusion

- **Intelligent Security:** Inheritance is shallow (max DIT ≈ 2) with a broad sensor family (high NoC), which supports extension without deep hierarchies. The orchestration hubs (SensorMonitoringManager, IncidentManager) have the highest coupling and RFC, making them change and test hotspots. IncidentManager's elevated LCOM signals mixed responsibilities; splitting verification, dispatch, and state-transition logic would improve cohesion. Services and repositories are cohesive with moderate coupling, and external integrations are sensibly isolated behind interfaces—keep those contracts stable and consider façade/event-driven patterns to reduce direct coupling.
- **Live Surveillance:** Inheritance is minimal (DIT 0–1) and overall coupling is moderate. SurveillanceManager concentrates multiple concerns (live view, two-way audio, search/playback, export), which drives up CBO and RFC; factoring these into focused collaborators would reduce complexity. Camera and StorageRepository remain cohesive and testable with clear boundaries.
- **System & User Management:** There is no inheritance (DIT 0), and layering is straightforward. Controllers show higher CBO and LCOM because they are stateless coordinators—acceptable so long as business rules live in services. Services and repositories are cohesive and lightly coupled, which supports maintainability; keep controller–service boundaries thin and consistent.
- **Remote Access & Account:** The domain is largely data-centric (many classes with WMC ≈ 0) and uses no inheritance. Complexity is concentrated in LoginManager and AccessControl (higher CBO/RFC) as they coordinate credentials, sessions, and role/permission checks. Heavy reliance

on StorageManager suggests strengthening port/adapter boundaries and using test doubles; cohesion is generally good, and security/policy checks could be further modularized.

- **Indoor Monitoring & Device Control:** The structure is flat (DIT 0) and simple. Controllers have modest coupling and some LCOM due to stateless orchestration; keeping logic in services preserves cohesion. Services and repositories are cohesive and lightly coupled, making the subsystem easy to test and extend; if automation evolves, consider extracting explicit policy/threshold objects

3. MOOD metric

a. AHF, AIF, MIF

Class	Ad	Ah	Ai	Aa	Md	Mi	Ma
Device <<Abstract>> (D1)	4	4	0	4	2	0	2
Sensor <<Abstract>> (D1)	2	2	4	6	2	2	4
Actuator <<Abstract>> (D1)	0	0	4	4	2	2	4
Incident (D1)	5	5	0	5	0	0	0
SecurityPolicy (D1)	3	3	0	3	0	0	0
EventLog (D1)	5	5	0	5	0	0	0
User (D2a)	2	2	0	2	1	0	1

SensorMonitoringManager	4	4	0	4	4	0	4
IHubGateway (D2a)	0	0	0	0	1	0	1
IIncidentRepository (D2a)	0	0	0	0	2	0	2
ICooldownRepository (D2a)	0	0	0	0	2	0	2
NotificationService (D2a)	0	0	0	0	2	0	2
DeviceControlService (D2a)	0	0	0	0	2	0	2
Sensor <<Abstract>> (D2b)	5	3	0	5	1	0	1
ContactSensor	2	2	5	7	2	1	3
ShockSensor	3	3	5	8	2	1	3
MotionSensor	4	4	5	9	2	1	3
FireSensor	3	3	5	8	2	1	3
COSensor	2	2	5	7	1	1	2

GasSensor	2	2	5	7	1	1	2
LeakSensor	2	2	5	7	2	1	3
SoundSensor	3	3	5	8	2	1	3
Camera (D2b)	1	0	0	1	4	0	4
SensorEventData	5	5	0	5	0	0	0
MotionEventData	5	5	0	5	0	0	0
SoundEventData	5	5	0	5	0	0	0
IHubGateway (D2b)	0	0	0	0	1	0	1
User (D2c)	2	2	0	2	1	0	1
Incident (D2c)	5	5	0	5	1	0	1
Actuator <<Abstract>> (D2c)	2	2	0	2	2	0	2
Siren (D2c)	4	4	2	6	2	2	4

SmartGasValve	3	3	2	5	1	2	3
DeviceControlService (D2c)	3	3	0	3	2	0	2
NotificationService (D2c)	2	2	0	2	2	0	2
IIncidentRepository (D2c)	0	0	0	0	3	0	3
ICooldownRepository (D2c)	0	0	0	0	3	0	3
CooldownEntry	2	2	0	2	0	0	0
EventLog (D2c)	7	7	0	7	3	0	3
Push/SMS Provider (D2c)	1	1	0	1	3	0	3
CommandResult	4	4	0	4	0	0	0
NotificationResult	3	3	0	3	0	0	0
User (D3)	2	2	0	2	1	0	1
IncidentManager	5	5	0	5	8	0	8

