Software Design Specification

Safehome Project

Team5

Sihun Chae (20190642)

Wooyoung Choi (20190659)

Donggeun Kim (20190074)

1. **Overview**
    1.1. Introduction
    1.2. Goal
    1.3. How the design work proceeded
    1.4. Assumptions
2. **Architectural Structure**
    2.1. Overall Architecture
    2.2. Intelligent Security
    2.3. Live Surveillance
    2.4. System and User Management
    2.5. Remote Access and Account
    2.6. Indoor Monitoring and Device Control
3. **Class Diagram**
    3.1. Whole System Overview
    3.2. Intelligent Security
    3.3. Live Surveillance
    3.4. System and User Management
    3.5. Remote Access and Account
    3.6. Pages
    3.7. Mobile Pages
    3.8. Indoor Monitoring and Device Control
4. **CRC Cards**
    4.1. Intelligent Security
    4.2. Live Surveillance
    4.3. System and User Management
    4.4. Remote Access and Account
    4.5. Pages
    4.6. Mobile Pages
    4.7. Indoor Monitoring and Device Control
5. **State Diagram**
    5.1. Intelligent Security
    5.2. Live Surveillance
    5.3. System and User Management
    5.4. Remote Access and Account
    5.5. Indoor Monitoring and Device Control
6. **Design Evaluation**
    6.1. Architectural Design Metric
    6.2. CK Metrics
    6.3. Mood Metric
    6.4. OO Metric Proposed by Lorenz and Kidd
    6.5. General Evaluation of Goal
7. **Who Did What**
8. **Meeting Logs**
9. **Appendix**
    9.1. Glossary

1. Overview

1.1. Introduction

This document presents the design model of the SafeHome system proposed in the previous report. As the design phase is directly linked to the implementation phase, this document emphasizes a well-structured and concrete design of the system. The architectural structure, class diagram, CRC cards, state diagrams, and sequence diagrams are provided to illustrate the overall system design.

1.2. Goal

1) Fully comply with the requirements and the analysis model.
2) Achieve low coupling, high cohesion, and modularity.
3) Pursue testability, integrity, efficiency, maintainability, and reliability.
4) Minimize complexity while considering reusability and flexibility.

1.3. How the design work proceeded

1) To achieve correctness of the design model, we used the method in chapter 8.7 of SEPA.
2) We reviewed nouns and verbs from use case scenarios to extract classes.
3) Based on the classes extracted and the use case scenario, we created the architectural structure of the SafeHome system.
4) On the basis of extracted classes and architectural structure, we created the class diagram considering the implementation.
5) We created the CRC card.
6) By testing the design using the CRC card and reviewing the requirement document and the first report, we refined the class diagram.
7) We focused on ways to achieve low coupling and high cohesion.
8) The actual implementation plan became more concrete and included some classes from Java and for database access.
9) We created the state diagram.
10) We refined the class diagram by adding some missing functions and attributes.
11) Based on the use case scenario, we created the sequence diagram.
12) It enabled us to check if the design followed the requirement specification and the first report.
13) We reviewed the state and sequence diagrams based on the first report.
14) We refined the class diagram again by adding some missing functions and attributes from the implementation viewpoint.

1.4. Assumptions

1) Pet Sensor Function is defined in safehome dialog slide 58-59
2) Alarm Trigger and Instant Notification Function is defined in safehome dialog slide 5

3) Emergency Service Integration and Auto Call Function is defined in safehome dialog slide 6, 7
4) One-Touch Modes (Away, Home, Sleep) Function is defined in safehome dialog slide 9
5) Sensor Bypass Function is defined in safehome dialog slide 10
6) Single Camera Live View Function is defined in safehome dialog slide 29-31
7) Two-Way Audio Function is defined in safehome dialog slide 16
8) Camera Lock and Unlock Function is defined in safehome dialog slide 19-31
9) Search and Playback Recordings Function is defined in safehome dialog slide 29-31
10) Recording Settings Function is defined in safehome dialog slide 29-31
11) Add and Place New Devices Function is defined in safehome dialog slide 58
12) Activity Logs and Timeline Function is defined in safehome dialog slide 39
13) User Role and Access Control Function is defined in safehome dialog slide 70
14) Sign Up Function is defined in safehome dialog slide 41
15) Log In Function is defined in safehome dialog slide 42
16) Log Out Function is defined in safehome dialog slide 43
17) Password Recovery and Reset Function is defined in safehome dialog slide 44
18) Edit Profile Information Function is defined in safehome dialog slide 45
19) Change Password Function is defined in safehome dialog slide 46
20) Two-Factor Authentication Management Function is defined in safehome dialog slide 47
21) Indoor Device Control Function is defined in safehome dialog slide 39
22) Indoor Air Quality Monitoring and Ventilation Integration Function is defined in safehome dialog slide 27
23) Real-Time Power Consumption Monitoring and Reporting Function is defined in safehome dialog slide 29
24) Secure Onboarding (Device Registration Security) Function was added because we determined it is necessary for improving the overall quality of the service. However, we have assumed that it should be developed in the next iteration, as the current SRS is for the first release.
25) OTA Firmware Update Function was added because we determined it is necessary for improving the overall quality of the service. However, we have assumed that it should be developed in the next iteration, as the current SRS is for the first release.
26) Health & Tamper Monitoring (Device Anomaly Detection) Function was added because we determined it is necessary for improving the overall quality of the service. However, we have assumed that it should be developed in the next iteration, as the current SRS is for the first release.
27) Global Priority & Version Policy Function was added because we determined it is necessary for improving the overall quality of the service. However, we have assumed that it should be developed in the next iteration, as the current SRS is for the first release.

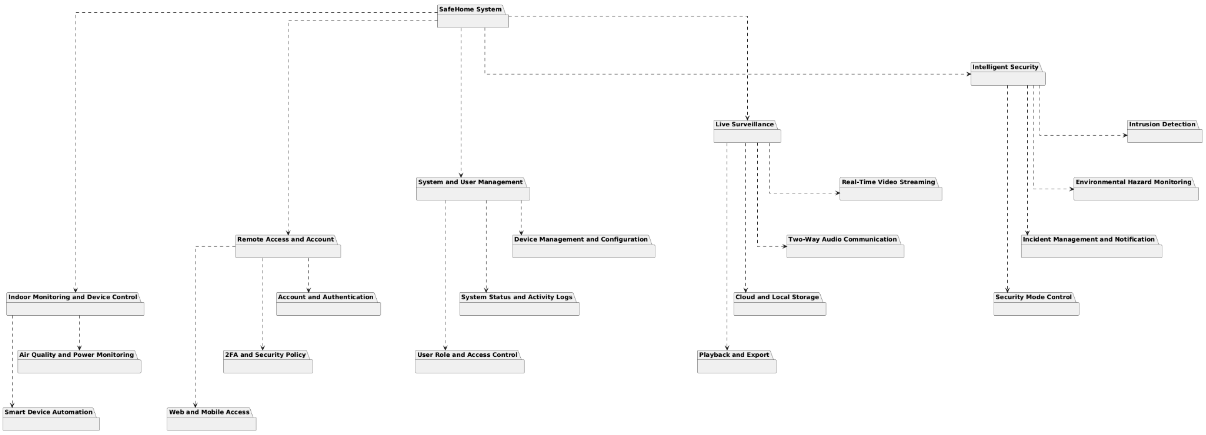
28) Policy & Compliance Function was added because we determined it is necessary for improving the overall quality of the service. However, we have assumed that it should be developed in the next iteration, as the current SRS is for the first release.
29) Data Retention & Deletion Policy Function was added because we determined it is necessary for improving the overall quality of the service. However, we have assumed that it should be developed in the next iteration, as the current SRS is for the first release.
30) Encryption/Transmission Security Policy Function was added because we determined it is necessary for improving the overall quality of the service. However, we have assumed that it should be developed in the next iteration, as the current SRS is for the first release.
31) Privacy Notice & Consent Function was added because we determined it is necessary for improving the overall quality of the service. However, we have assumed that it should be developed in the next iteration, as the current SRS is for the first release.
32) Time Synchronization (NTP) Policy Function was added because we determined it is necessary for improving the overall quality of the service. However, we have assumed that it should be developed in the next iteration, as the current SRS is for the first release.
33) It was decided to add the Physical Intrusion Detection Function in the team 1 meeting on 10.29 after discussion in the team 1 meeting on 10.26.
34) It was decided to add the Environmental Hazard Detection Function in the team 1 meeting on 10.29 after discussion in the team 1 meeting on 10.26.
35) It was decided to add the Outdoor Motion Detection Function in the team 1 meeting on 10.29 after discussion in the team 1 meeting on 10.26.
36) It was decided to add the Alarm Verification Step Function in the team 1 meeting on 10.29.
37) It was decided to add the Panic Button Function in the team 1 meeting on 10.29.
38) It was decided to add the Sensor Activation and Deactivation Function in the team 1 meeting on 10.29 after discussion in the team 1 meeting on 10.26.
39) It was decided to add the Camera Activation and Deactivation Function in the team 1 meeting on 10.29 after discussion in the team 1 meeting on 10.26.
40) It was decided to add the Evidence Sharing and Export Function in the team 1 meeting on 10.29.
41) It was decided to add the Notification Policy and Cooldown Function in the team 1 meeting on 10.29.
42) It was decided to add the System Status Dashboard Function in the team 1 meeting on 10.29.
43) Floor plan configuration and hardware deployment is complete and out of the scope of our project.

44) “System administrator” in our use case scenarios is not a person who is in charge of managing the system. It is the system itself acting as a facilitator for the use of system functionalities.

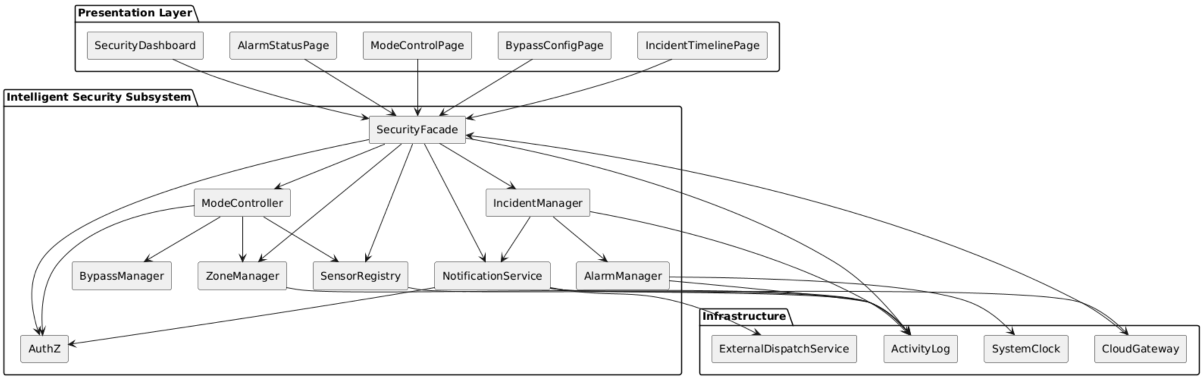
45) Between mobile and web, we have decided to make the mobile app our first release.

2. Architectural Structure

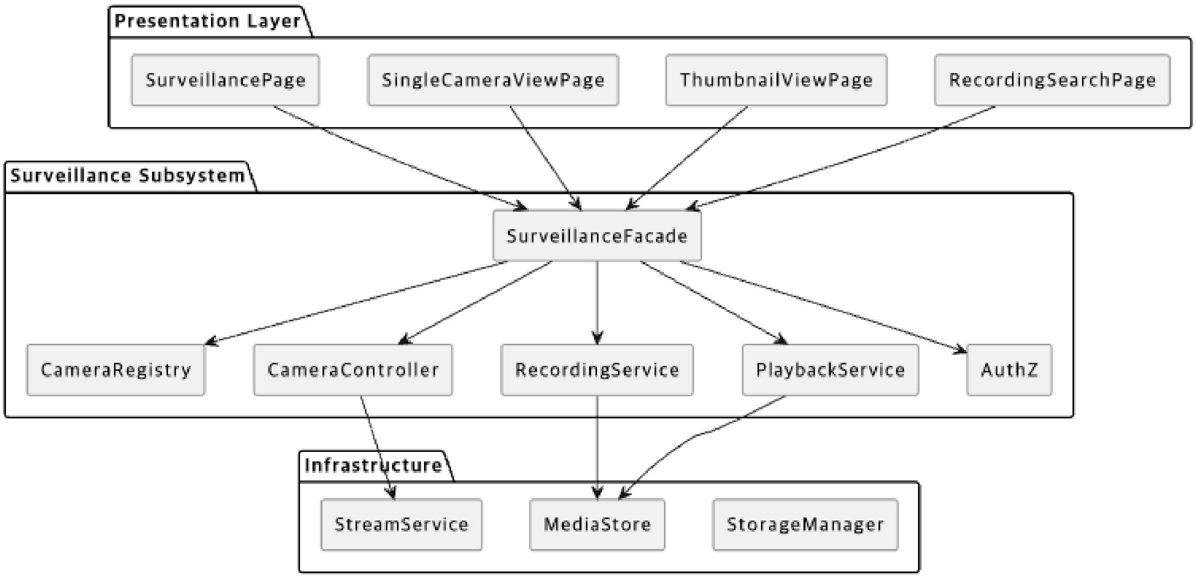
2.1. Overall Architecture


SafeHome System
Intelligent Security
Live Surveillance
Intrusion Detection
System and User Management
Real-Time Video Streaming
Environmental Hazard Monitoring
Remote Access and Account
Device Management and Configuration
Two-Way Audio Communication
Incident Management and Notification
Indoor Monitoring and Device Control
Account and Authentication
System Status and Activity Logs
Cloud and Local Storage
Security Mode Control
Air Quality and Power Monitoring
2FA and Security Policy
User Role and Access Control
Playback and Export
Smart Device Automation
Web and Mobile Access

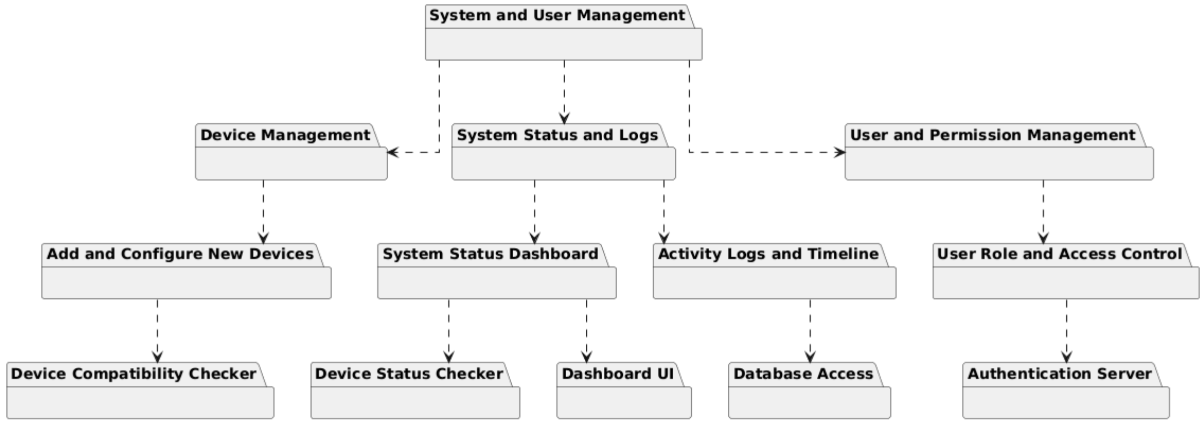
2.2. Intelligent Security



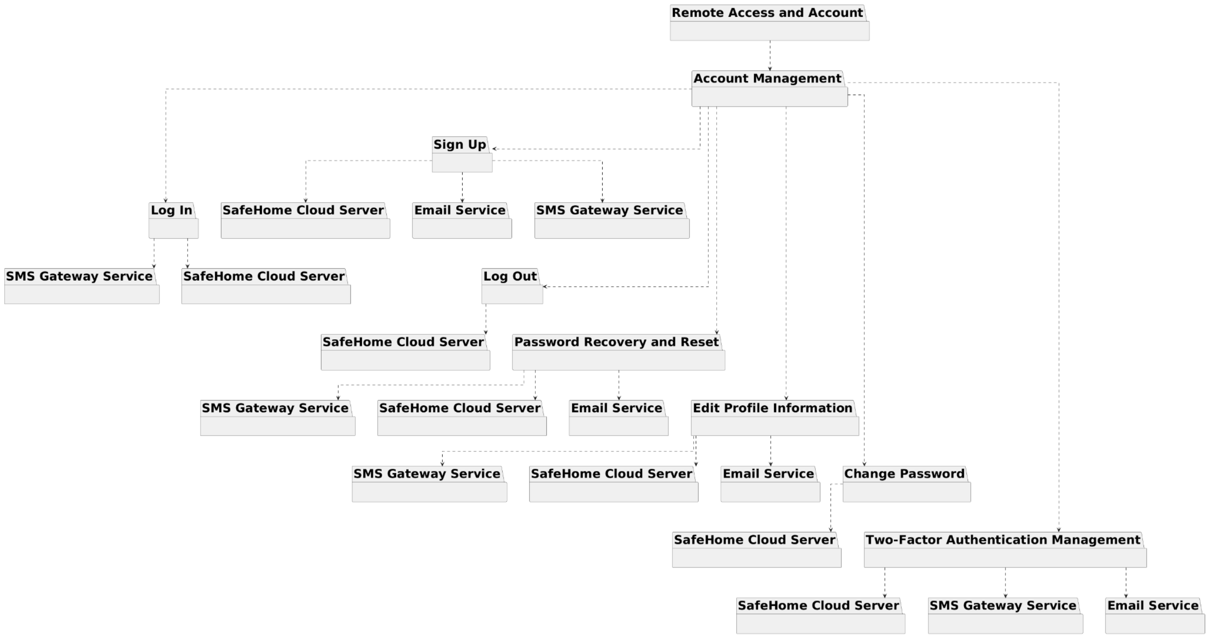
2.3. Live Surveillance



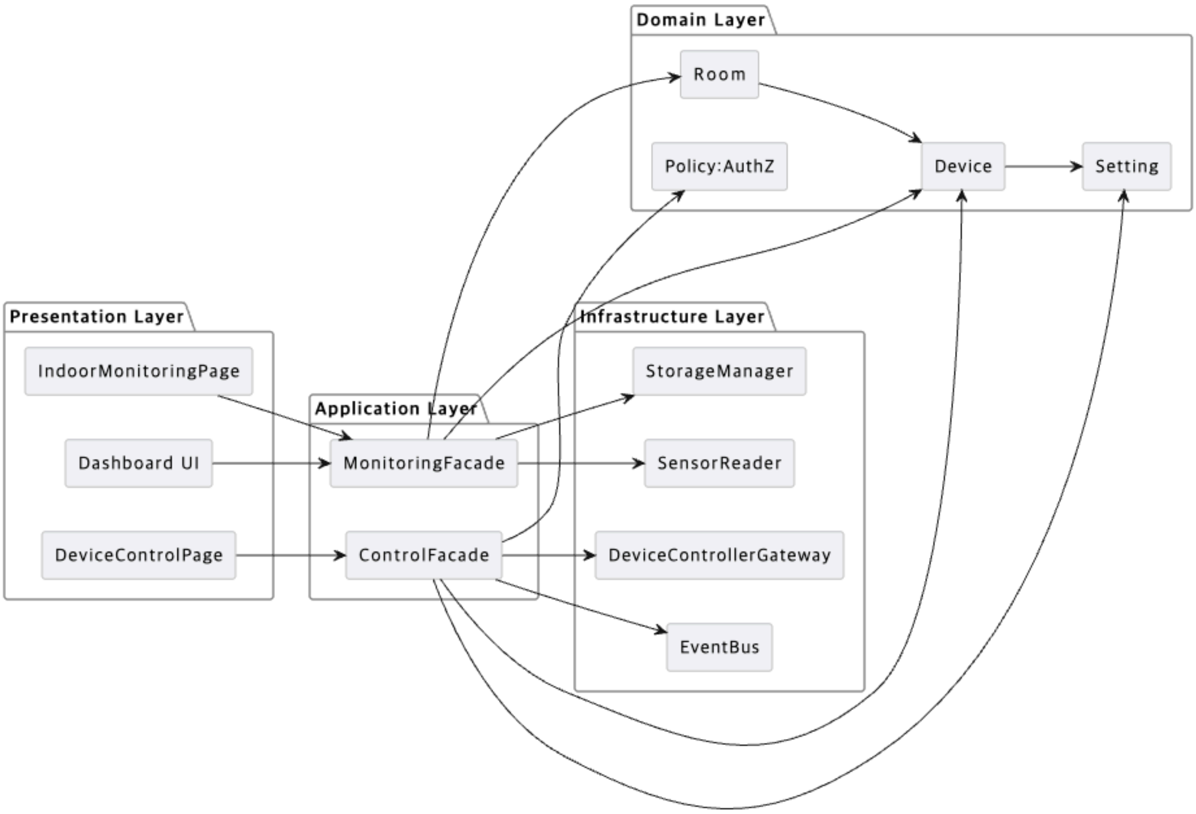
2.4. System and User Management


System and User Management
Device Management
System Status and Logs
User and Permission Management
Add and Configure New Devices
System Status Dashboard
Activity Logs and Timeline
User Role and Access Control
Device Compatibility Checker
Device Status Checker
Dashboard UI
Database Access
Authentication Server

2.5. Remote Access and Account

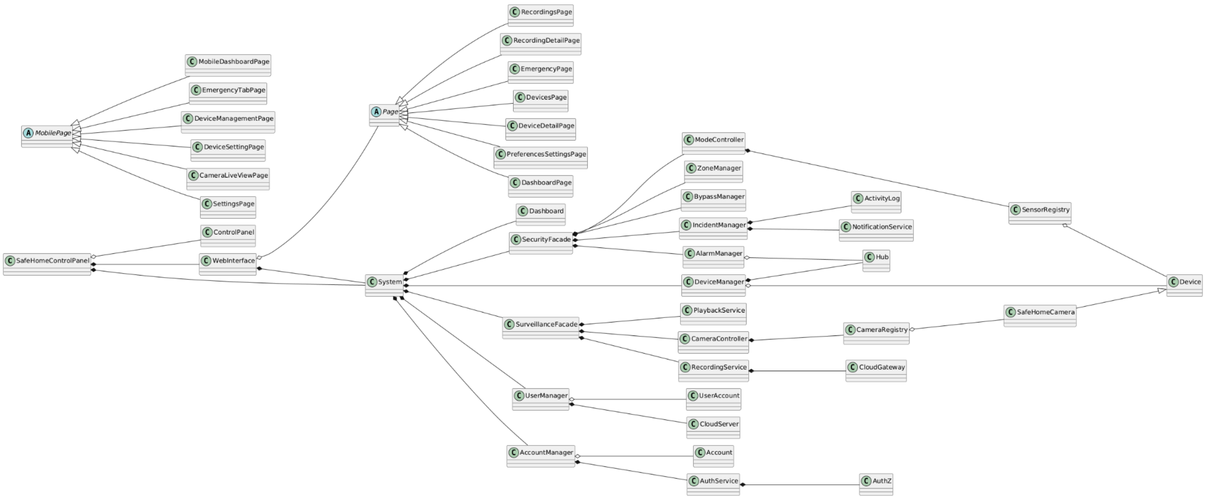


2.6. Indoor Monitoring and Device Control

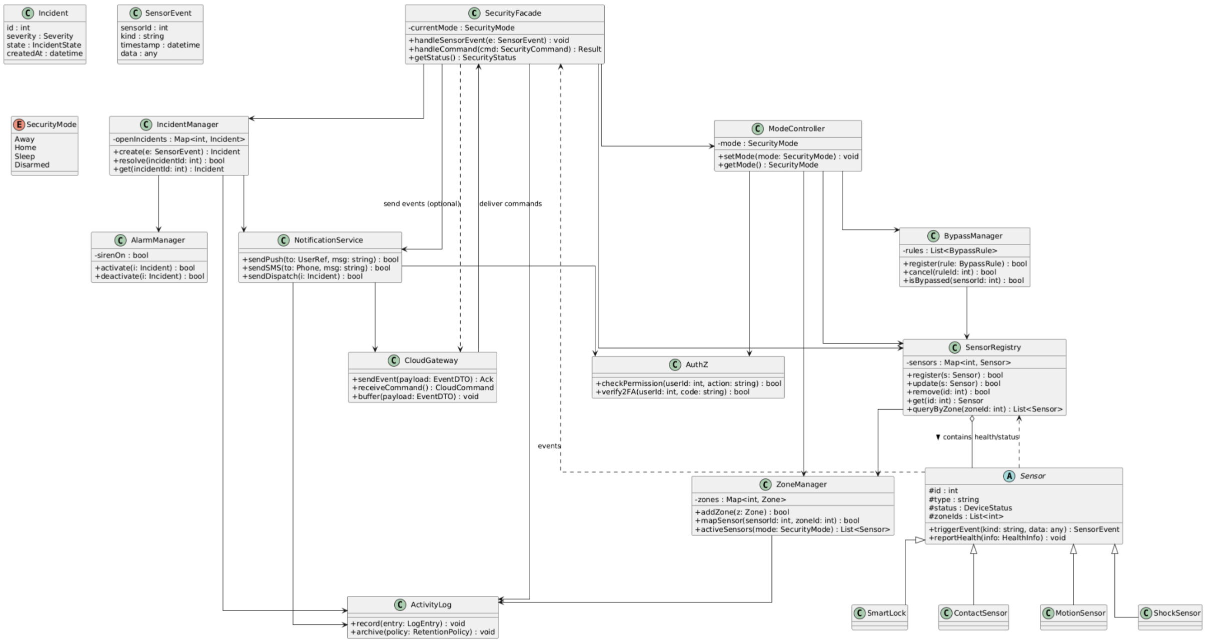


3. Class Diagram

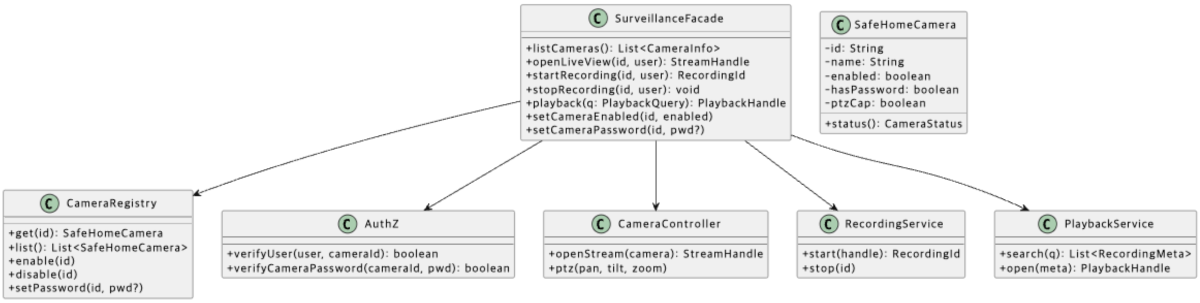
3.1. Whole System Overview



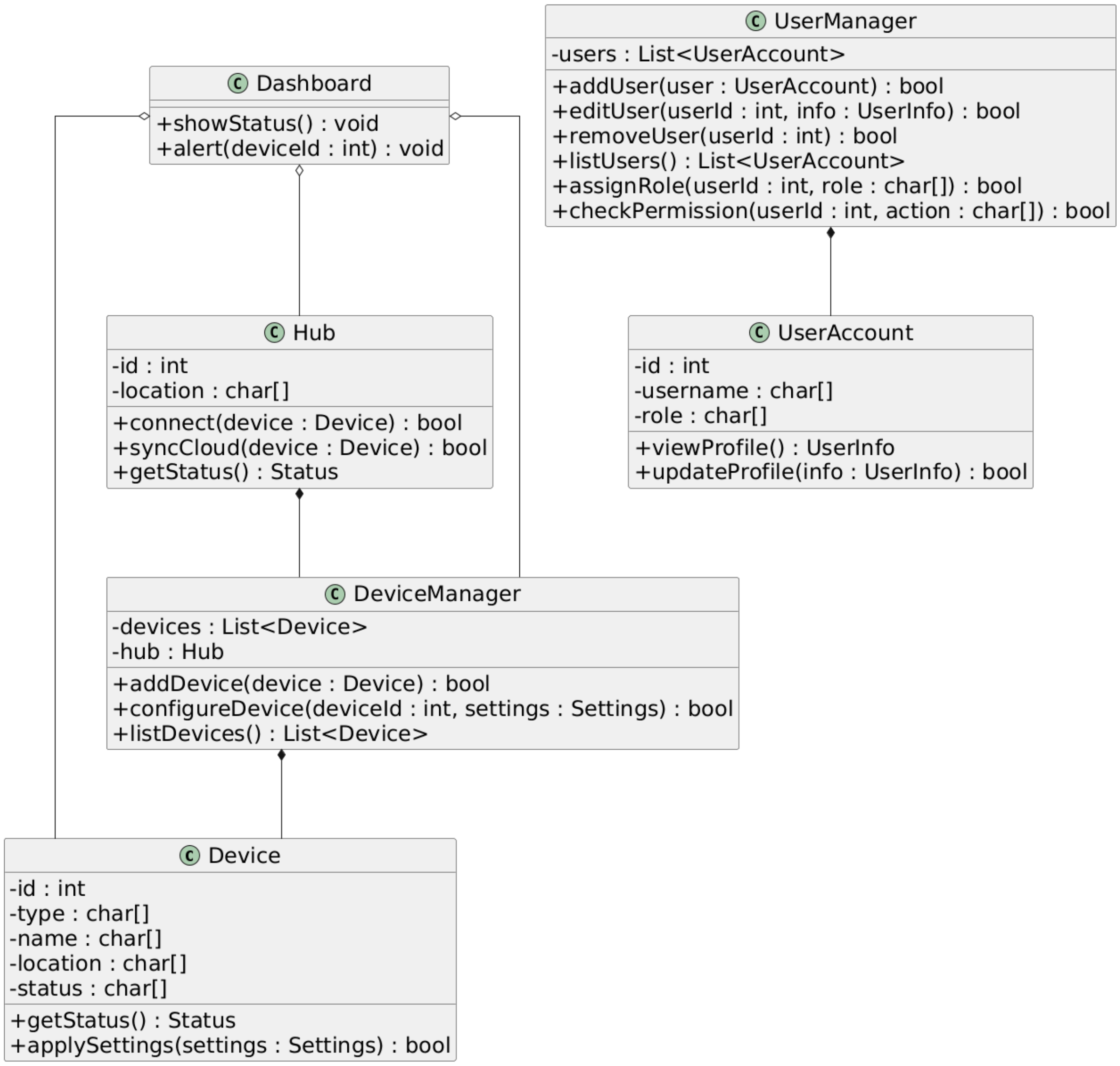
3.2. Intelligent Security


Incident
id : int
severity : Severity
state : IncidentState
createdAt : datetime
SensorEvent
sensorId : int
kind : string
timestamp : datetime
data : any
SecurityFacade
-currentMode : SecurityMode
+handleSensorEvent(e: SensorEvent) : void
+handleCommand(cmd: SecurityCommand) : Result
+getStatus() : SecurityStatus
SecurityMode
Away
Home
Sleep
Disarmed
IncidentManager
-openIncidents : Map<int, Incident>
+create(e: SensorEvent) : Incident
+resolve(incidentId: int) : bool
+get(incidentId: int) : Incident
ModeController
-mode : SecurityMode
+setMode(mode: SecurityMode) : void
+getMode() : SecurityMode
send events (optional)
deliver commands
AlarmManager
-sirenOn : bool
+activate(i: Incident) : bool
+deactivate(i: Incident) : bool
NotificationService
+sendPush(to: UserRef, msg: string) : bool
+sendSMS(to: Phone, msg: string) : bool
+sendDispatch(i: Incident) : bool
BypassManager
-rules : List<BypassRule>
+register(rule: BypassRule) : bool
+cancel(ruleId: int) : bool
+isBypassed(sensorId: int) : bool
CloudGateway
+sendEvent(payload: EventDTO) : Ack
+receiveCommand() : CloudCommand
+buffer(payload: EventDTO) : void
AuthZ
+checkPermission(userId: int, action: string) : bool
+verify2FA(userId: int, code: string) : bool
SensorRegistry
-sensors : Map<int, Sensor>
+register(s: Sensor) : bool
+update(s: Sensor) : bool
+remove(id: int) : bool
+get(id: int) : Sensor
+queryByZone(zoneId: int) : List<Sensor>
▼ contains
health/status
events
ZoneManager
-zones : Map<int, Zone>
+addZone(z: Zone) : bool
+mapSensor(sensorId: int, zoneId: int) : bool
+activeSensors(mode: SecurityMode) : List<Sensor>
Sensor
#id : int
#type : string
#status : DeviceStatus
#zoneIds : List<int>
+triggerEvent(kind: string, data: any) : SensorEvent
+reportHealth(info: HealthInfo) : void
ActivityLog
+record(entry: LogEntry) : void
+archive(policy: RetentionPolicy) : void
SmartLock
ContactSensor
MotionSensor
ShockSensor

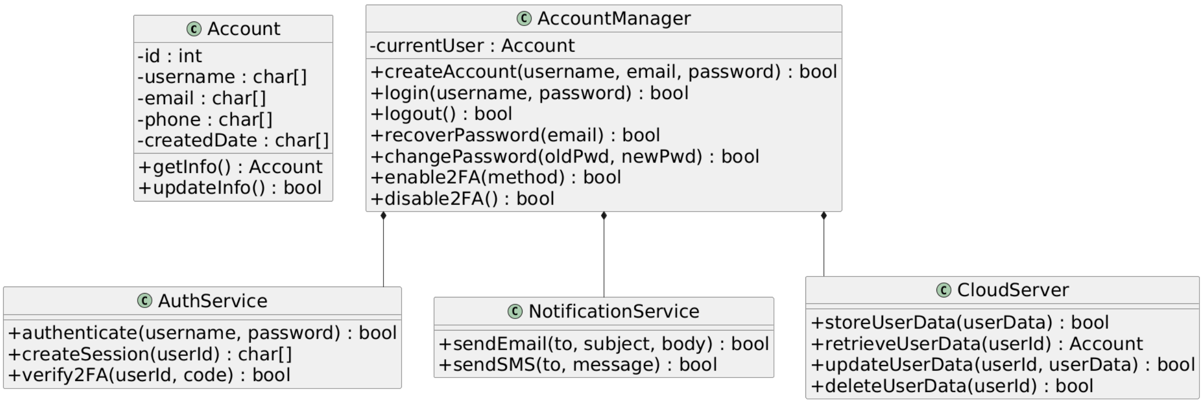
3.3. Live Surveillance


SurveillanceFacade
+listCameras(): List<CameraInfo>
+openLiveView(id, user): StreamHandle
+startRecording(id, user): RecordingId
+stopRecording(id, user): void
+playback(q: PlaybackQuery): PlaybackHandle
+setCameraEnabled(id, enabled)
+setCameraPassword(id, pwd?)
SafeHomeCamera
-id: String
-name: String
-enabled: boolean
-hasPassword: boolean
-ptzCap: boolean
+status(): CameraStatus
CameraRegistry
+get(id): SafeHomeCamera
+list(): List<SafeHomeCamera>
+enable(id)
+disable(id)
+setPassword(id, pwd?)
AuthZ
+verifyUser(user, cameraId): boolean
+verifyCameraPassword(cameraId, pwd): boolean
CameraController
+openStream(camera): StreamHandle
+ptz(pan, tilt, zoom)
RecordingService
+start(handle): RecordingId
+stop(id)
PlaybackService
+search(q): List<RecordingMeta>
+open(meta): PlaybackHandle

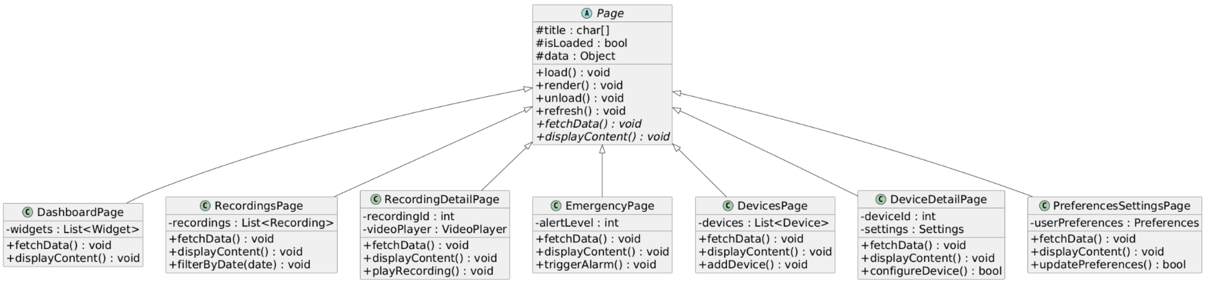
3.4. System and User Management


UserManager
-users : List<UserAccount>
+addUser(user : UserAccount) : bool
+editUser(userId : int, info : UserInfo) : bool
+removeUser(userId : int) : bool
+listUsers() : List<UserAccount>
+assignRole(userId : int, role : char[]) : bool
+checkPermission(userId : int, action : char[]) : bool
Dashboard
+showStatus() : void
+alert(deviceId : int) : void
Hub
-id : int
-location : char[]
+connect(device : Device) : bool
+syncCloud(device : Device) : bool
+getStatus() : Status
UserAccount
-id : int
-username : char[]
-role : char[]
+viewProfile() : UserInfo
+updateProfile(info : UserInfo) : bool
DeviceManager
-devices : List<Device>
-hub : Hub
+addDevice(device : Device) : bool
+configureDevice(deviceId : int, settings : Settings) : bool
+listDevices() : List<Device>
Device
-id : int
-type : char[]
-name : char[]
-location : char[]
-status : char[]
+getStatus() : Status
+applySettings(settings : Settings) : bool

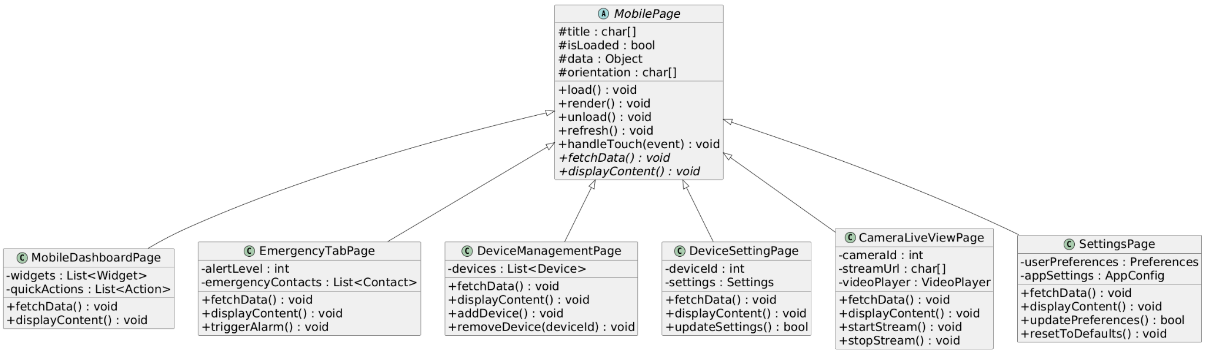
3.5. Remote Access and Account


Account
-id : int
-username : char[]
-email : char[]
-phone : char[]
-createdDate : char[]
+getInfo() : Account
+updateInfo() : bool
AccountManager
-currentUser : Account
+createAccount(username, email, password) : bool
+login(username, password) : bool
+logout() : bool
+recoverPassword(email) : bool
+changePassword(oldPwd, newPwd) : bool
+enable2FA(method) : bool
+disable2FA() : bool
AuthService
+authenticate(username, password) : bool
+createSession(userId) : char[]
+verify2FA(userId, code) : bool
NotificationService
+sendEmail(to, subject, body) : bool
+sendSMS(to, message) : bool
CloudServer
+storeUserData(userData) : bool
+retrieveUserData(userId) : Account
+updateUserData(userId, userData) : bool
+deleteUserData(userId) : bool

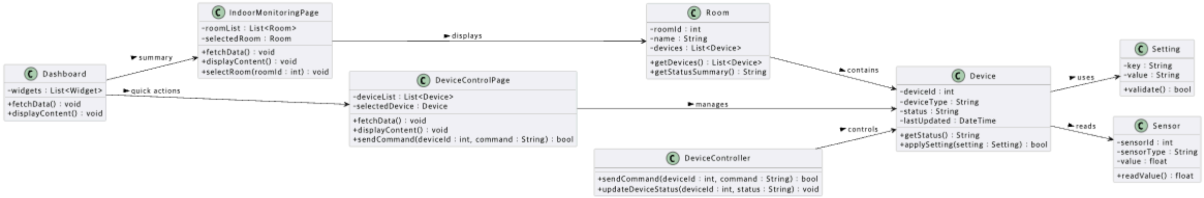
3.6. Pages



3.7. Mobile Pages



3.8. Indoor Monitoring and Device Control



4. CRC Cards

4.1. Intelligent Security

SecurityFacade	
Coordinates all security operations and exposes a unified API to the presentation layer	
Responsibilities	**Collaborators**
Receive and normalize sensor events	SensorRegistry
Apply security mode policies (Away / Home / Sleep)	ModeController
Trigger alarm or incident creation	IncidentManager
Route alerts to notification and dispatch services	NotificationService
Record event traces for audit and history	ActivityLog

<table>
<tr><th colspan="2">ModeController</th></tr>
<tr><td colspan="2">Controls the arming/disarming of the system and maintains active-sensor policies per mode</td></tr>
<tr><th>Responsibilities</th><th>Collaborators</th></tr>
<tr><td>Switch security modes and manage entry/exit delays</td><td>SecurityFacade</td></tr>
<tr><td>Activate / deactivate sensors according to mode</td><td>SensorRegistry</td></tr>
<tr><td>Handle temporary bypass rules</td><td>BypassManager</td></tr>
<tr><td>Resolve user command conflicts by priority</td><td>AuthZ</td></tr>
</table>

<table>
<tr><th colspan="2">SensorRegistry</th></tr>
<tr><td colspan="2">Central directory of all registered sensors and their operational status</td></tr>
<tr><th>Responsibilities</th><th>Collaborators</th></tr>
<tr><td>Register, update, and remove sensors</td><td>SecurityFacade</td></tr>
<tr><td>Provide sensor state and metadata queries</td><td>ModeController, IncidentManager</td></tr>
<tr><td>Map sensors to zones and manage multi-zone logic</td><td>ZoneManager</td></tr>
<tr><td>Report health, tamper, or offline conditions</td><td>NotificationService</td></tr>
</table>

<table>
<tr><th colspan="2">IncidentManager</th></tr>
<tr><td colspan="2">Handles creation, escalation, and lifecycle of security incidents</td></tr>
<tr><th>Responsibilities</th><th>Collaborators</th></tr>
</table>

Create new incident records upon verified triggers	SecurityFacade
Manage alarm activation and cooldown timing	AlarmManager
Persist results to system logs	ActivityLog

AlarmManager	
Controls all audible / visual alarm devices and ensures real-time responsiveness	
Responsibilities	**Collaborators**
Activate / deactivate local siren and indicators	SecurityFacade, IncidentManager
Enforce 3-second maximum activation delay	SystemClock
Handle hardware fault or offline fallback	NotificationService
Log alarm operations	ActivityLog

NotificationService	
Delivers emergency and status messages to users and external systems	
Responsibilities	**Collaborators**
Send push notifications and SMS alerts	CloudGateway
Retry failed notifications with escalation logic	AlarmManager
Provide delivery receipts to logs	ActivityLog
Integrate with two-factor authentication for secure access	AuthZ

<table>
<tr><th colspan="2">BypassManager</th></tr>
<tr><td colspan="2">Maintains temporary bypass settings for sensors and automatically restores them when expired</td></tr>
<tr><th>Responsibilities</th><th>Collaborators</th></tr>
<tr><td>Register / cancel / expire bypass requests</td><td>ModeController</td></tr>
<tr><td>Prevent conflict between overlapping bypass zones</td><td>SensorRegistry</td></tr>
<tr><td>Notify user when bypass expires</td><td>NotificationService</td></tr>
</table>

<table>
<tr><th colspan="2">CloudGateway</th></tr>
<tr><td colspan="2">Acts as a communication bridge between the local SafeHome hub and the cloud server</td></tr>
<tr><th>Responsibilities</th><th>Collaborators</th></tr>
<tr><td>Transmit incident and alarm events to the cloud</td><td>SecurityFacade, NotificationService</td></tr>
<tr><td>Receive remote commands (e.g., disarm, mode change)</td><td>SecurityFacade, ModeController</td></tr>
<tr><td>Handle offline buffering and retry logic</td><td>IncidentManager</td></tr>
<tr><td>Enforce encryption and secure channel policies</td><td>AuthZ</td></tr>
</table>

<table>
<tr><th colspan="2">ZoneManager</th></tr>
<tr><td colspan="2">Manages zone definitions and sensor-to-zone mapping according to the fixed floor plan</td></tr>
<tr><th>Responsibilities</th><th>Collaborators</th></tr>
<tr><td>Maintain zones and include sensors by red-dot rule</td><td>SensorRegistry</td></tr>
</table>

Determine active/inactive per zone and shared-sensor logic	ModeController
Provide queries: sensors in zone, zones for sensor	SecurityFacade
Persist zone changes and audit	ActivityLog

ActivityLog	
Maintains immutable event and audit logs for the Intelligent Security subsystem	
Responsibilities	**Collaborators**
Record all detection, alarm, and notification events	SecurityFacade, IncidentManager
Tag each record with timestamp, device ID, and user context	SensorRegistry, AuthZ
Support long-term archival and retention policy	CloudGateway

4.2. Live Surveillance

SurvilanceFacade	
Coordinates all surveillance operations and exposes a unified API to the presentation layer	
Responsibilities	**Collaborators**
Retrieve available cameras	Camera Registry
Open live view for a camera	CameraController
Start or stop recoding	RecordingService
Authorize access to protected cameras	AuthZ

<table>
<tr><th colspan="2">CameraRegistry</th></tr>
<tr><td colspan="2">Manage camera information and configuration</td></tr>
<tr><th>Responsibilities</th><th>Collaborators</th></tr>
<tr><td>Look up a camera by ID</td><td>SafeHomeCamera</td></tr>
<tr><td>Provide a list of all cameras</td><td>Storage Manager</td></tr>
<tr><td>Update camera enable/disable state</td><td>SafeHomeCamera</td></tr>
<tr><td>Update camera password</td><td>SafeHomeCamera</td></tr>
</table>

<table>
<tr><th colspan="2">CameraController</th></tr>
<tr><td colspan="2">Directly controls camera streaming and PTZ actions</td></tr>
<tr><th>Responsibilities</th><th>Collaborators</th></tr>
<tr><td>Oepn a camera stream</td><td>StreamService</td></tr>
<tr><td>Execute pan/tilt/zoom</td><td>StreamService</td></tr>
<tr><td>Query camera’s operational status</td><td>SafeHomeCamera</td></tr>
</table>

<table>
<tr><th colspan="2">RecordingService</th></tr>
<tr><td colspan="2">Handles creation and maintenance of recording sessions</td></tr>
<tr><th>Responsibilities</th><th>Collaborators</th></tr>
<tr><td>Start recording from an active stream</td><td>MediaStore</td></tr>
<tr><td>Stop recording and finilize file</td><td>MediaStore</td></tr>
<tr><td>Store recording metadata</td><td>Storage Manager</td></tr>
</table>

PlaybackService

Supports retrieving and playing recorded videos	
Responsibilities	**Collaborators**
Search for recodrings	Storage Manager
Retrieve recording file	MediaStore
Provide playback handle	MediaStore

AuthZ	
Provides access control for camera operations	
Responsibilities	**Collaborators**
Validate camera password	CameraRegistry
Check user access permissions	CameraRegistry
Enforce lockout after failed attempts	Survillance Facade

SafeHomeCamera	
Represents a single camera’s internal state	
Responsibilities	**Collaborators**
Return camera status	Camera Registry
Store enable/disable state	Camera Registry
Maintain password requirement flag	Camera Registry

4.3. System and User Management

Device
Manage individual devices in the SafeHome system, including registration, status, and settings.