IIncidentRepository (D3)	0	0	0	0	3	0	3
IPolicyRepository	0	0	0	0	2	0	2
IDispatchGateway	0	0	0	0	3	0	3
NotificationService (D3)	0	0	0	0	3	0	3
DeviceControlService (D3)	0	0	0	0	1	0	1
Incident (D3)	4	4	0	4	1	0	1
SecurityPolicy (D3)	2	2	0	2	1	0	1
Camera (D3)	1	0	0	1	2	0	2
Siren (D3)	1	0	0	1	1	0	1
IVR / Call Gateway	5	5	0	5	5	0	5
External Security Service	5	5	0	5	4	0	4
Push/SMS Provider (D3)	5	5	0	5	4	0	4

User (D4)	2	2	0	2	1	0	1
SecurityPolicyManager	4	4	0	4	6	0	6
ISecurityPolicyRepository	0	0	0	0	4	0	4
IHubGateway (D4)	0	0	0	0	3	0	3
AutomationService	0	0	0	0	1	0	1
NotificationService (D4)	0	0	0	0	2	0	2
Sensor <<Abstract>> (D4)	5	5	0	5	2	0	2
SecurityPolicy (D4)	3	3	0	3	2	0	2
AutomationRule	3	3	0	3	0	0	0
Push/SMS Provider (D4)	6	6	0	6	5	0	5
SurveillanceManager	2	2	0	2	8	0	8
User (Surv)	2	2	0	2	1	0	1

Device <<Abstract>> (Surv)	4	4	0	4	0	0	0
Camera (Surv)	2	2	4	6	6	0	6
RecordingSettings	5	5	0	5	0	0	0
Recording	5	5	0	5	3	0	3
StorageRepository	0	0	0	0	4	0	4
SystemDeviceController	0	0	0	0	3	0	3
SystemStatusController	0	0	0	0	3	0	3
SystemLogController	0	0	0	0	3	0	3
UserPermissionController	0	0	0	0	4	0	4
SystemDeviceService	0	0	0	0	3	0	3
SystemHealthService	0	0	0	0	2	0	2
SystemLogService	0	0	0	0	2	0	2

UserPermissionService	0	0	0	0	1	0	1
SystemDeviceRepository	1	1	0	1	2	0	2
UserAccountRepository	1	1	0	1	2	0	2
SystemLogRepository	1	1	0	1	2	0	2
UserAccount	8	8	0	8	8	0	8
Credential	4	4	0	4	2	0	2
Session	6	6	0	6	2	0	2
ResetToken	5	5	0	5	2	0	2
TwoFactorConfig	5	5	0	5	2	0	2
Role	3	3	0	3	0	0	0
Permission	3	3	0	3	0	0	0
RoleAssignment	3	3	0	3	0	0	0

RolePermission	2	2	0	2	0	0	0
LoginInterface	8	8	0	8	7	0	7
LoginManager	0	0	0	0	3	0	3
AccessControl	0	0	0	0	1	0	1
StorageManager	7	7	0	7	4	0	4
IndoorLightController	0	0	0	0	2	0	2
IndoorAirQualityController	0	0	0	0	3	0	3
IndoorPowerMonitoringController	0	0	0	0	2	0	2
IndoorLightService	0	0	0	0	1	0	1
IndoorAirQualityService	0	0	0	0	2	0	2
IndoorPowerService	0	0	0	0	2	0	2
IndoorLightRepository	1	1	0	1	2	0	2

AirSensorRepository	1	1	0	1	2	0	2
PowerMeterRepository	1	1	0	1	2	0	2

AHF: 233 / 238 **0.9790** **97.90%**
This indicates that nearly 98% of all attributes are private or protected, adhering to the principle of encapsulation. Data is well-hidden within classes, which is a sign of a robust design.

AIF: 56 / 294 **0.1905** **19.05%**
This shows that only about 19% of the attributes within classes are inherited from parent classes. The majority of attributes are newly defined within the classes themselves.

MIF: 16 / 242 **0.0661** **6.61%**
This extremely low value means that less than 7% of the methods in classes are inherited directly (i.e., without being overridden).

b. Coupling Factor

1. Intelligent Security (UC 1.x)

Class Numbers: 1 User, **2** SensorMonitoringManager, **3** IncidentManager, **4** SecurityPolicyManager, **5** NotificationService, **6** DeviceControlService, **7** IHubGateway, **8** IIncidentRepository, **9** ICooldownRepository, **10** ISecurityPolicyRepository, **11** IDispatchGateway, **12** Incident, **13** SecurityPolicy, **14** EventLog, **15** Camera, **16** Siren, **17** SmartGasValve

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	1	1	0	1	1	0	0	1	0	1	0	0	0
3	1	0	0	0	1	1	0	1	0	1	1	1	1	0	1	1	0

4	1	0	0	0	1	0	1	0	0	1	0	0	1	0	0	0	0
5	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
7	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
11	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
13	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	0	0	0	0	0
15	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	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Calculation of CF:

- N = 17, C = 34 → CF = 34 / (17×16) = 0.125
- Low–moderate coupling. Managers talk to services/repos, but most other pairs remain independent.

2. Live Surveillance (UC 2.x)

Class Numbers: **1** SurveillanceManager, **2** User, **3** Device, **4** Camera, **5** RecordingSettings, **6** Recording, **7** StorageRepository

#	1	2	3	4	5	6	7
1	0	1	0	1	1	1	1
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	1	1	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0

Calculation of CF:

- N = 7, C = 8 → CF = 8 / (7×6) ≈ 0.190
- Highest of the subsystems. A central SurveillanceManager coordinates cameras, storage, and users, creating more links.

3. System & User Management (UC 3.x)

Class Numbers: **1** SystemDeviceController, **2** SystemStatusController, **3** SystemLogController, **4** UserPermissionController, **5** SystemDeviceService, **6** SystemHealthService, **7** SystemLogService, **8** UserPermissionService, **9** SystemDeviceRepository, **10** UserAccountRepository, **11** SystemLogRepository

#	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	1	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	1	0	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	0	0

10	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0

Calculation of CF:

- N = 11, C = 8 → CF = 8 / (11×10) ≈ 0.073
- Low coupling. Controllers delegate to services, which then hit repositories—clean layers, few cross-links.

4. Remote Access & Account (UC 4.x)

Class Numbers: **1** UserAccount, **2** Credential, **3** Session, **4** ResetToken, **5** TwoFactorConfig, **6** Role, **7** Permission, **8** RoleAssignment, **9** RolePermission, **10** LoginInterface, **11** LoginManager, **12** AccessControl, **13** StorageManager

#	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	1	1	1	0	0	1	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0	1
6	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
8	0	0	0	0	0	1	0	0	0	0	0	0	1
9	0	0	0	0	0	1	1	0	0	0	0	0	1
10	0	0	0	0	0	0	0	0	0	0	0	0	0
11	1	1	1	0	0	0	0	0	0	1	0	0	1
12	1	0	0	0	0	0	1	1	1	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0

Calculation of CF:

- N = 13, C = 25 → CF = 25 / (13×12) ≈ 0.160
- Medium coupling. LoginManager/AccessControl and shared StorageManager produce more dependencies than a pure layered split.

5. Indoor Monitoring & Device Control (UC 5.x)
Class Numbers:**1** IndoorLightController, **2** IndoorAirQualityController, **3** IndoorPowerMonitoringController, **4** IndoorLightService, **5** IndoorAirQualityService, **6** IndoorPowerService, **7** IndoorLightRepository, **8** AirSensorRepository, **9** PowerMeterRepository

#	1	2	3	4	5	6	7	8	9
1	0	0	0	1	0	0	0	0	0
2	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0

Calculation of CF:

- N = 9, C = 6 → CF = 6 / (9×8) = 0.083. Low coupling. Each controller→service→repository chain is mostly vertical with minimal cross-talk.

6. Coupling Factor Conclusion

Overall, the system exhibits **low coupling** by CF, with a rough subsystem average of **≈0.13** (Intelligent Security **0.125**, Live Surveillance **0.190**, System & User Management **0.073**, Remote Access & Account **0.160**, Indoor Monitoring & Device Control **0.083**). The **lowest CF** appears in System & User Management and Indoor Monitoring, reflecting clean vertical controller→service→repository chains and minimal cross-talk. **Intelligent Security** sits near the overall average, suggesting balanced coordination between managers, services, and repositories. The **highest CF** concentrations are in **Live Surveillance (0.190)**—driven by a hub-like SurveillanceManager touching cameras, recordings, and storage—and in **Remote Access & Account (0.160)** where LoginManager/AccessControl and a shared StorageManager introduce extra dependencies. If further decoupling is desired, the primary opportunities are to reduce central manager fan-out (e.g., facade/ports, events) and to narrow shared data-access points; otherwise the current coupling profile is consistent with a layered architecture and should scale with manageable risk.