Responsibilities	**Collaborators**
Provide current status	Dashboard, Hub
Apply settings	DeviceManager

DeviceManager	
Manage all devices, including adding, configuring, and listing them.	
Responsibilities	**Collaborators**
Add new devices	Device
Configure device settings	Device, Hub
List all devices	Dashboard
Manage device-hub relationships	Hub

Hub	
Manage connections with devices and synchronize their data with the cloud.	
Responsibilities	**Collaborators**
Connect devices	Device
Sync device data with cloud	Device
Provide overall system status	Dashboard, DeviceManager

Dashboard	
Monitor the system by displaying statuses and triggering alerts.	
Responsibilities	**Collaborators**
Show device and hub status	DeviceManager, Hub, Device
Trigger alerts for devices	DeviceManager, Device

UserAccount

Represent individual users and manage their profile information.	
Responsibilities	**Collaborators**
View profile	UserManager
Update profile	UserManager

UserManager	
Manage users, including creation, modification, deletion, and role assignments.	
Responsibilities	**Collaborators**
Add, edit, remove users	UserAccount
List all users	UserAccount
Assign roles	UserAccount
Check permissions	UserAccount

4.4. Remote Access and Account

Account	
Represent a user account, storing personal and authentication-related information.	
Responsibilities	**Collaborators**
Store account information	AccountManager, CloudServer
Provide account information	AccountManager
Update account information	AccountManager, CloudServer

AccountManager	
Manage account lifecycle, authentication, and security settings.	
Responsibilities	**Collaborators**
Create new accounts	Account, AuthService, CloudServer
Login and logout users	AuthService
Recover and change passwords	AuthService, NotificationService

Enable and disable 2FA	AuthService, NotificationService
Maintain current user session	AuthService

AuthService	
Handle authentication, sessions, and two-factor verification.	
Responsibilities	**Collaborators**
Authenticate users	AccountManager
Create user sessions	AccountManager
Verify 2FA codes	AccountManager

NotificationService	
Send notifications via email or SMS.	
Responsibilities	**Collaborators**
Send email notifications	AccountManager
Send SMS notifications	AccountManager

CloudServer	
Store, retrieve, update, and delete user account data.	
Responsibilities	**Collaborators**
Store user data	AccountManager
Retrieve user data	AccountManager, Account
Update user data	AccountManager
Delete user data	AccountManager

4.5. Pages

Page (Abstract)

Provides common page lifecycle management and rendering framework for all pages	
Responsibilities	**Collaborators**
Manage page loading and unloading lifecycle	
Handle page refresh operations	
Store page title and load state	
Maintain page data object	

DashboardPage	
Displays real-time system status with widgets for quick overview	
Responsibilities	**Collaborators**
Fetch system status and device data	DeviceManager, Hub
Display collection of status widgets	Widget
Render real-time device conditions	Device
Provide quick access to key features	Dashboard
Update widgets with fresh data	DeviceManager

RecordingsPage	
Manages and displays list of recorded surveillance footage	
Responsibilities	**Collaborators**
Fetch list of available recordings	CloudServer
Display recordings in organized view	-
Filter recordings by date criteria	-
Provide navigation to recording details	RecordingDetailPage
Show recording thumbnails and metadata	-

RecordingDetailPage

Plays and manages individual video recording	
Responsibilities	**Collaborators**
Load specific recording by ID	CloudServer
Initialize and control video player	-
Display recording metadata	-
Enable video playback controls	-
Provide recording download option	CloudServer

EmergencyPage	
Handles emergency situations and alarm triggers	
Responsibilities	**Collaborators**
Fetch current alert level status	IncidentManager
Display emergency alert information	IncidentManager
Trigger emergency alarm system	Hub
Contact emergency services	NotificationService
Show emergency response options	DeviceManager

DevicesPage	
Manages list of all connected security devices	
Responsibilities	**Collaborators**
Fetch all registered devices	DeviceManager
Display device list with status	Device
Add new device to system	DeviceManager, Hub
Provide navigation to device details	DeviceDetailPage
Show device connectivity status	Device

DeviceDetailPage	
Displays and configures individual device settings	
Responsibilities	**Collaborators**
Load device data by ID	Device
Display device settings interface	Settings
Configure device parameters	Device, DeviceManager
Apply and validate new settings	Device
Show device activity history	ActivityLog

PreferencesSettingsPage	
Manages user preferences and system settings	
Responsibilities	**Collaborators**
Fetch user preferences data	UserAccount
Display preferences configuration form	Preferences
Update user preferences	UserAccount, CloudServer
Validate preference changes	Preferences
Provide reset to defaults option	Preferences

4.6. Mobile Pages

MobilePage (Abstract)	
Defines common structure and behavior for all mobile pages	
Responsibilities	**Collaborators**
Manage page lifecycle (load, render, unload, refresh)	-
Handle touch events	-
Provide abstract interface for data fetching and content display	-
Store and manage page metadata (title,	-

orientation, data)	

MobileDashboardPage	
Displays a mobile-optimized dashboard with system widgets and quick actions	
Responsibilities	**Collaborators**
Fetch and display system status data	MobilePage
Render widgets and quick action buttons	Widget, Action
Provide real-time overview of device conditions	Device
Handle updates and refresh interactions	Hub

EmergencyTabPage	
Provides quick access to emergency controls and alerts	
Responsibilities	**Collaborators**
Fetch and display current alert level and contacts	MobilePage
Trigger alarms and broadcast emergency signals	Hub
Display emergency contact list	UserAccount
Handle panic button interactions	IncidentManager, NotificationService

DeviceManagementPage	
Manages mobile view for listing and controlling connected devices	
Responsibilities	**Collaborators**
Fetch and display list of all connected devices	MobilePage
Add new devices via scanning or registration	Hub
Remove devices with user confirmation	DeviceManager

Navigate to device settings	DeviceSetting

DeviceSettingPage	
Handles configuration of individual devices through the mobile interface	
Responsibilities	**Collaborators**
Load and display device configuration	Device
Update and validate device settings	DeviceManager
Save modified settings	CloudServer
Display feedback to user after updates	Hub

CameraLiveViewPage	
Streams real-time video from security cameras on mobile	
Responsibilities	**Collaborators**
Fetch live camera feed	Device
Manage video playback (start, stop, display)	-
Handle stream optimization for mobile bandwidth	Hub
Display PTZ (pan-tilt-zoom) controls	Device
Capture and save snapshots	CloudServer

SettingsPage	
Manages mobile app preferences and user configurations	
Responsibilities	**Collaborators**
Fetch and display user and app settings	MobilePage
Update user preferences and app configurations	AccountManager
Reset settings to default values	CloudServer

Manage notification preferences	NotificationService
Synchronize settings with the cloud	CloudServer

4.7. Indoor Monitoring and Device Control

SurveilanceFacade	
Coordinates all surveillance use cases through a single entry point.	
Responsibilities	**Collaborators**
Open single-camera live view	Camera Controller
Start/stop manual recording	Recording Service
Enable/disable a camera	Camera Registry
Verify per-camera access (password/user)	AuthZ

CameraRegistry	
Maintain registration and operational flags for cameras	
Responsibilities	**Collaborators**
Look up camera metadata by ID	Storage Manager
Update enabled/disabled status	SafeHomeCamera
Persist per-camera password hash/policy	Credential Store

SafeHomeCamera	
Represents an individual camera and its capabilities / state	
Responsibilities	**Collaborators**
Provide current status (enabled/disabled, PTZ cap)	Surveillance Facade

Transition enable <-> disable	Camera Registry
Expose password-required flag for access checks	AuthZ

CameraController	
Talk to devices / services to open stream and perform PTZ	
Responsibilities	**Collaborators**
Open/close live stream	StreamService
Apply PTZ (pan/tilt/zoom)	CameraDevice
Poll/refreh camera health/state	CameraRegistry

RecordingService	
Manage manual recording lifecycle and clip persistence	
Responsibilities	**Collaborators**
Start recording from a live stream	MediaStore
Stop/finalize and index a clip	RecordingIndexer
Report storage/quota status	StorageManager

PlaybackService	
Provide search and playback for stored clips	
Responsibilities	**Collaborators**
Search recordings by camera/time	MediaStore
Open playback session for a clip	PlaybackAdapter
Control playback (play/pause/seek)	SurveillanceFacade

AuthZ	
Enforce user/camera access rules and lockouts	
Responsibilities	**Collaborators**
Verify user’s access to camera	UserService
Validate per-camera password	CredentialStore
Apply retry/lockout policy	LockoutStore

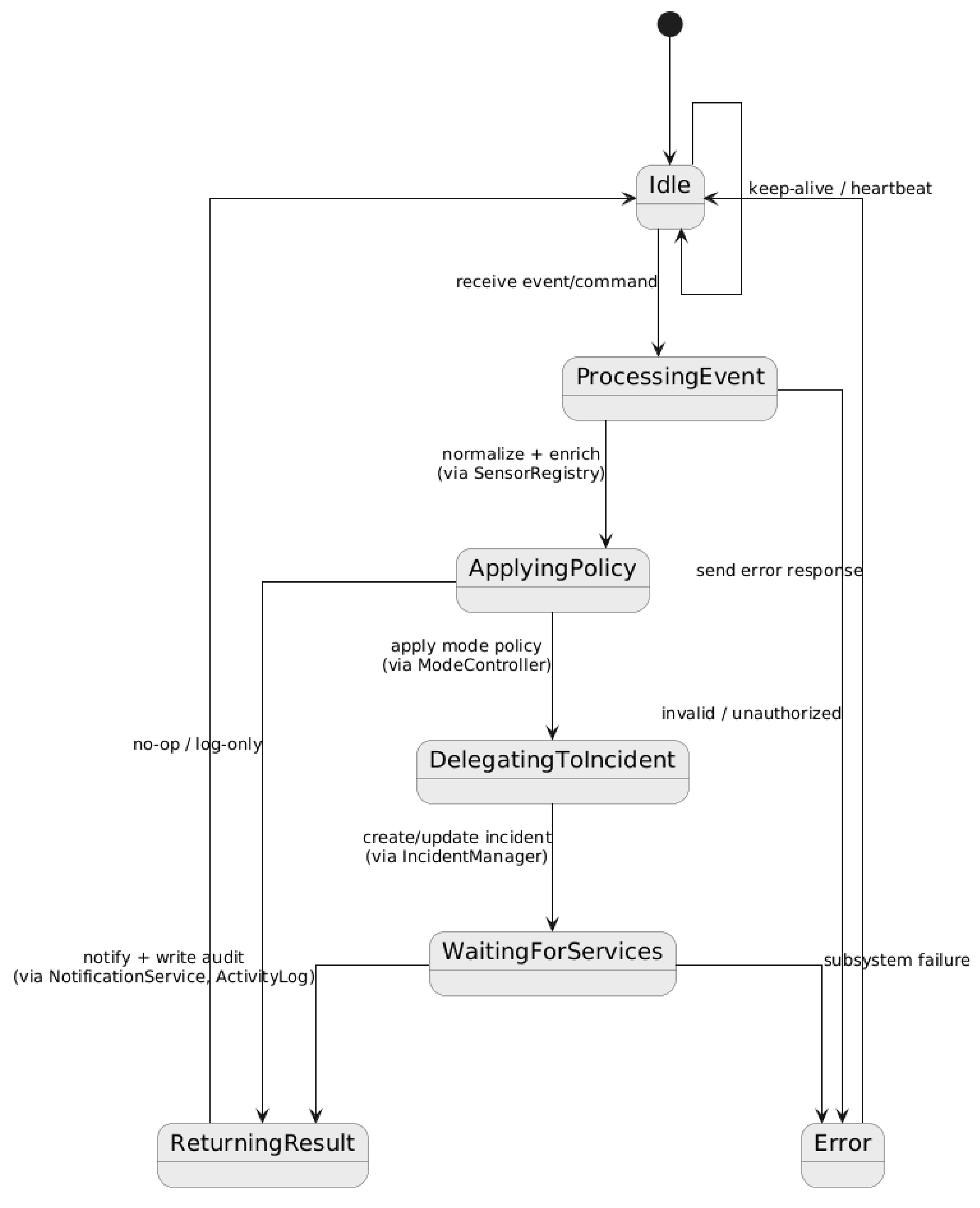
IndoorMonitoring	
High-level indoor monitoring workflow (multi-tile -> single view)	
Responsibilities	**Collaborators**
List indoor cameras with status / tiles	CameraRegistry
Open single-camera view from tiles	ThumbnailService
Overlay alerts on active views	NotificationBus

DeviceControl	
Generic device control within SafeHome (lights, plugs, etc.).	
Responsibilities	**Collaborators**
Toggle device operational state	DeviceManager
Apply a scene/preset to multiple devices	SceneEngine
Read device telemetry/health	Hub

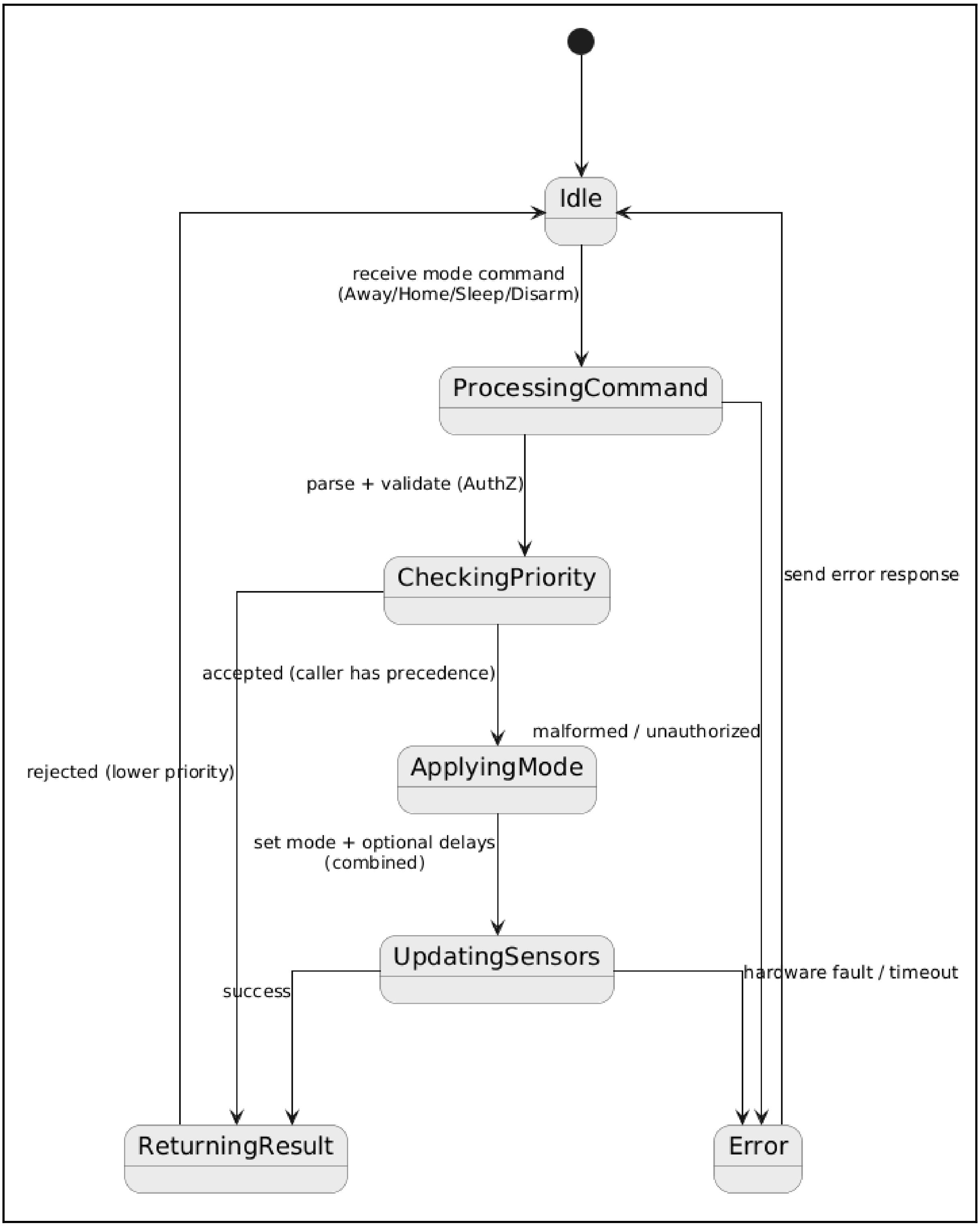
5. State Diagram

5.1. Intelligent Security

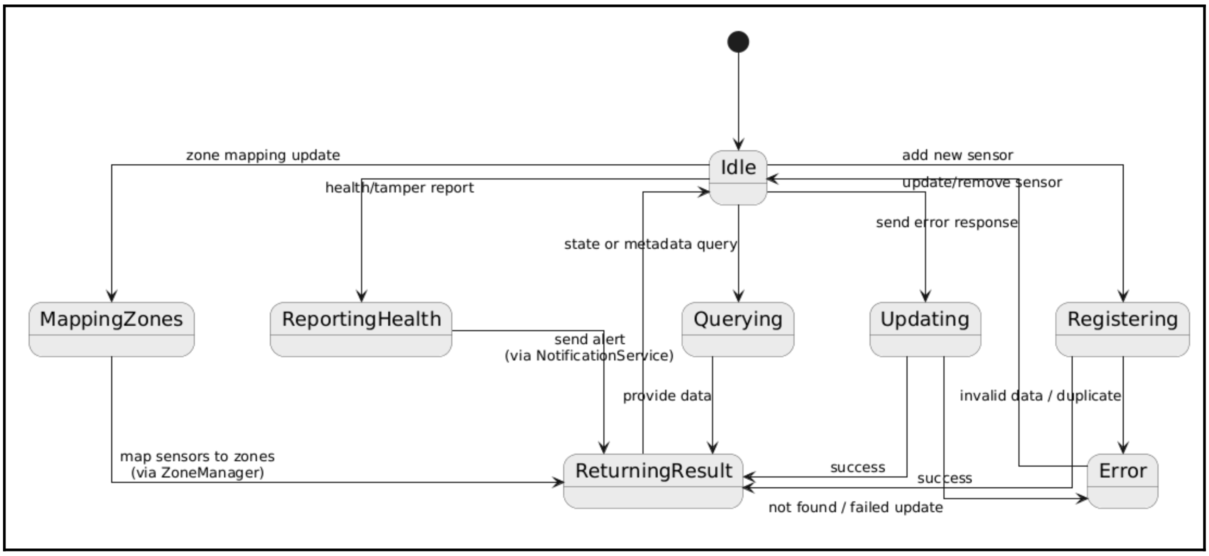
SecurityFacade



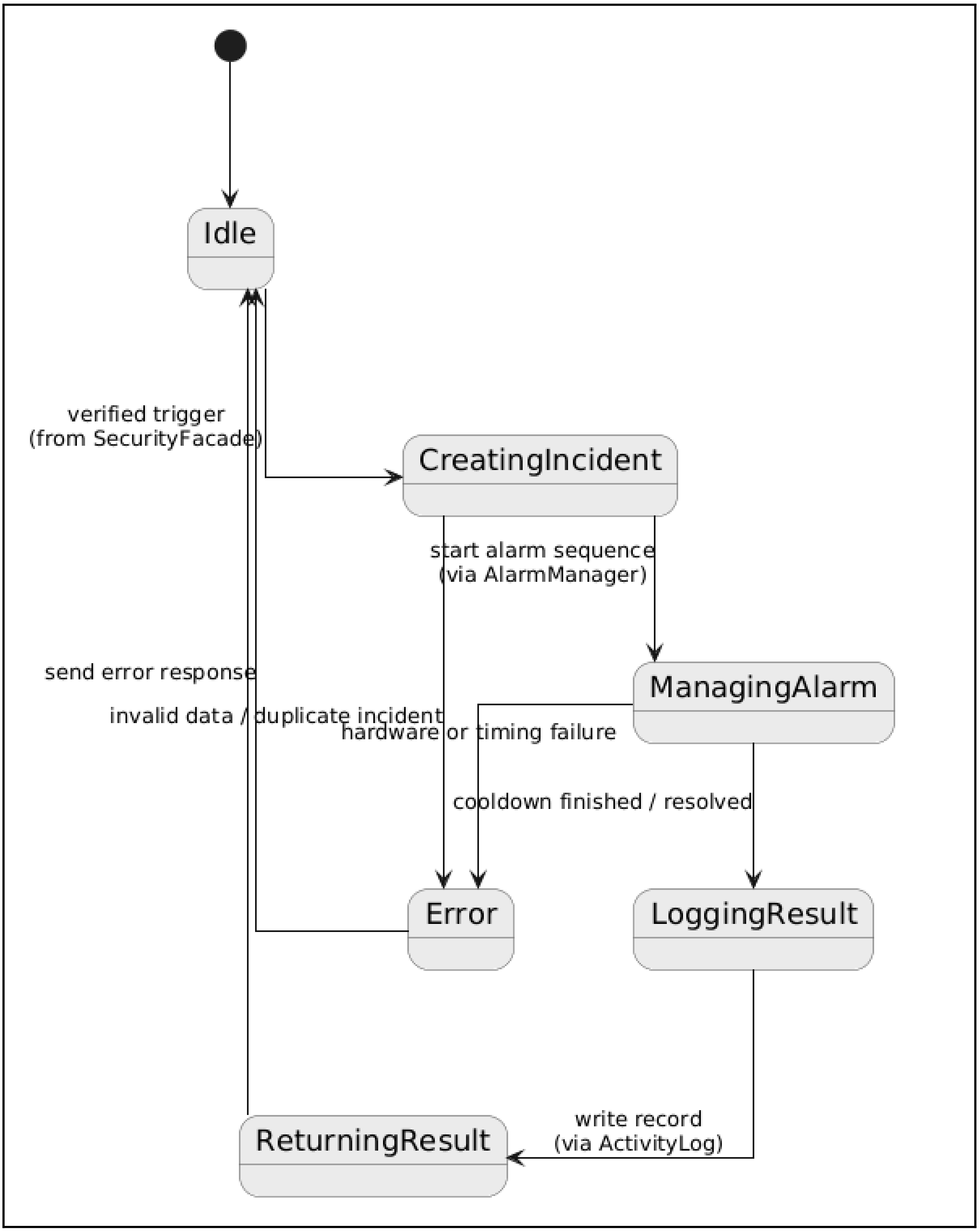
ModeController



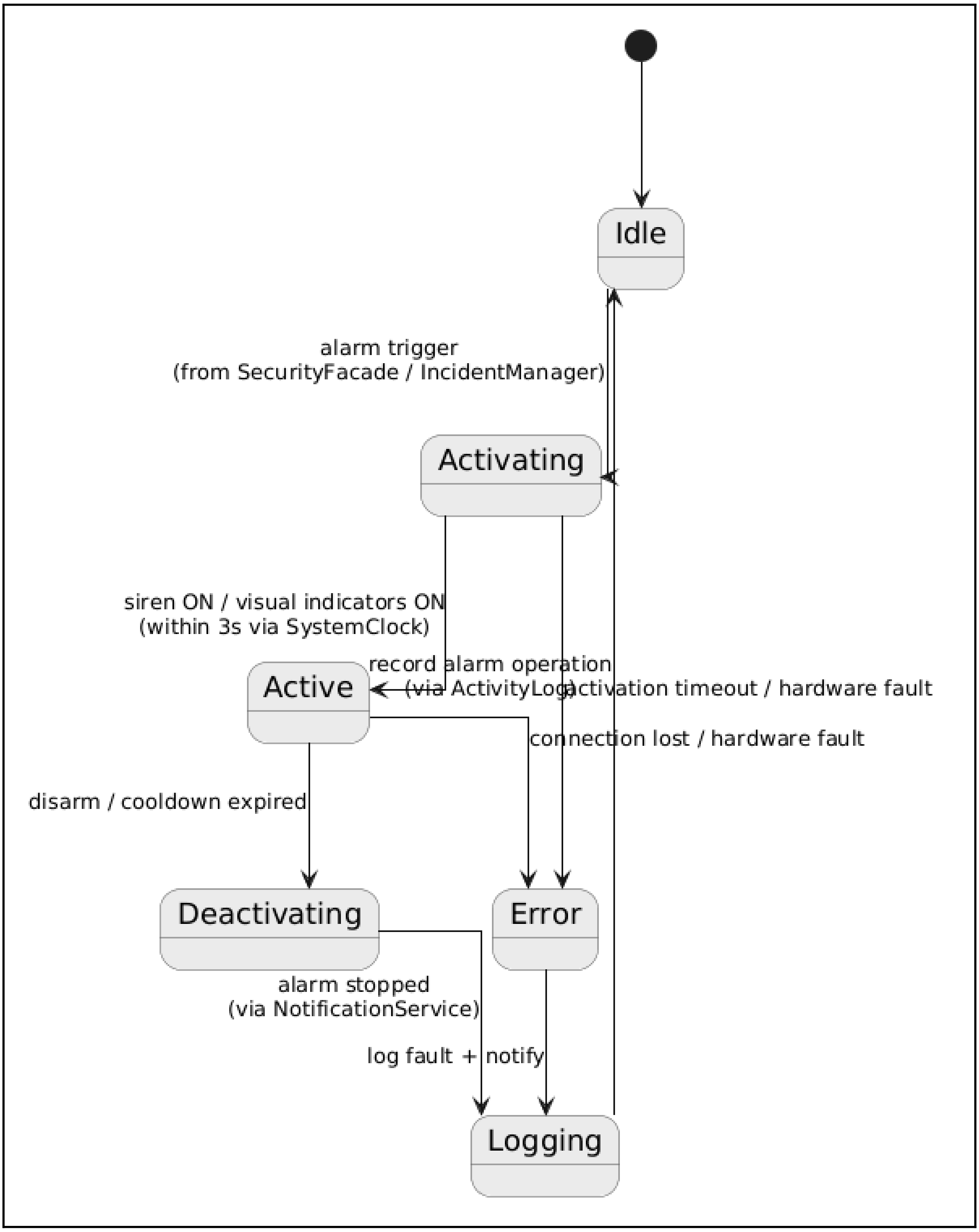
SensorRegistry



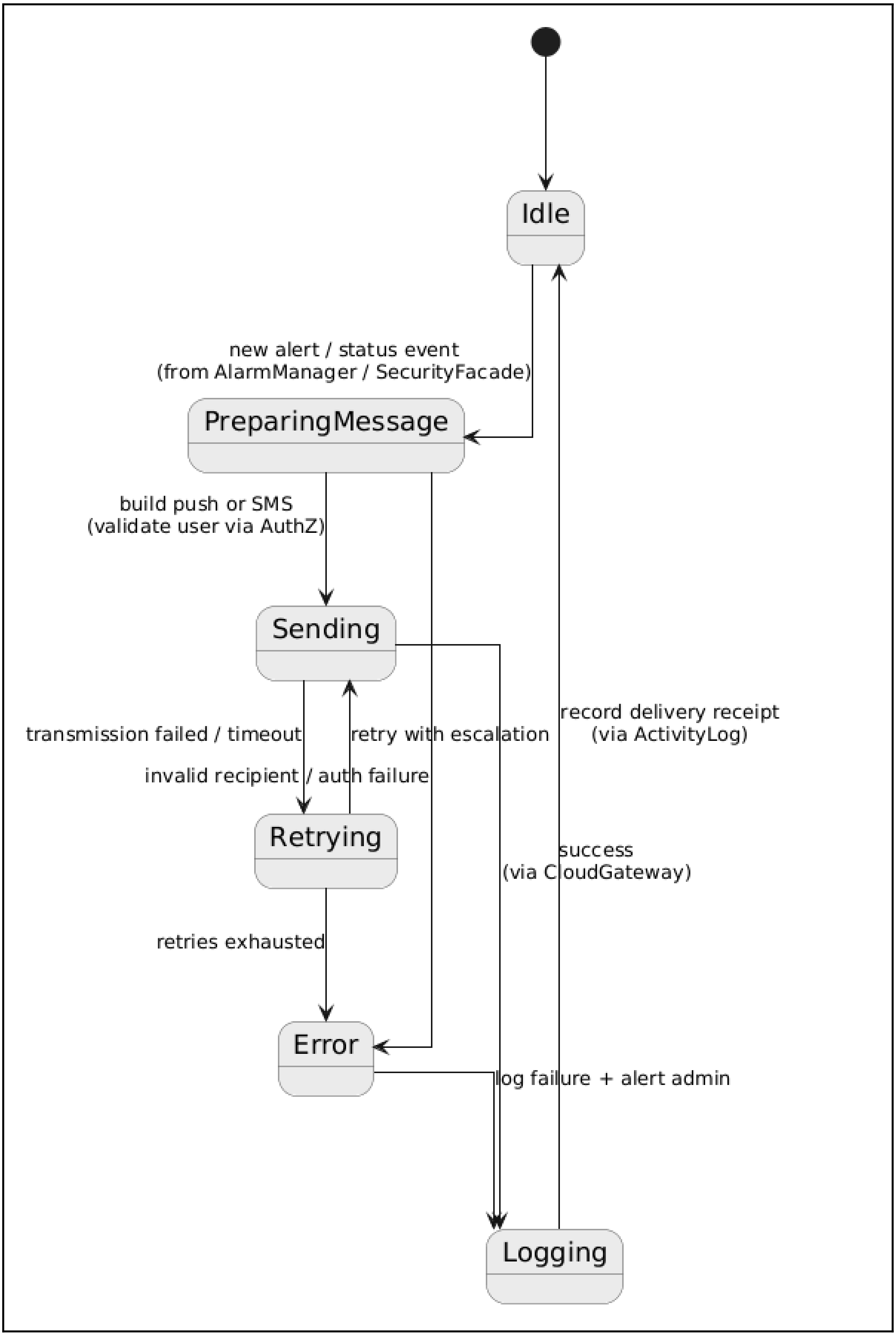
IncidentManager



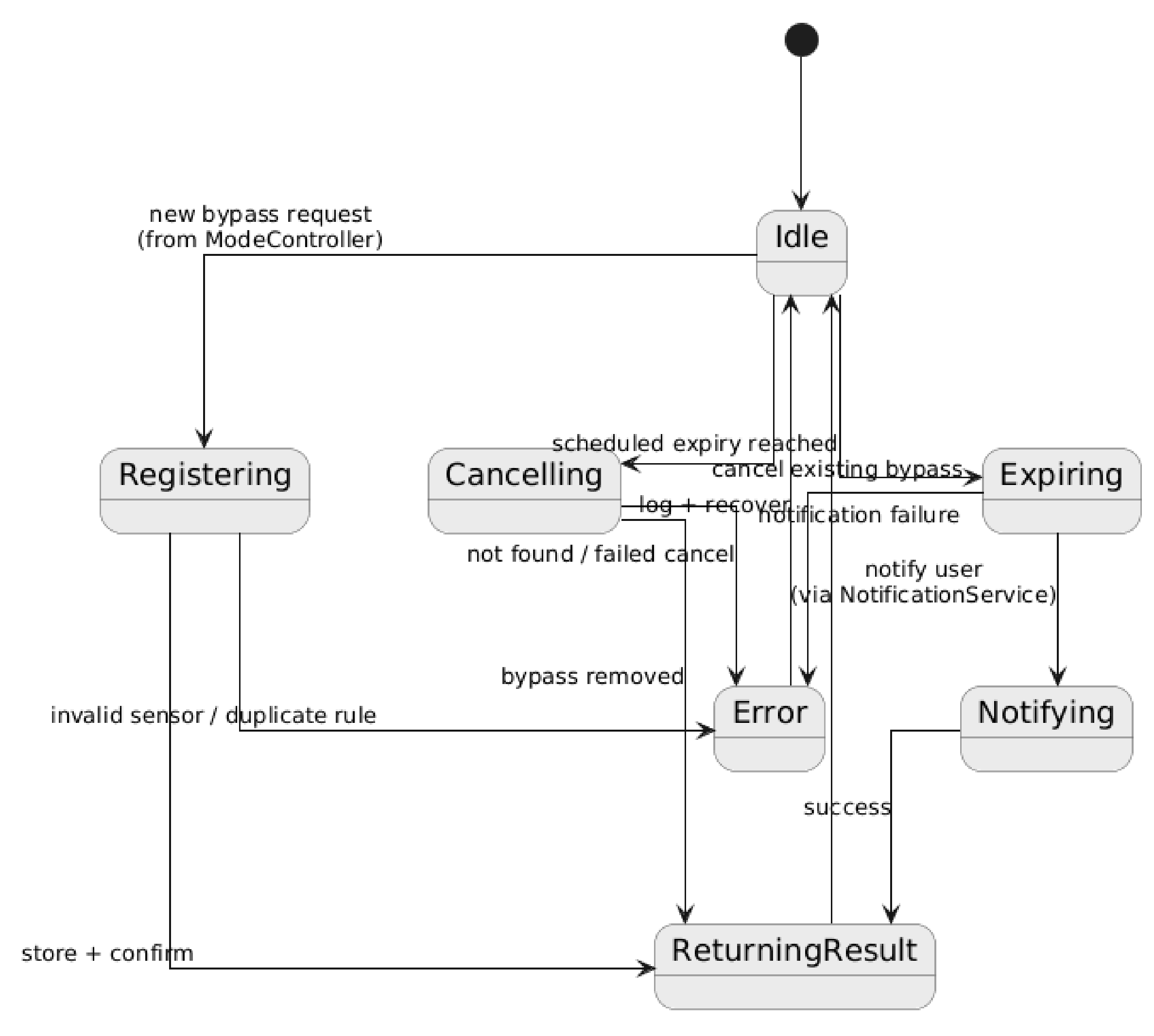
AlarmManager


Idle
alarm trigger
(from SecurityFacade / IncidentManager)
Activating
siren ON / visual indicators ON
(within 3s via SystemClock)
record alarm operation
(via ActivityLog)
activation timeout / hardware fault
Active
connection lost / hardware fault
disarm / cooldown expired
Deactivating
Error
alarm stopped
(via NotificationService)
log fault + notify
Logging

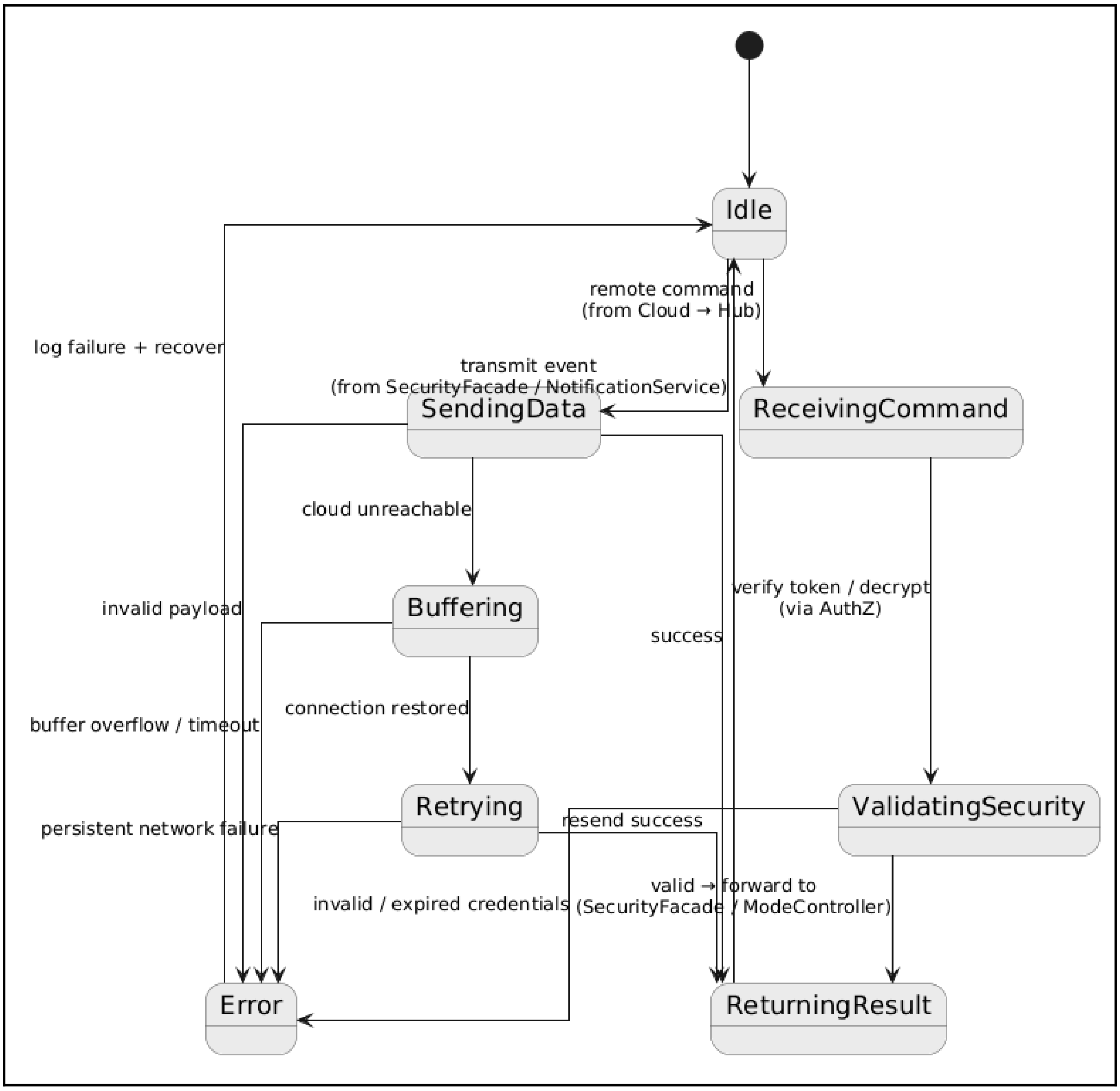
NotificationService


Idle
new alert / status event
(from AlarmManager / SecurityFacade)
PreparingMessage
build push or SMS
(validate user via AuthZ)
Sending
transmission failed / timeout
retry with escalation
record delivery receipt
(via ActivityLog)
invalid recipient / auth failure
Retrying
success
(via CloudGateway)
retries exhausted
Error
log failure + alert admin
Logging

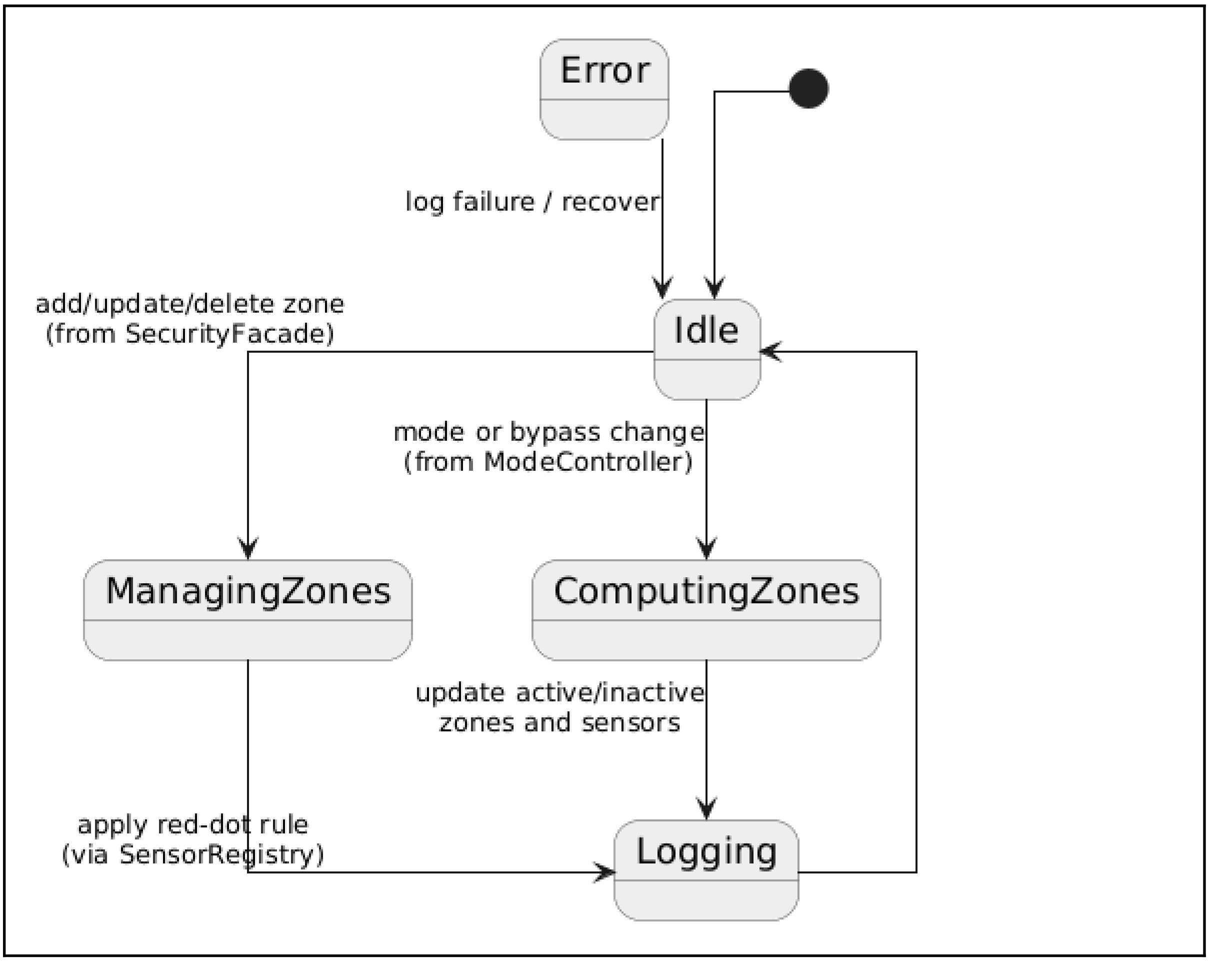
BypassManager



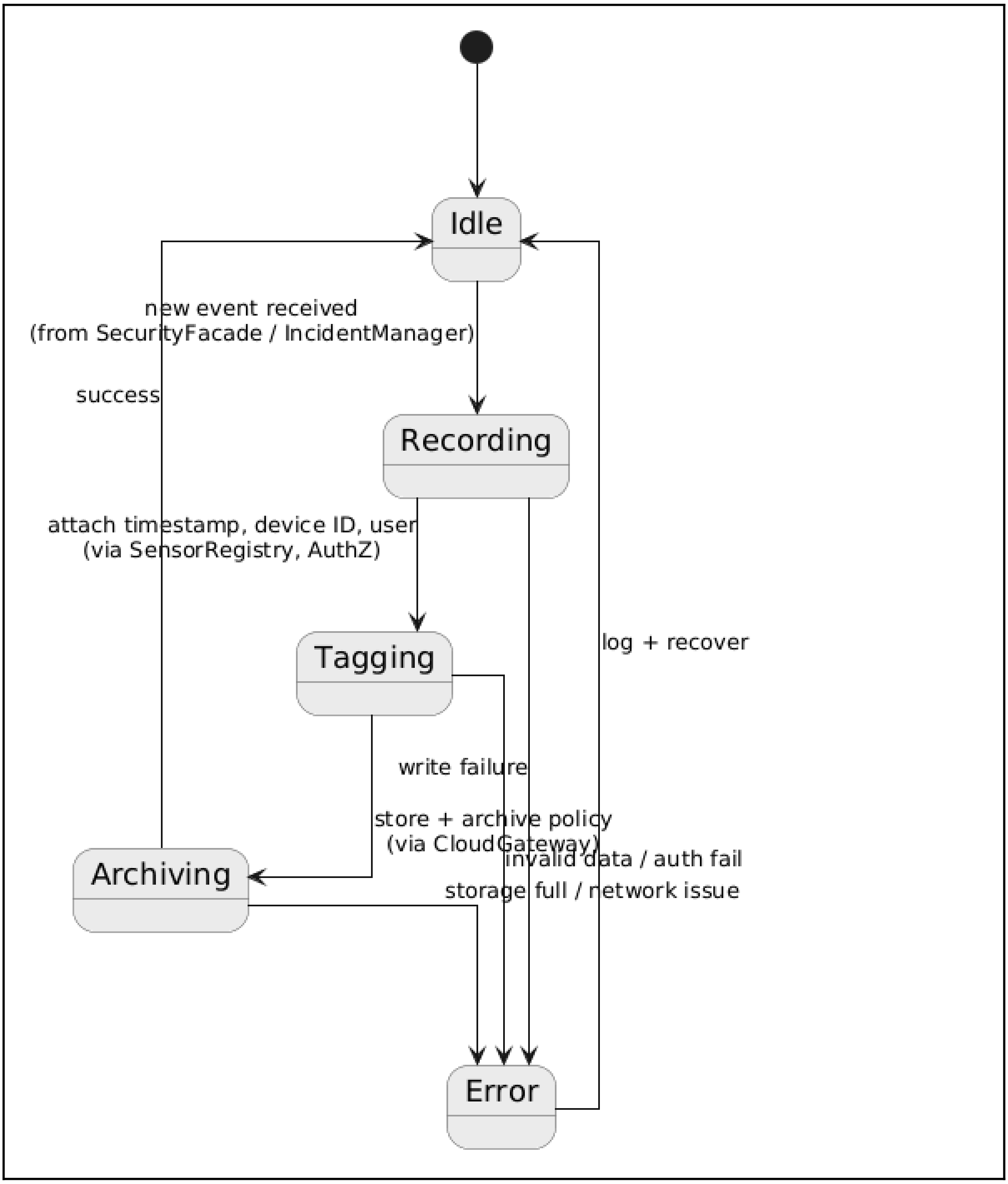
CloudGateway


Idle
remote command (from Cloud → Hub)
log failure + recover
transmit event (from SecurityFacade / NotificationService)
SendingData
ReceivingCommand
cloud unreachable
verify token / decrypt (via AuthZ)
invalid payload
Buffering
success
connection restored
buffer overflow / timeout
Retrying
ValidatingSecurity
resend success
persistent network failure
valid → forward to (SecurityFacade / ModeController)
invalid / expired credentials
Error
ReturningResult