VII. WHO DID WHAT

Jamal Alibalayev
1. System and User Management a. Architectural structure b. CRC cards c. State diagram 2. Indoor Monitoring and Sound Control a. Architectural structure b. CRC cards c. State diagram 3. Consistency of classes checking
Yonas Alexander Grossard Amin
1. Remote Access and Account a. Architectural structure b. CRC cards c. State diagram 2. Checking style of the diagrams 3. Glossary
Alan Pak To Cheung
1. Live surveillance a. Architectural structure b. CRC cards c. State diagram 2. Overall

3. Meeting log
4. Who did what

Jongyoon Baek

1. Intelligent Security
    a. Architectural structure
    b. CRC cards
    c. State diagram
2. Editing assumption
3. Design Evaluation

VIII. Meeting Log

Meeting logs should clearly describe 5W1H (who will do what by when with why, where and how)

Meeting 1

Date	2025/11/09, 5pm - 6pm
Location	Student lounge, Second floor, MirHall dormitory
Attendees	Jamal Alibalayev Yonas Alexander Grossard Amin Alan Pak To Cheung Jongyoon Baek
Summary	**Meeting Objective**: Divide the responsibility for Overall, and each use case's Architectural structure **Reason**: To ensure parallel progress **Key Discussions:** ● The team reviewed all primary use cases from the Software Requirements Specification (SRS) document. ● After discussion, the team reached a consensus to divide responsibilities based on functional modules. **Result**: The job was divided in the following way: ● Jamal Alibalayev - System and User Management & Indoor Monitoring and Sound Control

	● Yonas Alexander Grossard Amin - Remote Access and Account ● Alan Pak To Cheung - Overall, Live surveillance ● Jongyoon Baek - Intelligent Security Everybody will do their tasks in a separate way until the next meeting, if any issue arises, the person should notify other team members about it. **TO-DO(before next meeting):** ● Finish the Overall and Architectural structure ● Check the potential inconsistency Next meeting: 2025/11/11, 7pm - 8pm

Meeting 2

Date	2025/11/11, 7pm - 8pm
Location	Student lounge, Second floor, MirHall dormitory
Attendees	Jamal Alibalayev Yonas Alexander Grossard Amin Alan Pak To Cheung Jongyoon Baek

Summary	**Meeting Objective**: Reviewed previous work (overall progress & architectural structure); discussed approaches for creating the class diagram, CRC cards, and state diagrams. **Reason**: To ensure consistent design and maintain alignment across all diagrams. **Key Discussions:** ● All members presented their initial architectural diagrams. The team confirmed that all core use cases were covered. **Result**: The team agreed to unify the diagram style and to complete the class diagram, CRC cards, and state diagrams together to ensure coherence and consistency. If any issue arises, the person should notify other team members about it. **TO-DO(before next meeting):** ● Standardize the diagram style across all design documents(All team members). ● Complete the class diagram, CRC cards, and state diagrams. ● Jamal Alibalayev - System and User Management & Indoor Monitoring and Sound Control ● Yonas Alexander Grossard Amin - Remote Access and Account ● Alan Pak To Cheung - Overall, Live surveillance ● Jongyoon Baek - Intelligent Security Next meeting: 2025/11/12, 7pm - 8pm

Meeting 3

Date	2025/11/12, 7pm - 8pm
Location	Student lounge, Second floor, MirHall dormitory

Attendees	Jamal Alibalayev Yonas Alexander Grossard Amin Alan Pak To Cheung Jongyoon Baek
Summary	**Meeting Objective**: Reviewed the progress on class diagram, CRC cards, and state diagrams. Discussed the consistency of the diagram style and integration between different design components. If any issue arises, the person should notify other team members about it. **Reason**: To check the completion status of diagrams and ensure alignment across the system design. **Key Discussions:** ● The team conducted a cross-review of all submitted PlantUML diagrams. The diagram style was confirmed to be mostly unified. **Result**: The team reviewed the submitted diagrams and confirmed that the style has been mostly unified. Minor adjustments are still needed to improve readability and consistency between CRC cards and class diagrams. If any issue arises, the person should notify other team members about it. **TO-DO(before next meeting):** ● Yonas Alexander Grossard Amin - Finalize updates to the class diagram based on feedback. ● Alan Pak To Cheung - Revise CRC cards to ensure consistency with class responsibilities. ● Jamal Alibalayev - Refine the state diagrams for clarity and logical flow. ● Jongyoon Baek - Finish the design evaluation Next meeting: 2025/11/14, 10am - 11am(final meeting)