ZoneManager

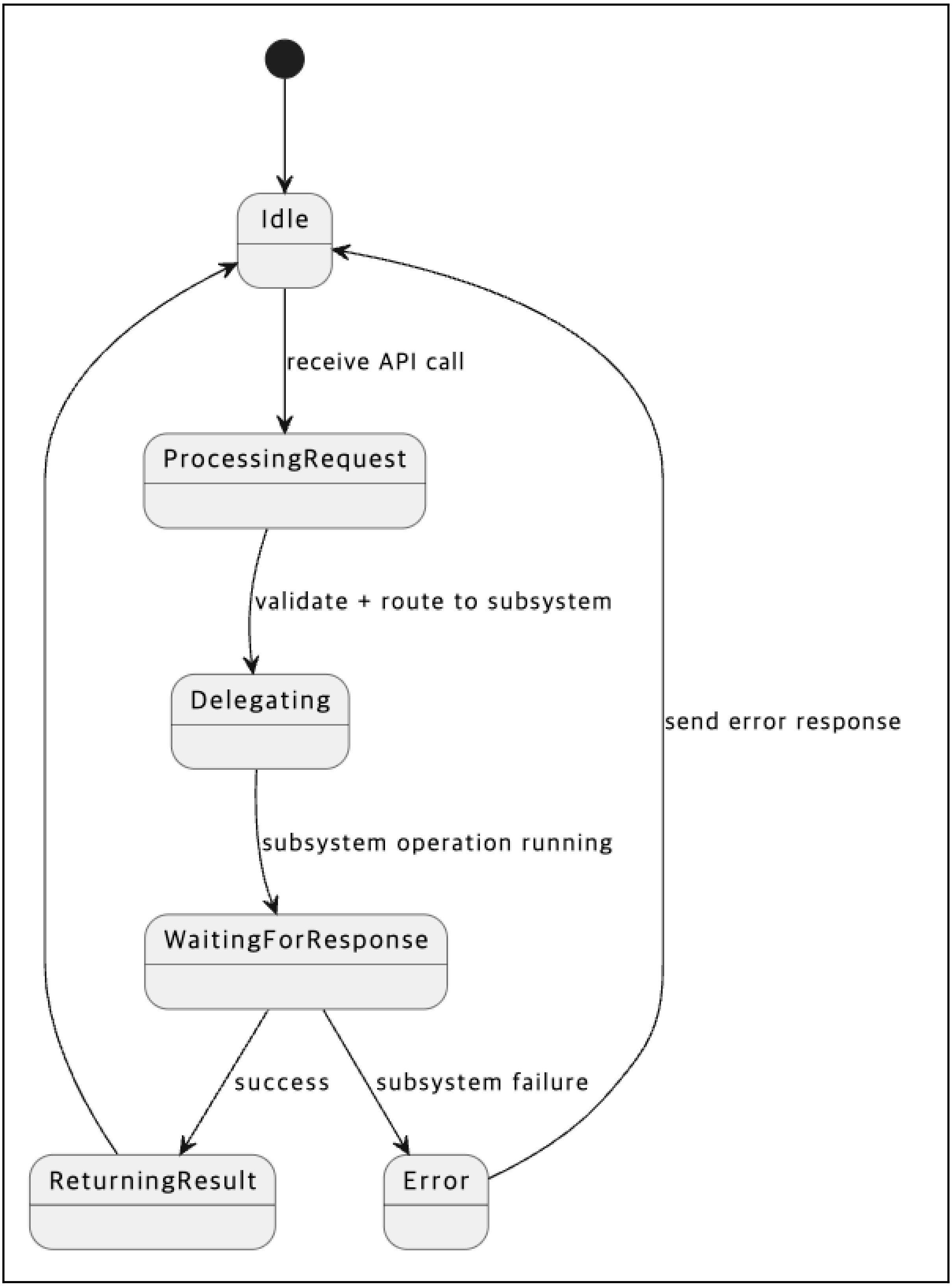


ActivityLog

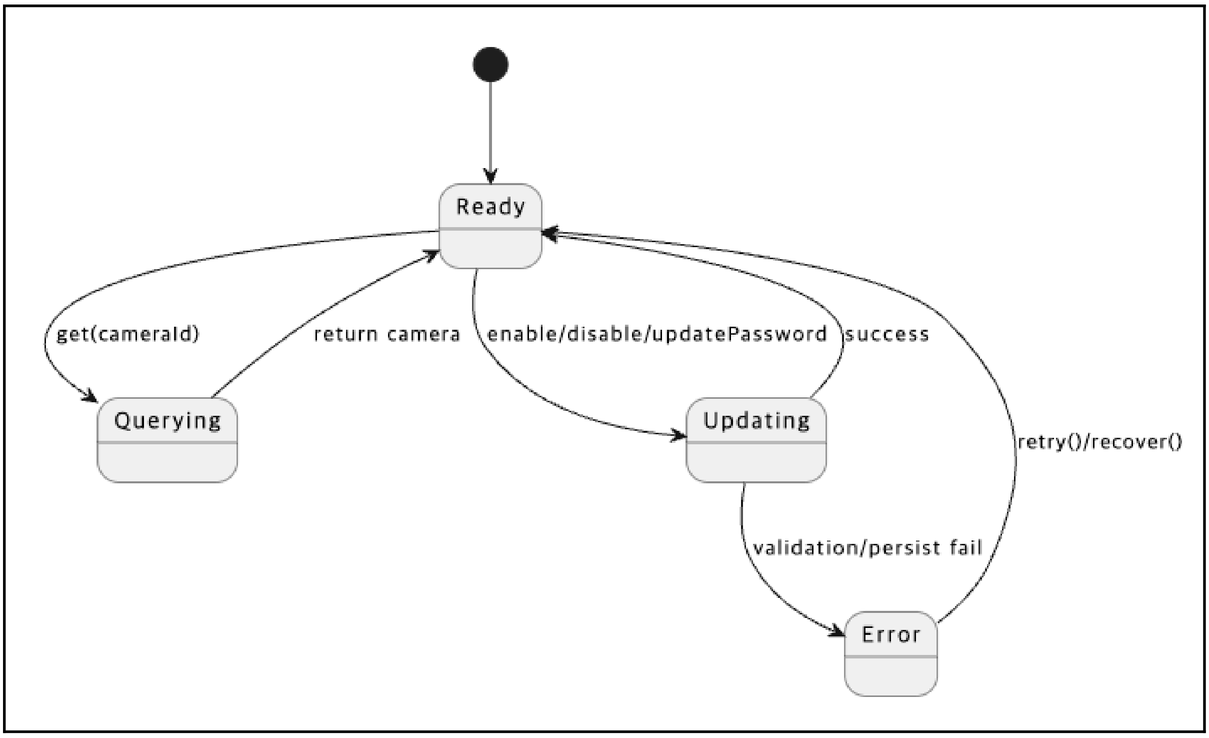


5.2. Live Surveillance

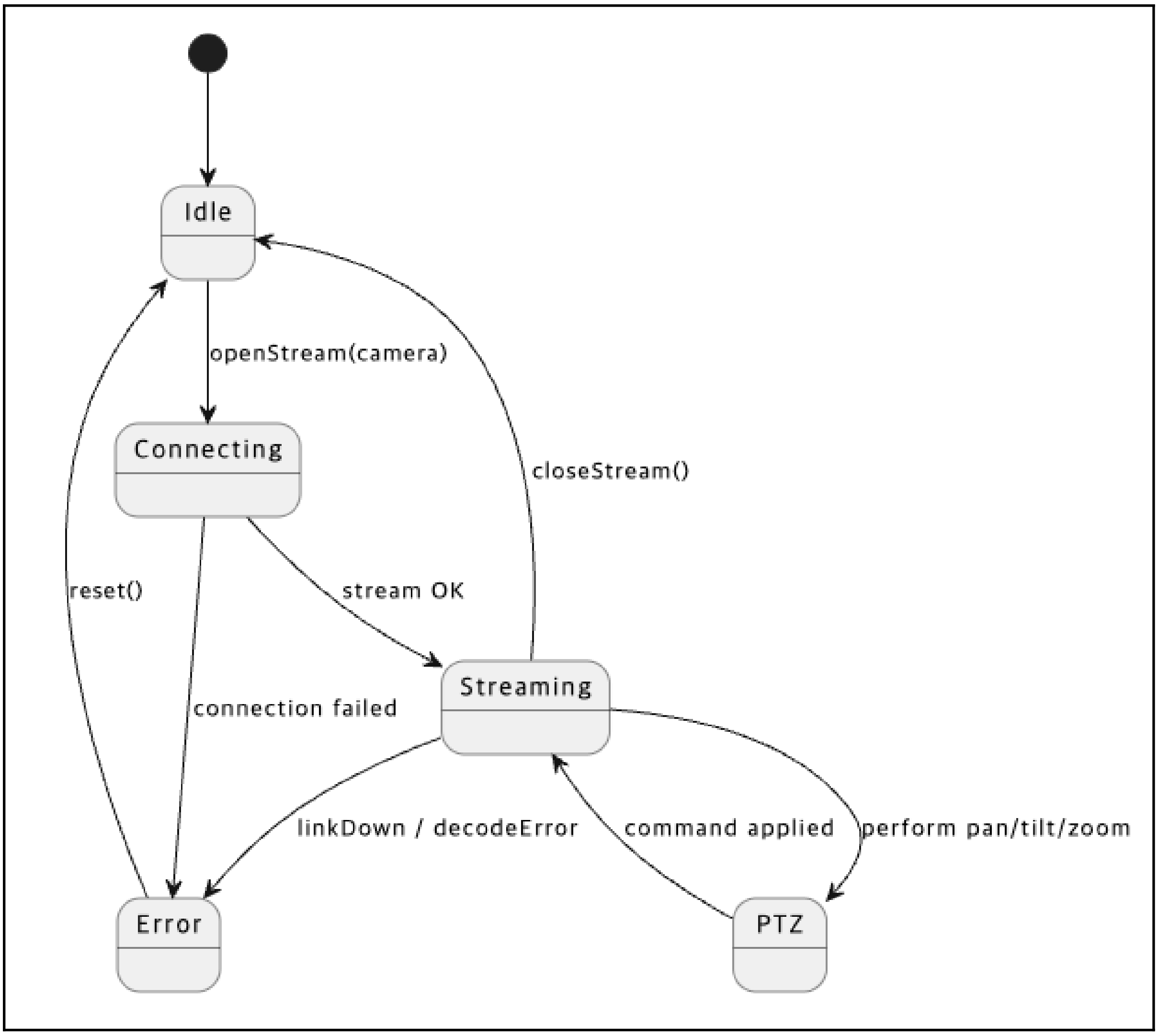
SurveilanceFacade



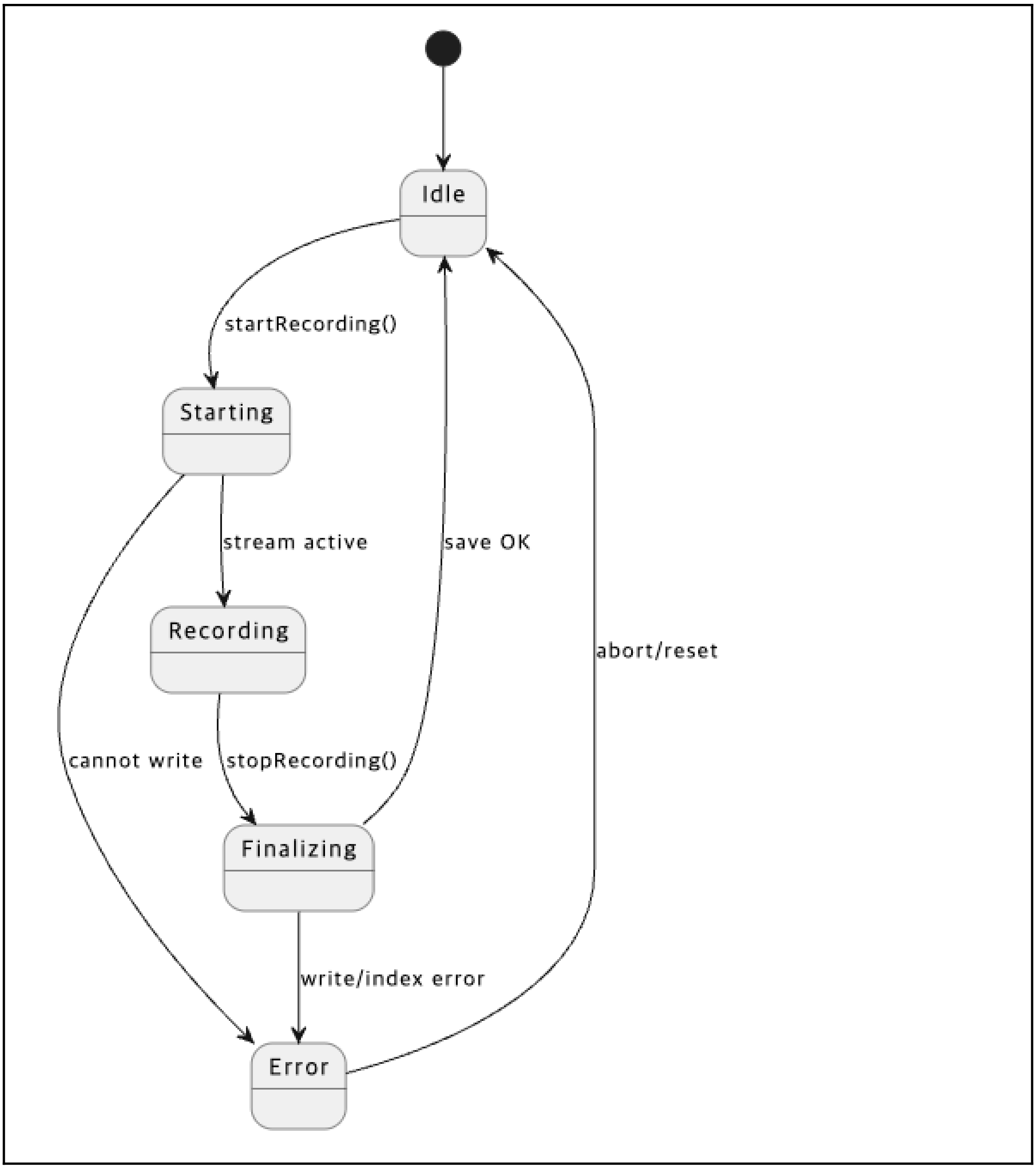
CameraRegistry



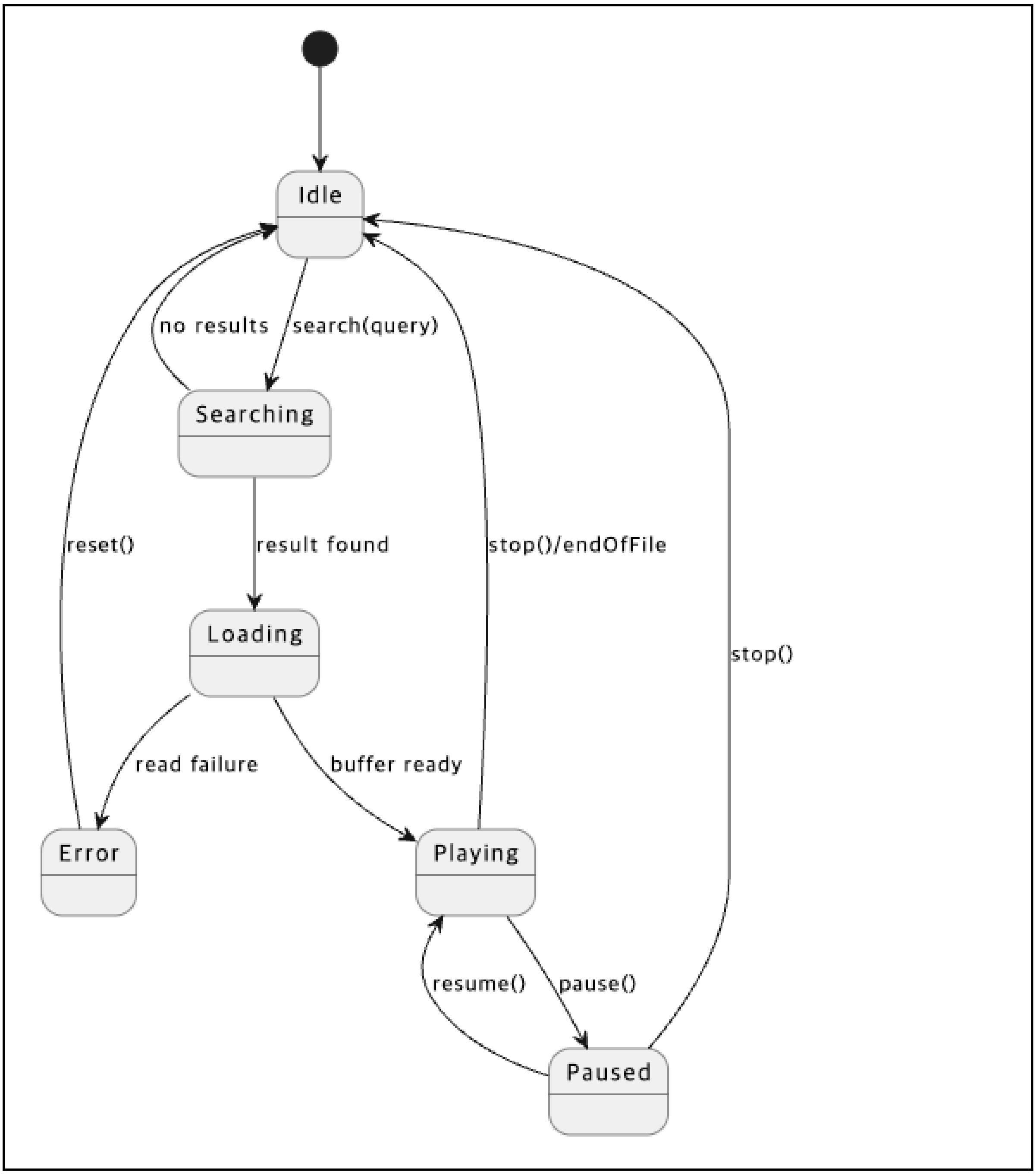
CameraController



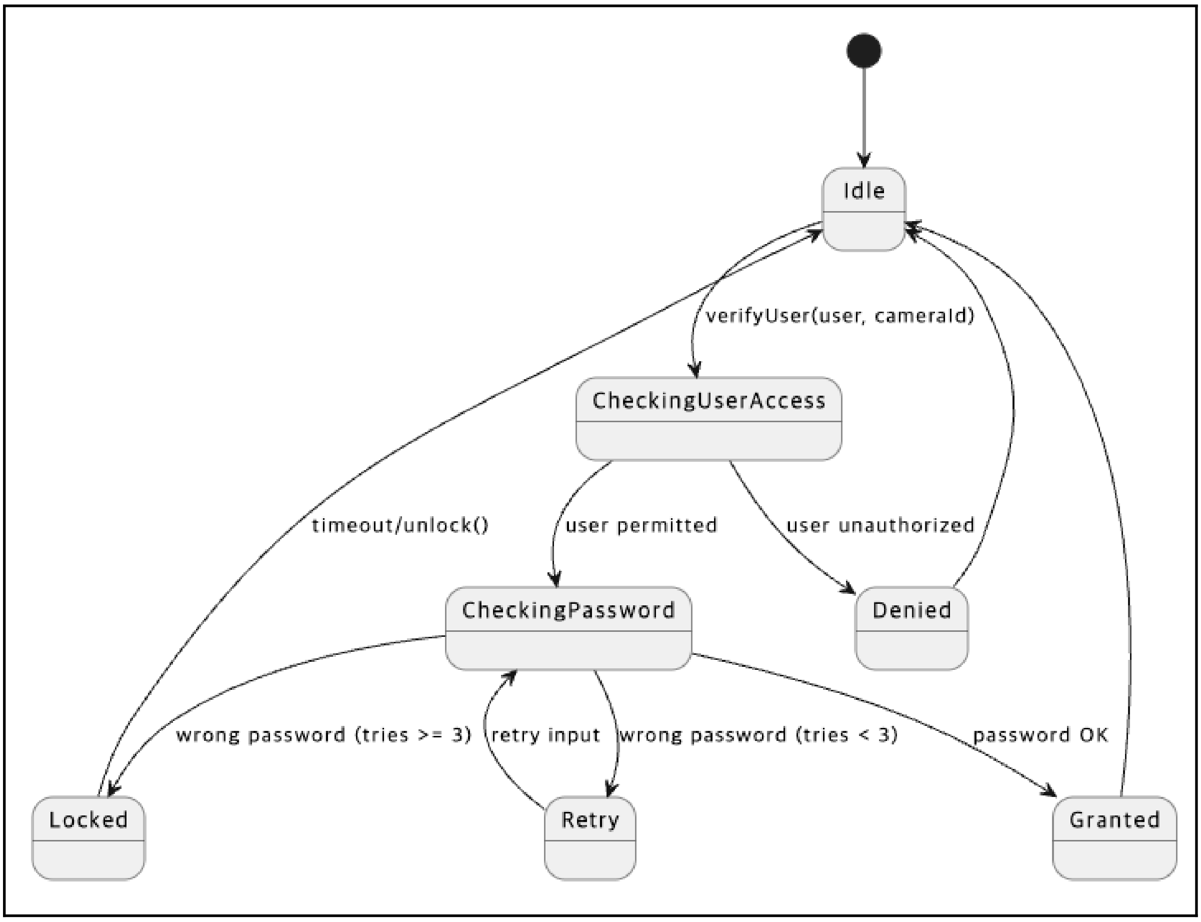
RecordingService



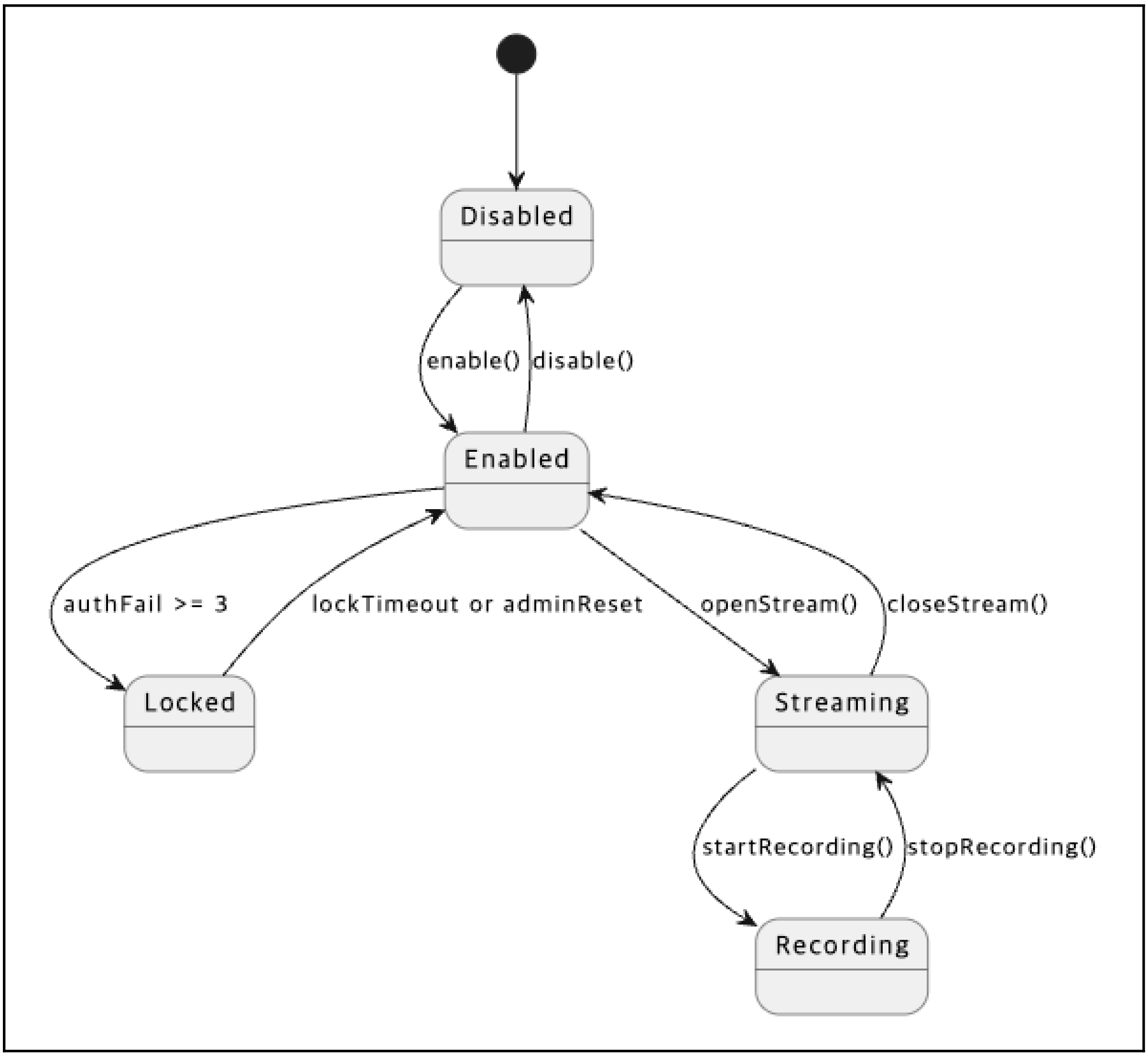
PlaybackService State Diagram


Idle
no results
search(query)
Searching
reset()
result found
stop()/endOfFile
stop()
Loading
read failure
buffer ready
Error
Playing
resume()
pause()
Paused

AuthZ

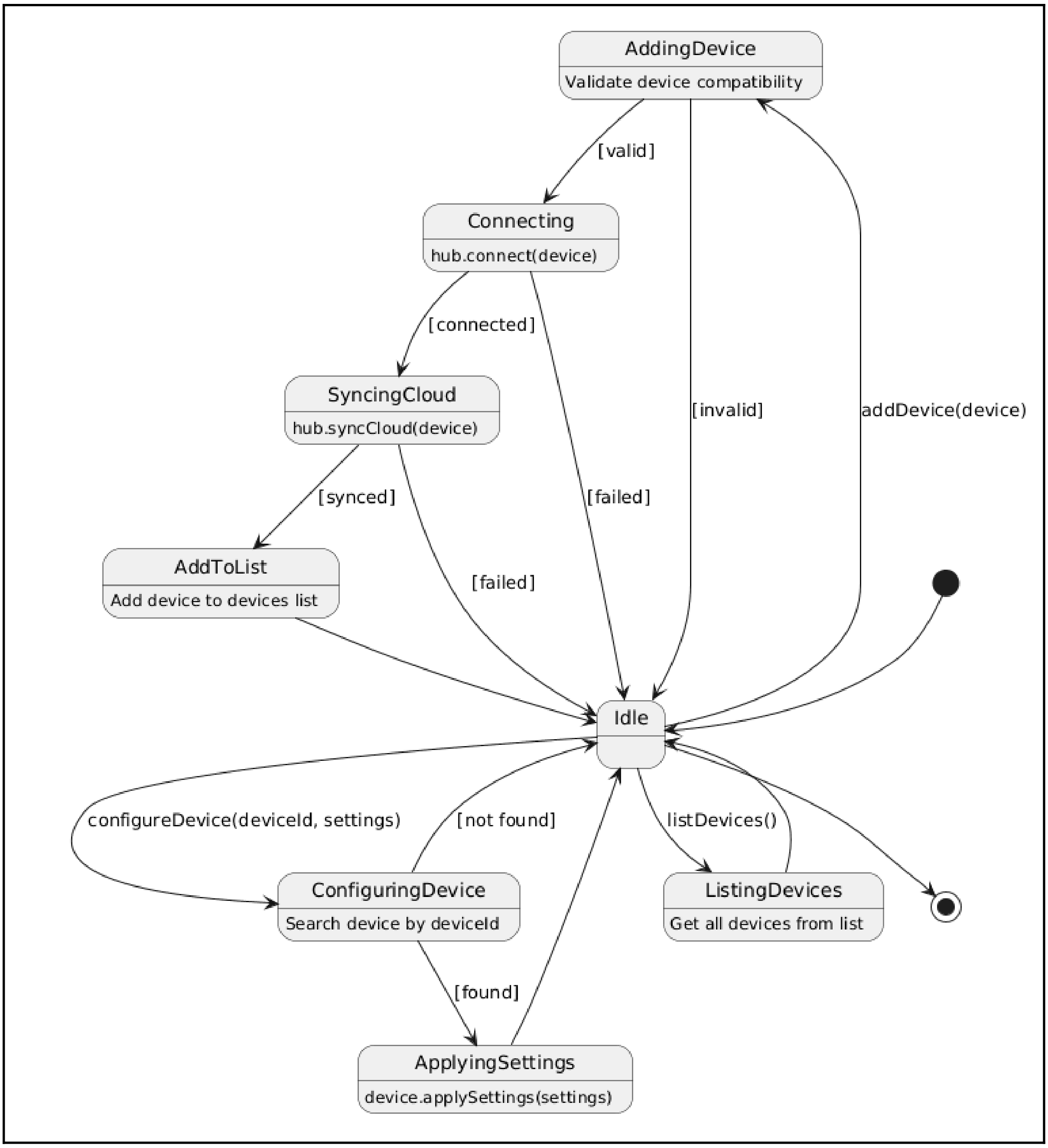


SafeHomeCamera

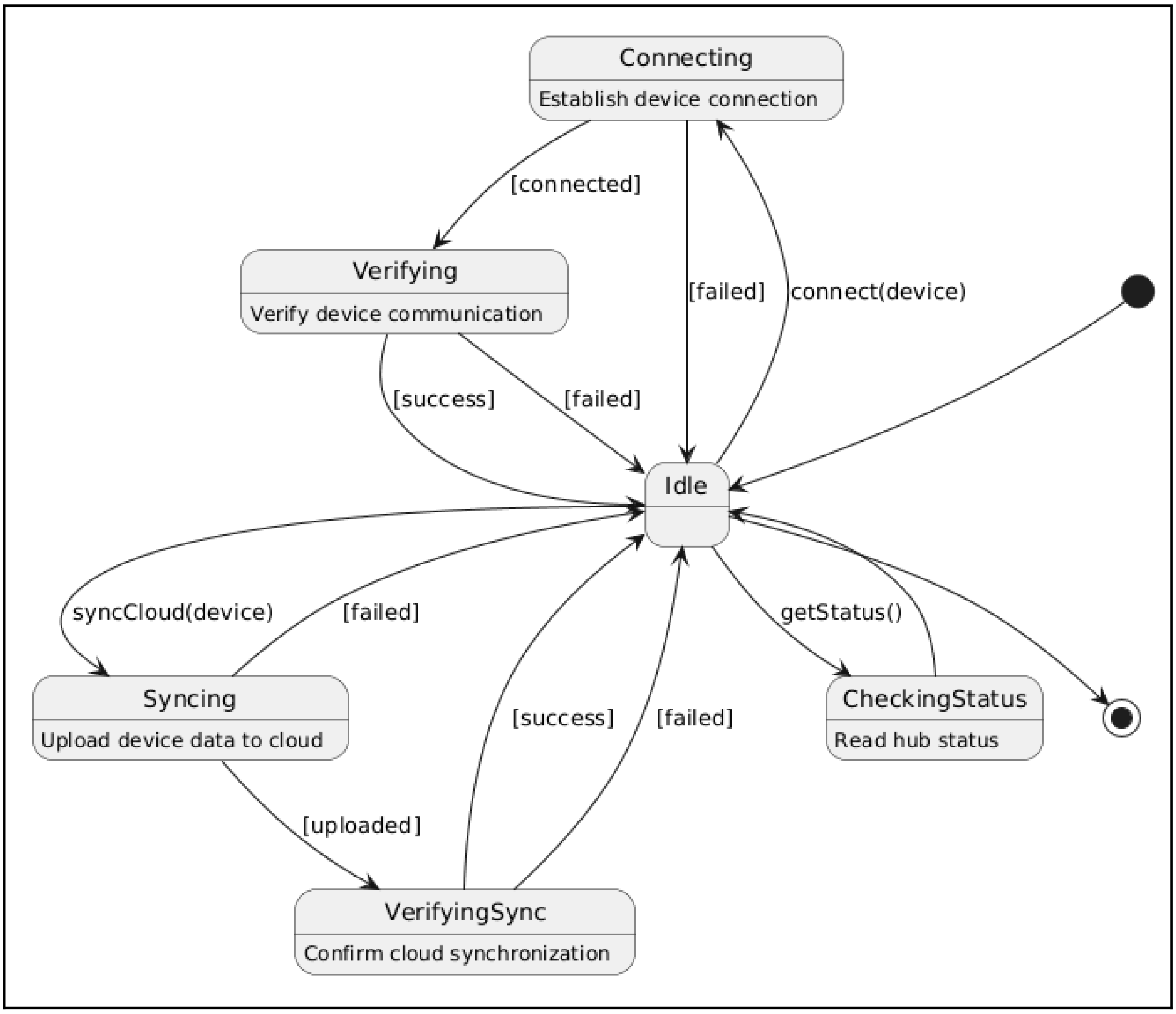


5.3. System and User Management

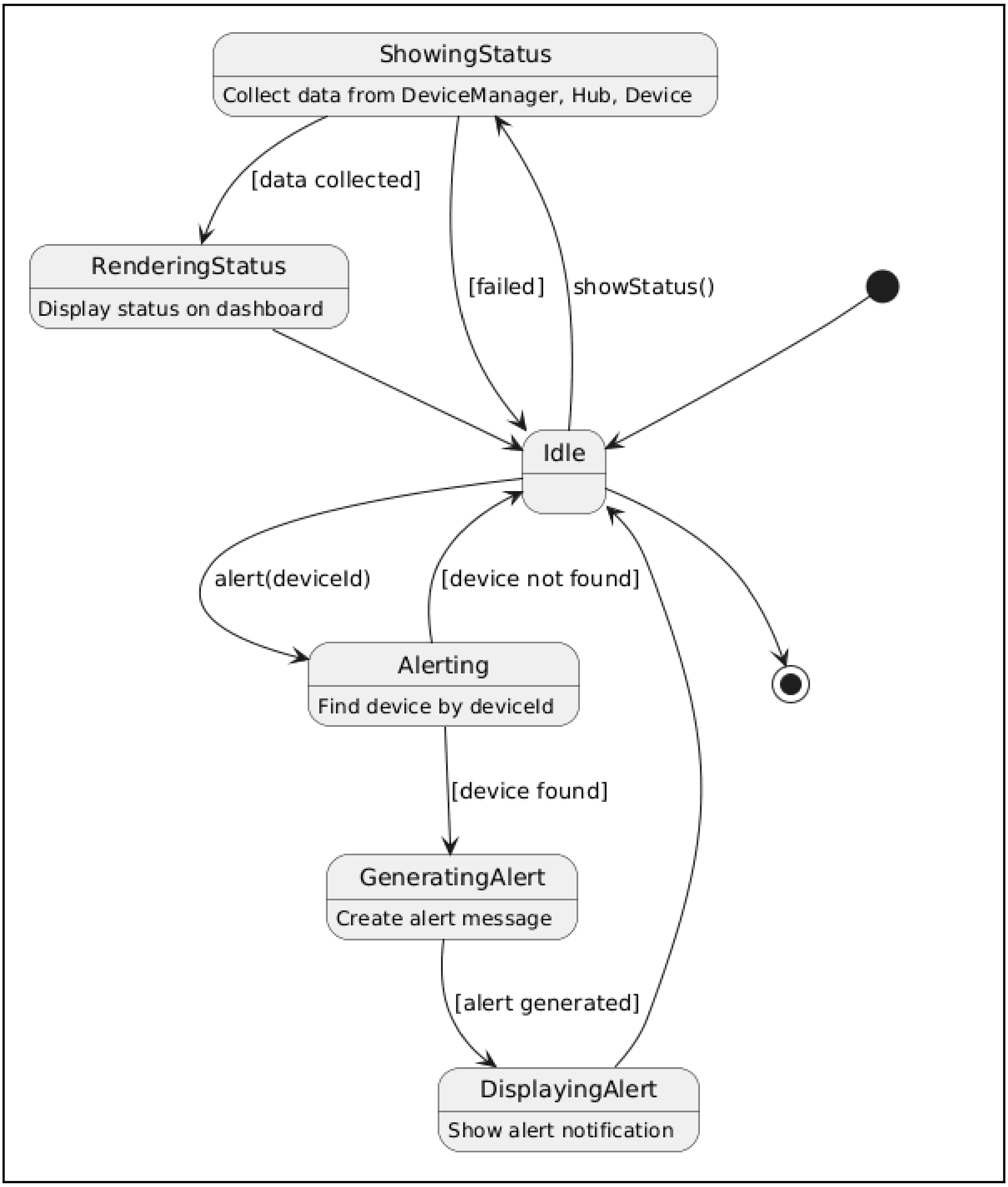
DeviceManager


AddingDevice
Validate device compatibility
[valid]
Connecting
hub.connect(device)
[connected]
SyncingCloud
hub.syncCloud(device)
[invalid]
addDevice(device)
[synced]
[failed]
AddToList
Add device to devices list
[failed]
Idle
configureDevice(deviceId, settings)
[not found]
listDevices()
ConfiguringDevice
Search device by deviceId
ListingDevices
Get all devices from list
[found]
ApplyingSettings
device.applySettings(settings)

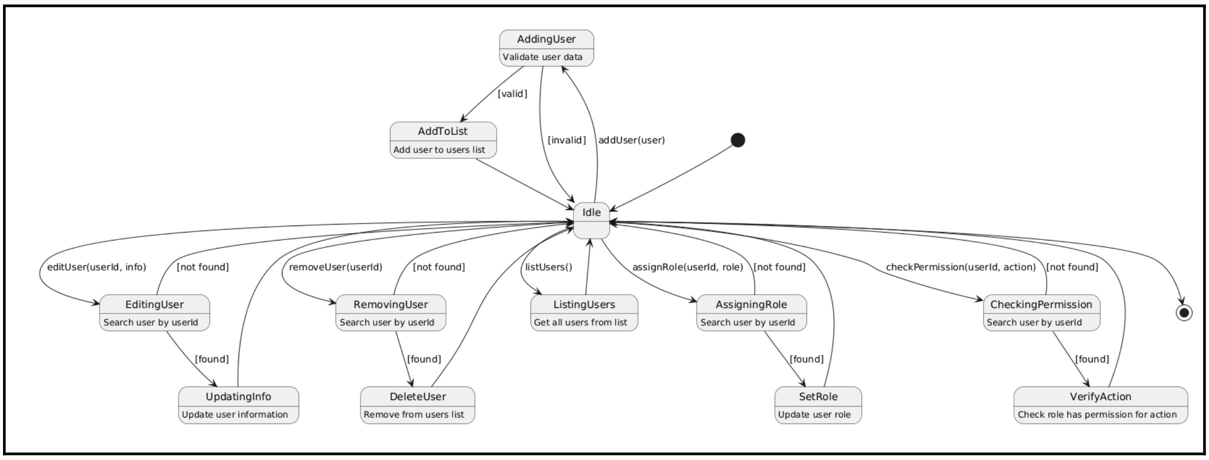
Hub


Connecting
Establish device connection
[connected]
Verifying
Verify device communication
[failed]
connect(device)
[success]
[failed]
Idle
syncCloud(device)
[failed]
getStatus()
Syncing
Upload device data to cloud
[success]
[failed]
CheckingStatus
Read hub status
[uploaded]
VerifyingSync
Confirm cloud synchronization