Meeting 4(final meeting)

Date	2025/11/14, 10am - 11am
Location	Student lounge, Second floor, MirHall dormitory
Attendees	Jamal Alibalayev Yonas Alexander Grossard Amin Alan Pak To Cheung Jongyoon Baek
Summary	**Meeting Objective**: To complete the design evaluation and perform a full consistency check across the entire SDS. **Reason**: This is the final stage of the project, and we need to ensure that all design components align with each other, without conflicts, and that the overall system architecture and design comply with the SRS and project goals. If any issue arises, the person should notify other team members about it. **Key Discussions:** ● Reviewed each section of the SDS to verify consistency and alignment. ● Checked class diagrams, state diagrams, CRC cards, and architectural diagrams for contradictions or missing elements. ● Identified remaining items that need to be updated or refined for the final version of the SDS**.** **Result**: Completed the draft design evaluation, performed a consistency review of the SDS, and documented minor issues that need adjustment before final submission.

IX. Glossary

1. General Design Concepts

- **Software Design Specification** – The document defining system architecture, class structure, state behavior, and design logic.

- **Model–View–Controller (MVC)** – Architectural pattern separating UI (View), request routing (Controller), and business logic/data (Model).

2. Primary Devices

- **SafeHome Hub** – The pre-installed central hardware unit interacting with sensors and devices.

- **Sensor** – Physical input device (motion sensor, door/window, shock sensor, outdoor motion, dog barking detection).

- **Camera** – Provides live video, recording, and state transitions.

- **Smart Device / Actuator** – Devices controlled by the system (locks, alarms, etc., depending on your design scope).

3. Software Subsystems

- **Intelligent Security** – Handles sensor monitoring, incident management, alarm logic, and security policy.

- **Live Surveillance** – Manages camera viewing, recording, and storage.

- **System and User Management** – Manages system status, permissions, logs, and device configuration.

- **Remote Access and Account** – Handles authentication, login/logout, account data, and session rules.

- **Indoor Monitoring and Sound Control** – Manages sound detection and indoor monitoring conditions.

4. Manager, Controller, Repository, and Service Classes

a. Manager Classes

- **SensorMonitoringManager** – Processes all sensor inputs and related event flows.
- **IncidentManager** – Handles alarm conditions, incident responses, and escalations.
- **SecurityPolicyManager** – Controls arming, disarming, mode changes, sensor activation, and bypass.
- **SurveillanceManager** – Handles camera operations, recording, and live surveillance events.

b. Domain Entities & State Classes

- **Recording** – Represents a video recording instance with its own lifecycle.
- **RecordingSettings** – Stores video configuration parameters.
- **Camera** – Represents the camera's operational state and transitions.
- **UserAccount** – Represents an end user account, credentials, and configuration.

c. Repositories

- **StorageRepository** – Stores recordings and other multimedia data.
- **CameraRepository** – Stores camera configuration and metadata.

d. Controllers

- **SystemDeviceController** – Controls device registration and interactions.
- **SystemStatusController** – Provides system-wide operational state (armed/disarmed, online/offline).

- **SystemLogController** – Records events, changes, and audit log entries.
- **UserPermissionController** – Manages user roles, access levels, and authorization logic.

e. Authentication & Session Components

- **Authentication** – Validates login credentials and manages authentication rules.
- **Session Management** – Handles active user sessions, token lifecycle, and session expiration.

5. Security Policies & Rules

- **Security Mode** – Global mode defining system behavior (e.g., armed/disarmed).
- **Sensor Bypass** – Temporarily disables a sensor during arming or configuration.
- **Sensor Activation / Deactivation** – Enables or disables sensor monitoring.
- **Alarm Conditions** – Rules governing when alarms or notifications are raised.

6. Camera & Recording Operations

- **Live Surveillance** – Viewing real-time camera streams.
- **Recording** – Stored video footage triggered manually or by events.
- **Camera States** – The operational phases of a camera (On, Off, Idle, Recording).

7. User Roles

- **Homeowner (Primary User)** – Has full system control and configuration rights.
- **Guest User** – Has restricted access to certain system functions.
- **System Administrator (Software Actor)** – An automated system agent performing internal system actions (not a human).

8. General Technical Concepts

- **Repository** – A class that abstracts database access.
- **Gateway** – Connects software components to external services or devices.
- **State Diagram** – UML model describing state transitions and events for a class.
- **CRC Card** – Class-Responsibility-Collaboration card defining a class's purpose and collaborators.