Dashboard

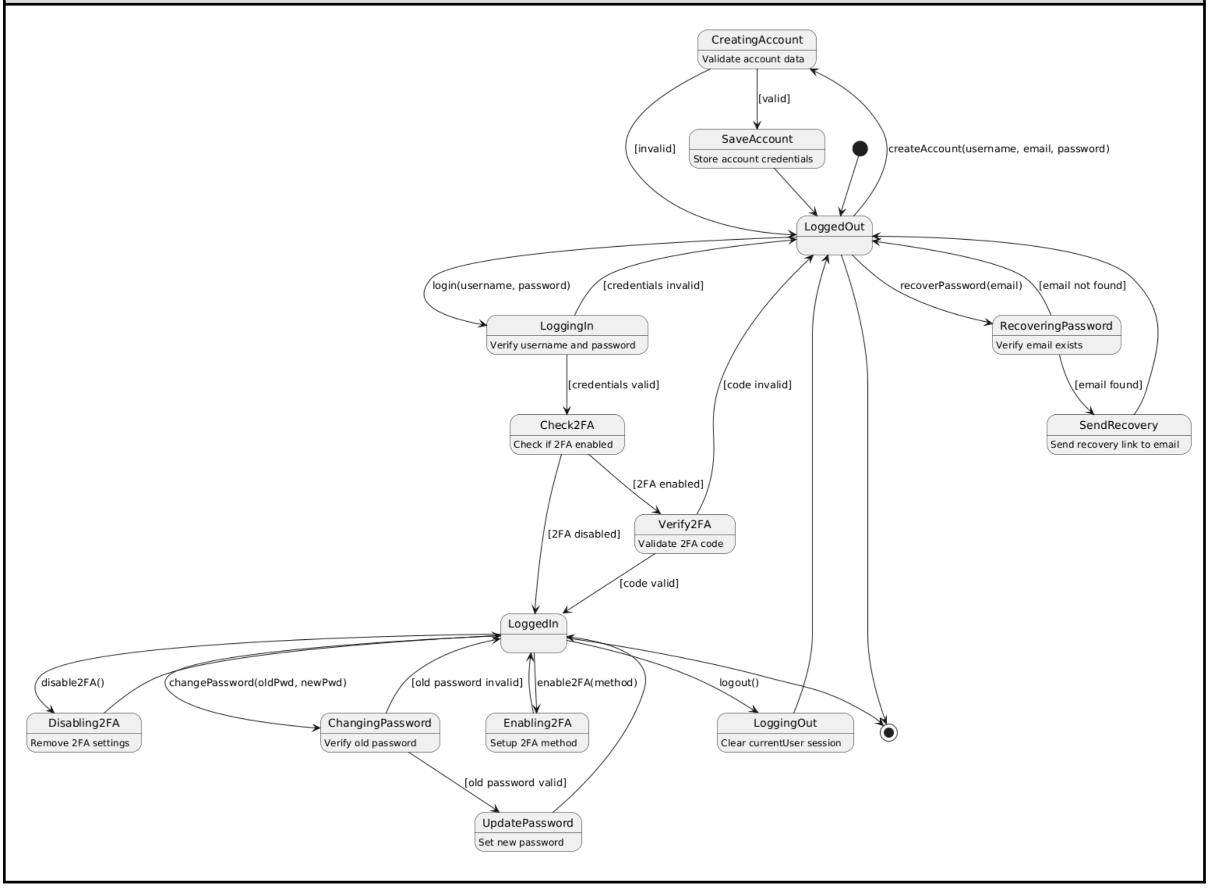


UserManager

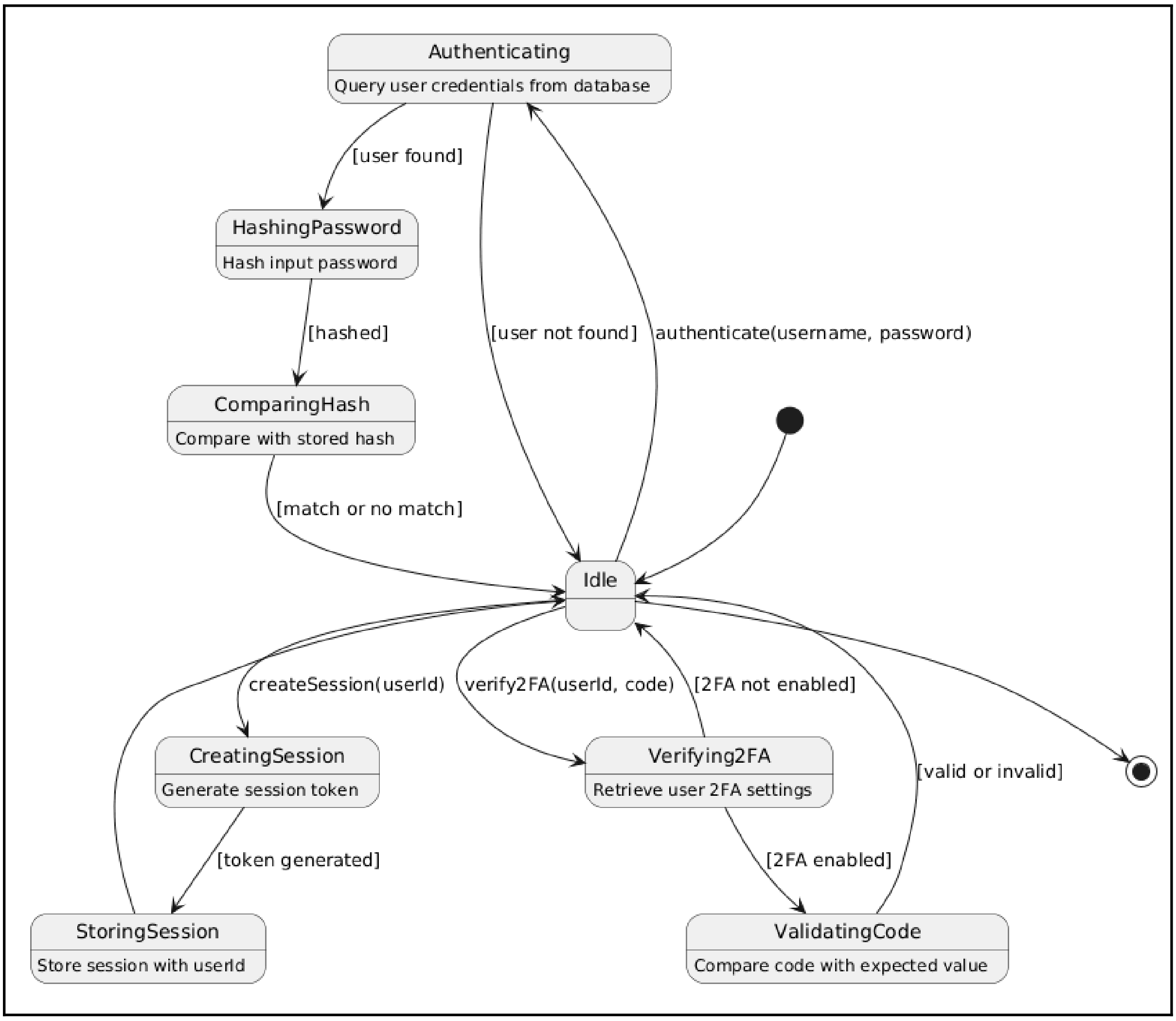


5.4. Remote Access and Account

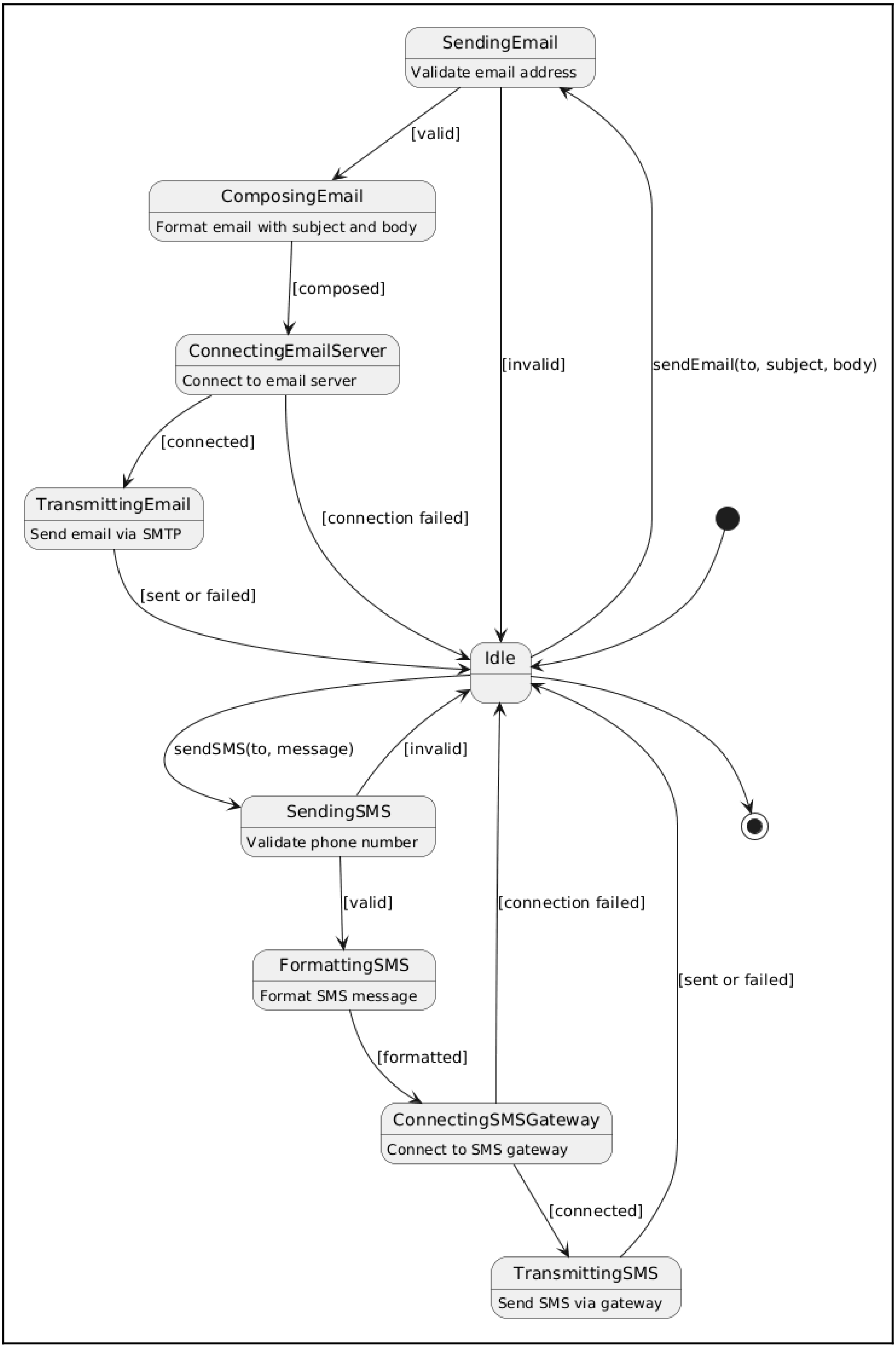
AccountManager



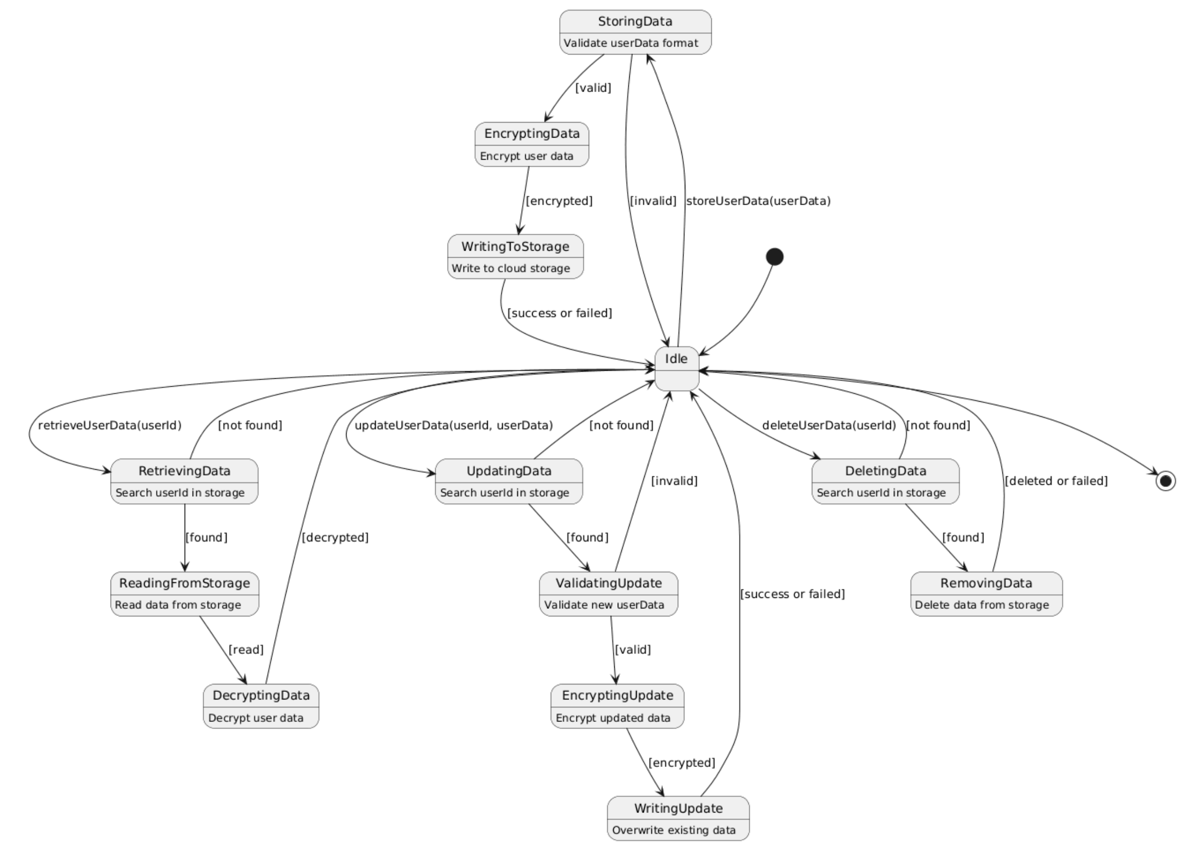
AuthService



NotificationService

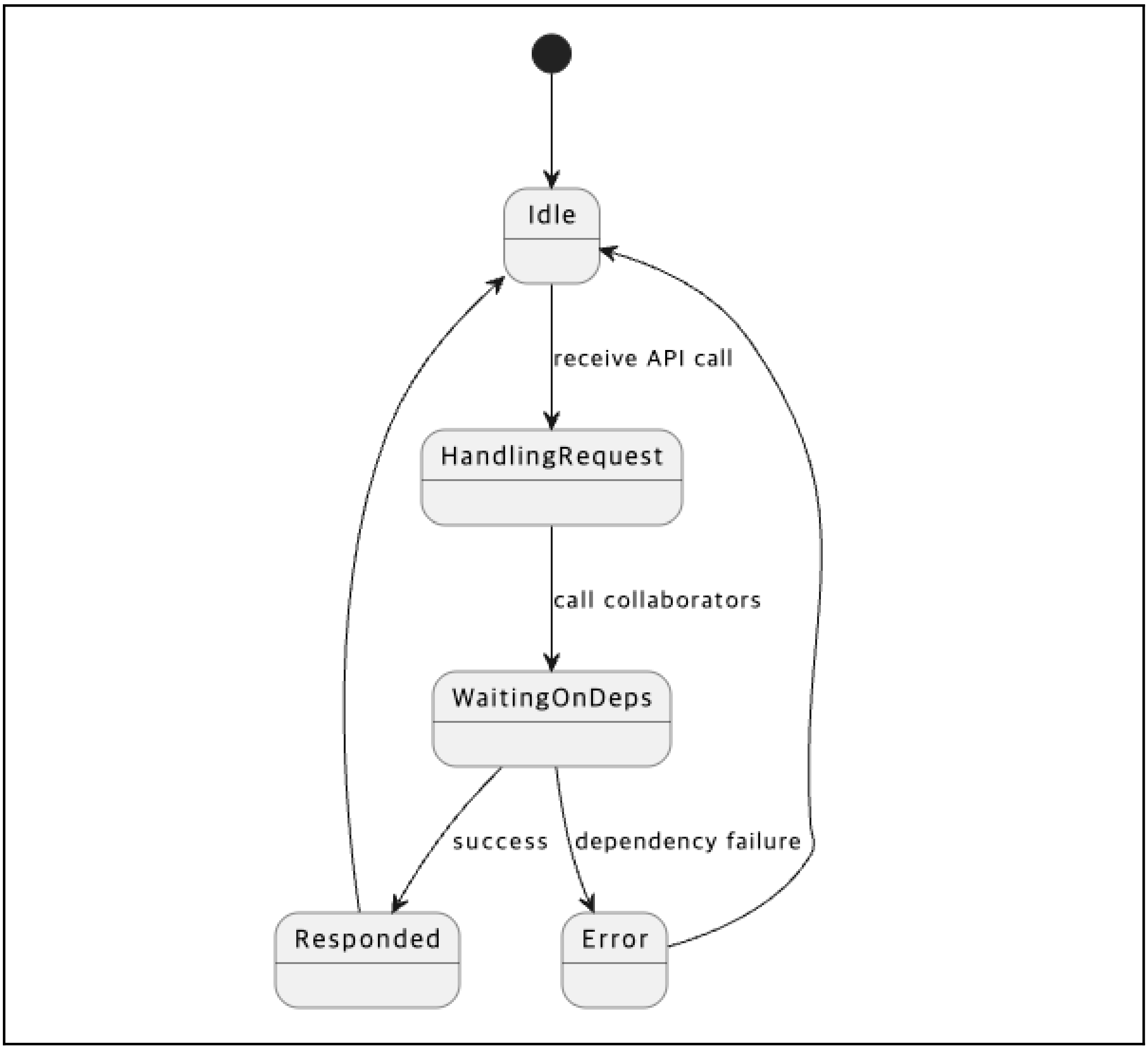

SendingEmail
Validate email address
[valid]
ComposingEmail
Format email with subject and body
[composed]
ConnectingEmailServer
Connect to email server
[invalid]
sendEmail(to, subject, body)
[connected]
TransmittingEmail
Send email via SMTP
[connection failed]
[sent or failed]
Idle
sendSMS(to, message)
[invalid]
SendingSMS
Validate phone number
[valid]
[connection failed]
FormattingSMS
Format SMS message
[sent or failed]
[formatted]
ConnectingSMSGateway
Connect to SMS gateway
[connected]
TransmittingSMS
Send SMS via gateway

CloudServer

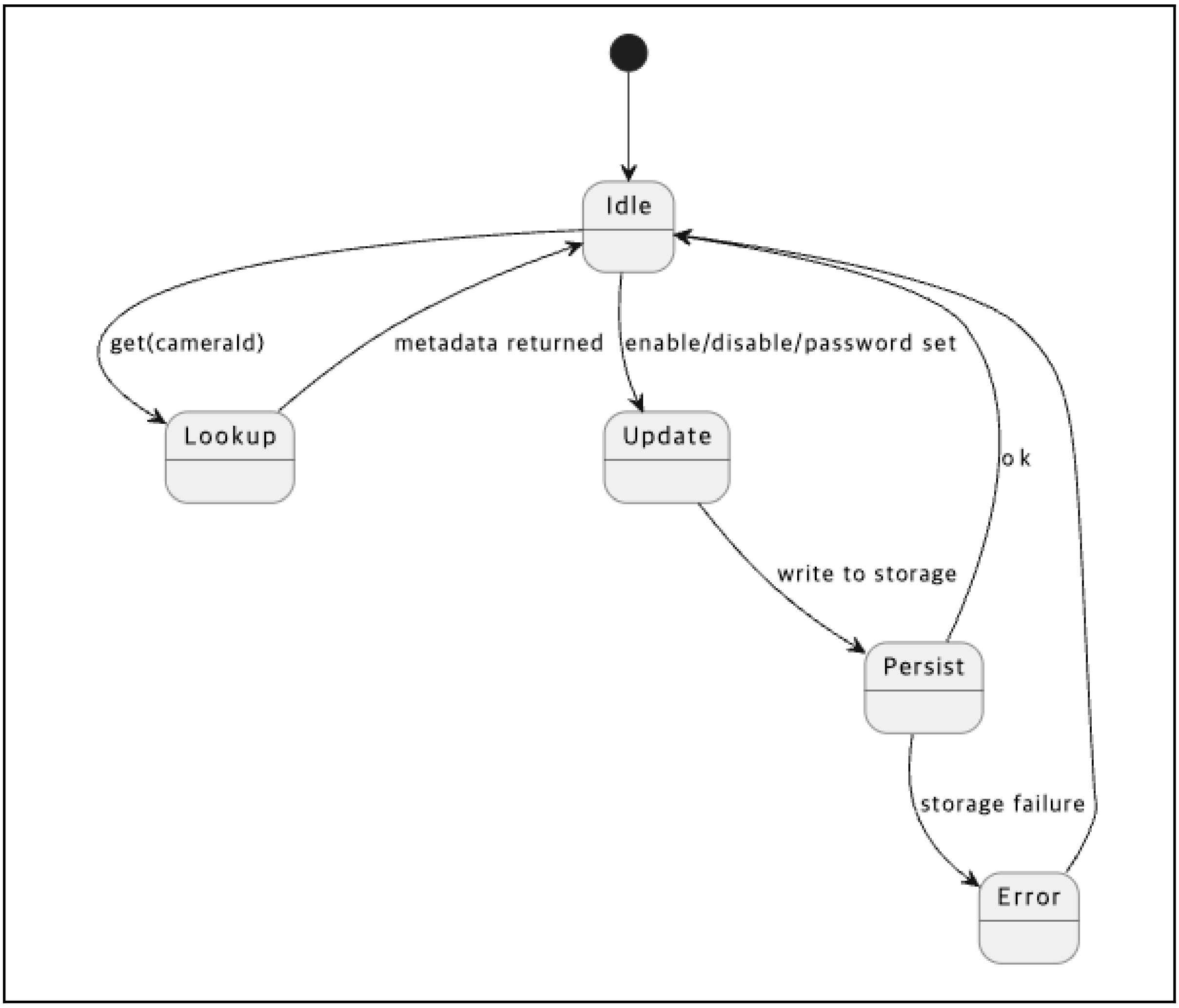


5.5. Indoor Monitoring and Device Control

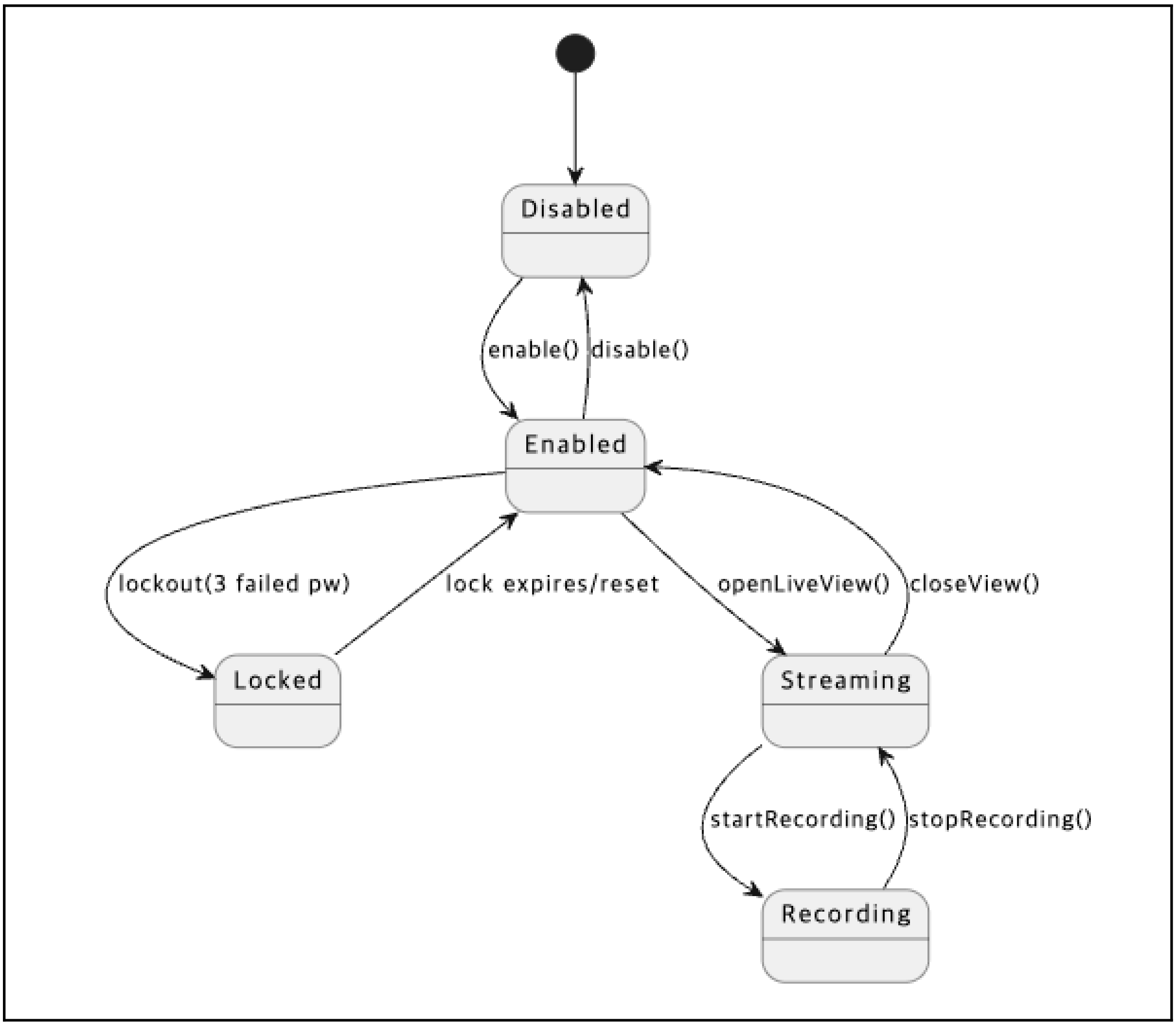
Surveillance Facade



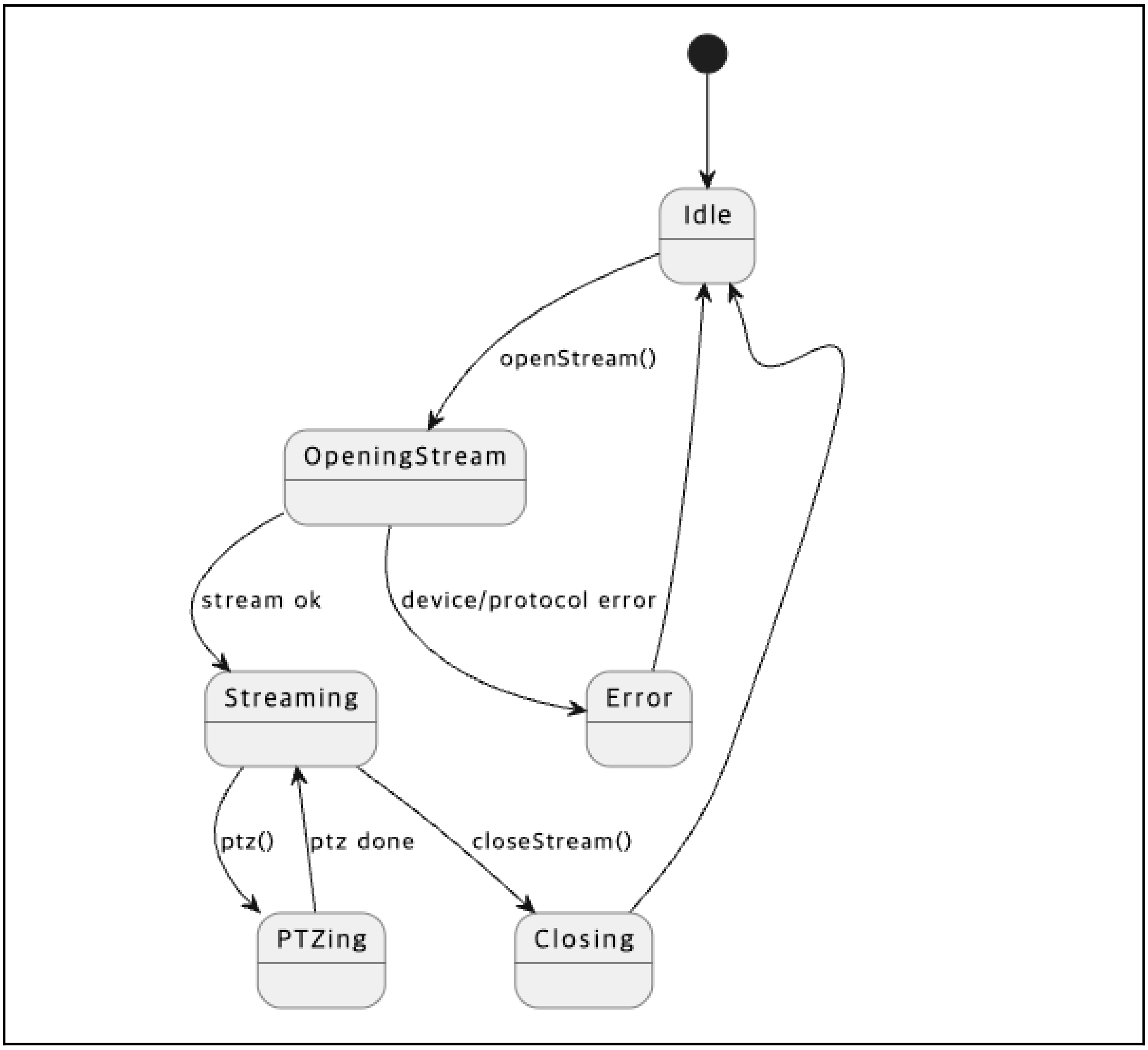
Camera Registry



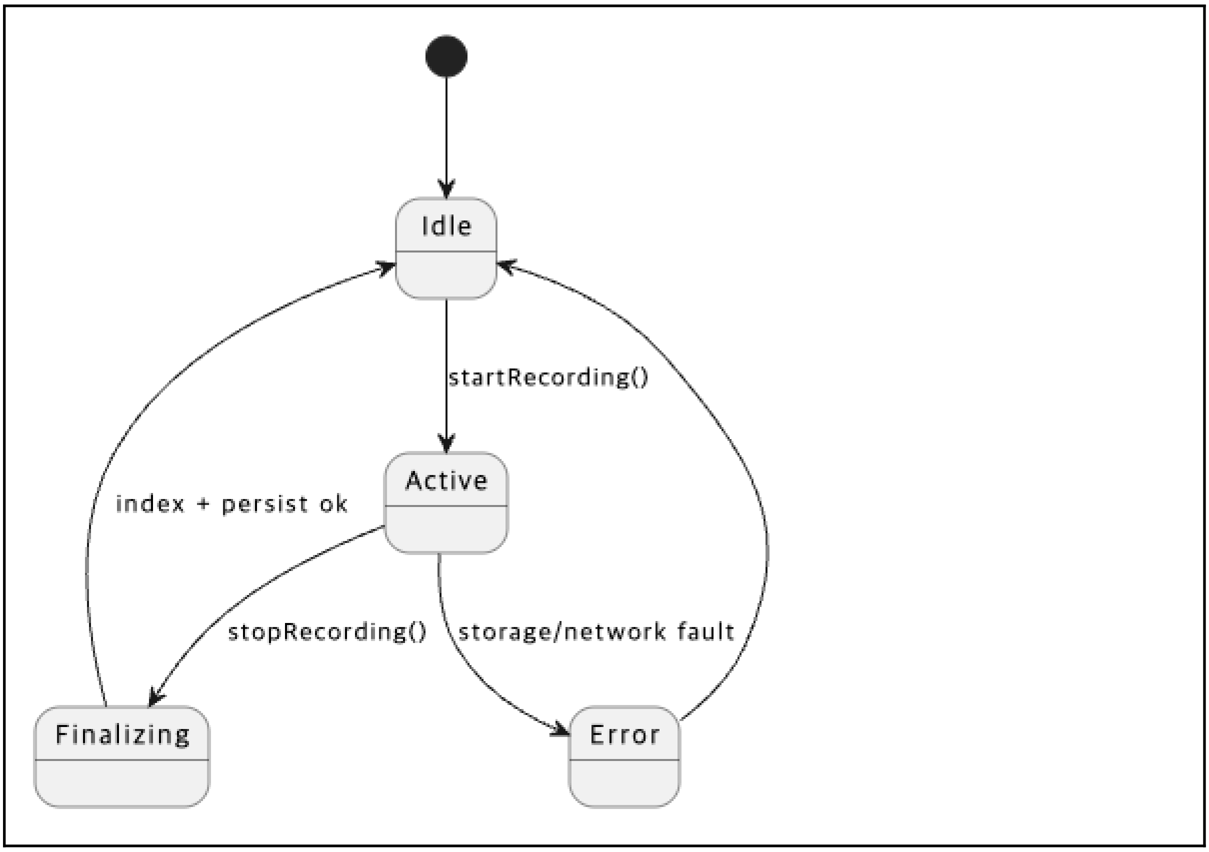
SafeHomeCamera



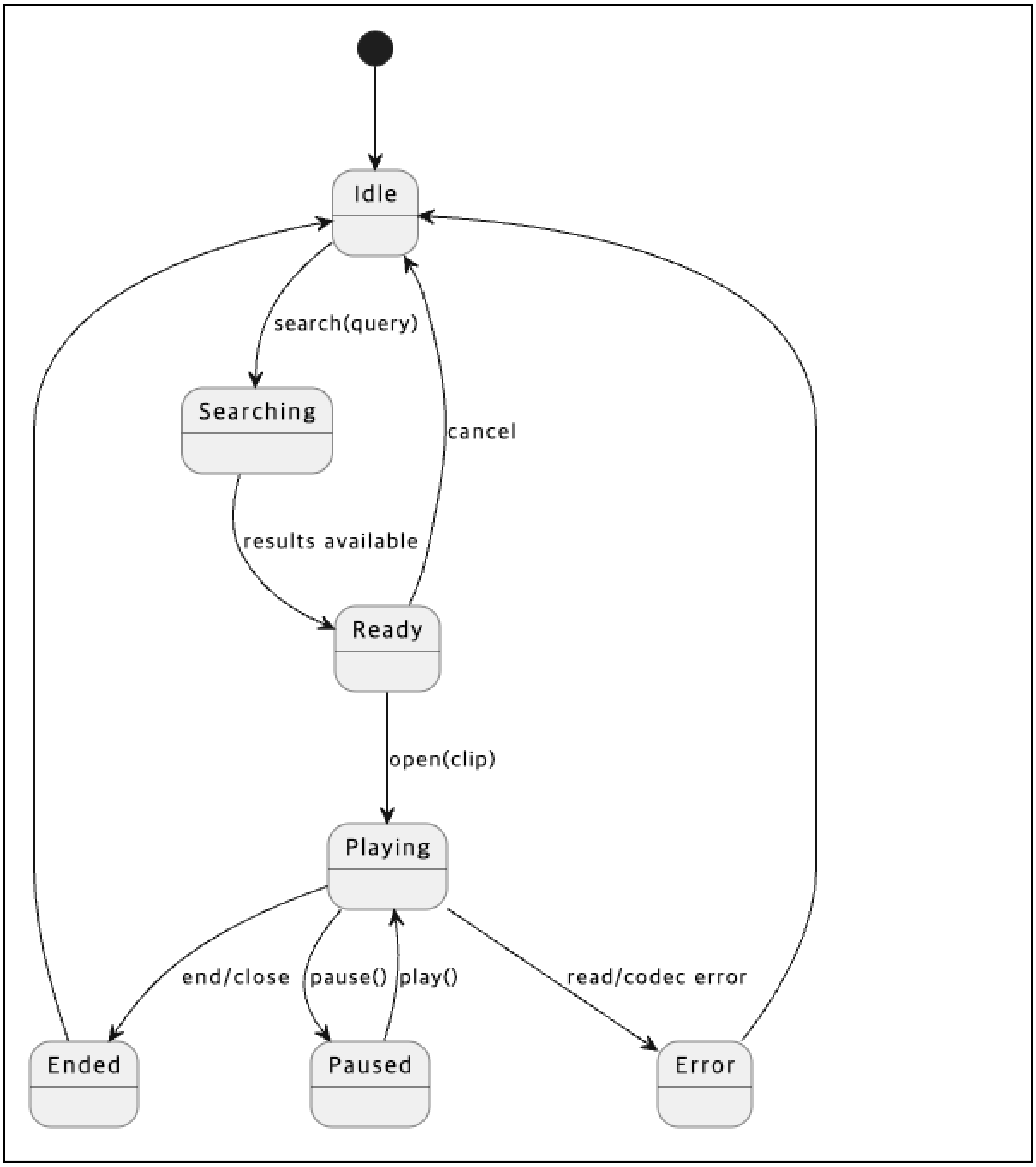
Camera Controller



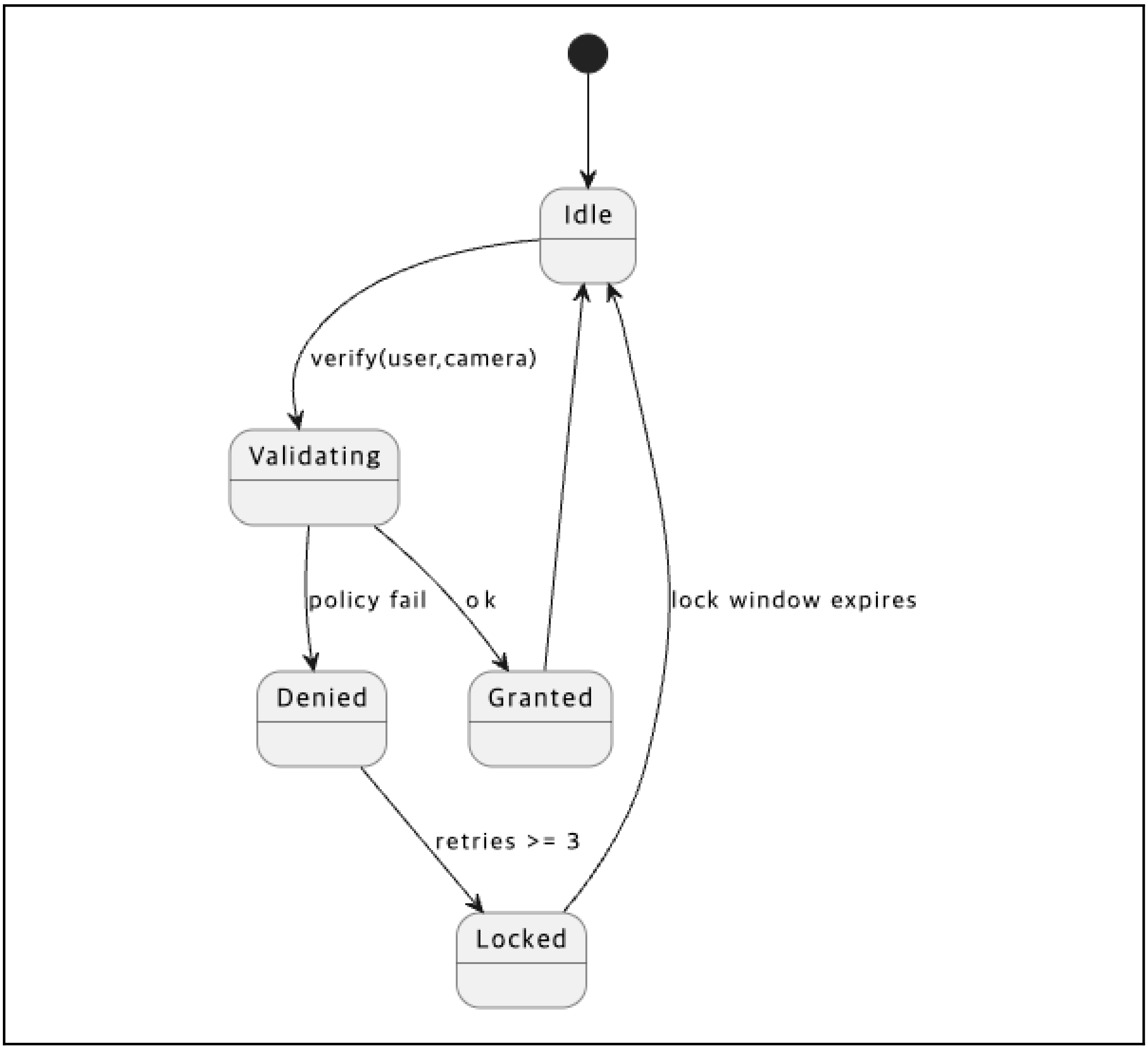
Recording Service



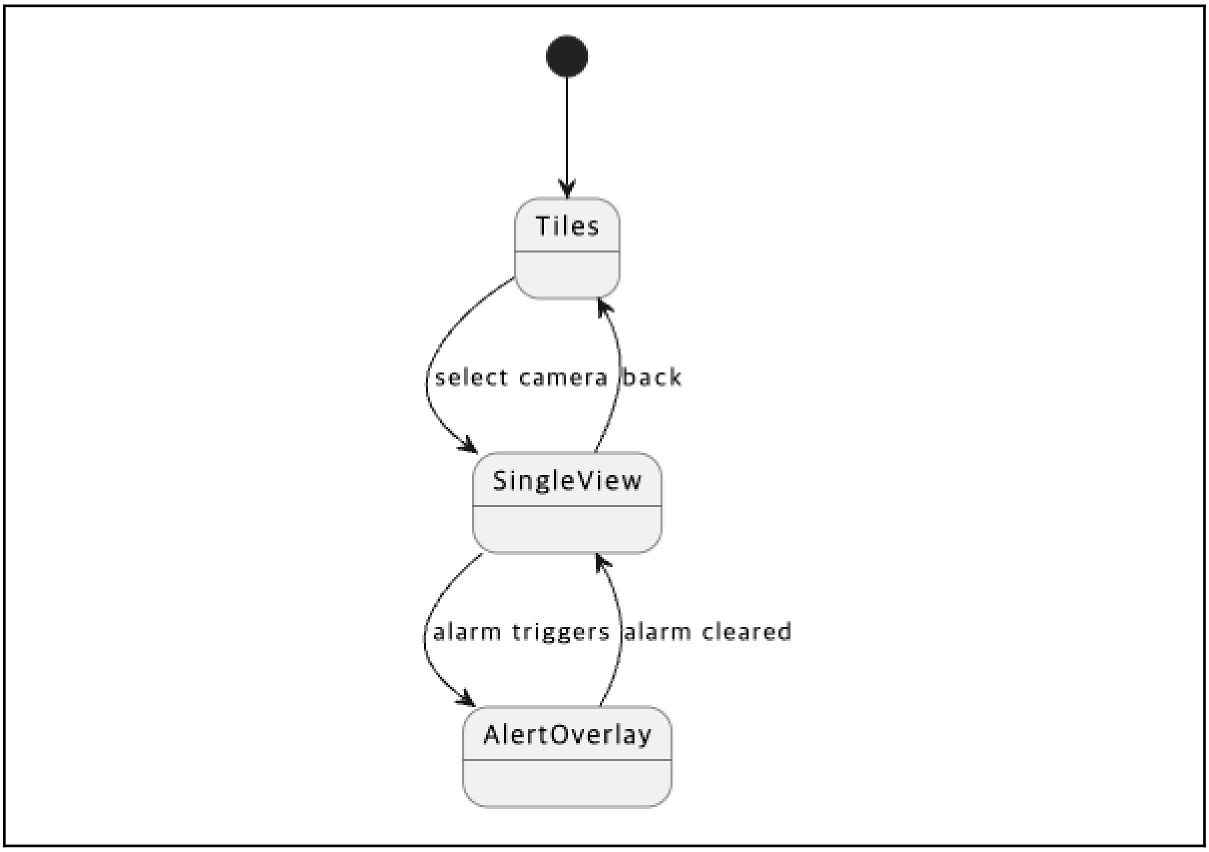
Playback Service


Idle
search(query)
Searching
cancel
results available
Ready
open(clip)
Playing
end/close
pause()
play()
read/codec error
Ended
Paused
Error

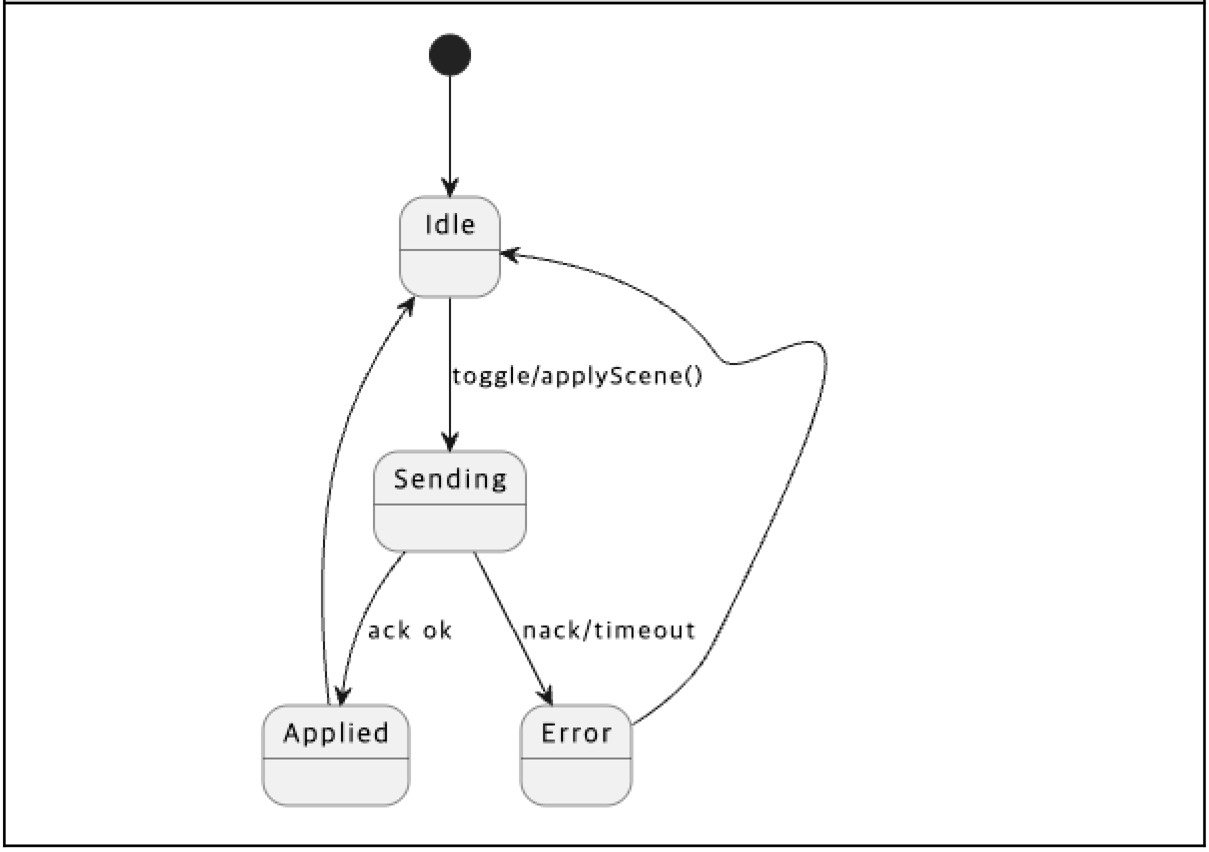
AuthZ



Indoor Monitoring



Indoor Monitoring



6. Design Evaluation

6.1. Architectural Design Metric

a. Design Structure Quality Index (DSQI)

S1 = 44
S2 = 11
S3 = 8
S4 = 10
S5 = 8
S6 = 5
S7 = 40

D1 = Program Structure = 0
D2 = Module Independence = 0.75
D3 = Modules not dependent on prior processing = 0.82
D4 = Database size = 0.2
D5 = Database compartmentalization = 0.5
D6 = Module entrance and exit characteristic = 0.91

W1 = 0.25
W2 = 0.3
W3 = 0.25
W4 = 0.05
W5 = 0.05
W6 = 0.1

DSQI = 0.47

6.2. CK Metrics

Depth of the inheritance tree	2
Maximum Number of Children	7
Average Number of Children	0.30
Maximum Coupling Between Object classes	5
Average Coupling Between Object classes	3.56

6.3. Mood Metric

a. MIF (Method Inheritance Factor)

Number	Class	Md(Ci)	Mi(Ci)	Ma(Ci)
1	SecurityFacade	3	0	3
2	ModeController	2	0	2
3	SensorRegistry	5	0	5
4	IncidentManager	3	0	3
5	AlarmManager	2	0	2
6	NotificationService	3	0	3
7	BypassManager	3	0	3
8	CloudGateway	3	0	3
9	ZoneManager	3	0	3
10	ActivityLog	2	0	2
11	SurvilanceFacade	7	0	7
12	CameraRegistry	5	0	5
13	CameraController	2	0	2
14	RecordingService	2	0	2
15	PlaybackService	2	0	2
16	AuthZ	2	0	2
17	SafeHomeCamera	1	2	3
18	Device	2	0	2
19	DeviceManager	3	0	3
20	Hub	3	0	3
21	Dashboard	2	0	2
22	UserAccount	2	0	2
23	UserManager	6	0	6
24	Account	2	0	2
25	AccountManager	7	0	7
26	AuthService	3	0	3

27	NotificationService	2	0	2
28	CloudServer	4	0	4
29	Page	6	0	6
30	DashboardPage	0	6	6
31	RecordingsPage	1	6	7
32	RecordingDetailPage	1	6	7
33	EmergencyPage	1	6	7
34	DevicesPage	1	6	7
35	DeviceDetailPage	1	6	7
36	PreferencesSettingsPage	1	6	7
37	MobilePage	7	0	7
38	MobileDashboardPage	0	7	7
39	EmergencyTabPage	1	7	8
40	DeviceManagementPage	2	7	9
41	DeviceSettingPage	1	7	8
42	CameraLiveViewPage	2	7	9
43	SettingsPage	2	7	9
44	IndoorMonitoring	3	0	3

MIF = 86 / 202 ≈ 0.426

b. CF (Coupling Factor)

CF = 157 / (44 x 43) ≈ 0.08296

6.4. OO Metric Proposed by Lorenz and Kidd

	# of operations	# of attributes	NOA
SecurityFacade	3	1	
ModeController	2	1	
SensorRegistry	5	1	

IncidentManager	3	1	
AlarmManager	2	1	
NotificationService	3	0	
BypassManager	3	1	
CloudGateway	3	0	
ZoneManager	3	1	
ActivityLog	2	0	
SurvilanceFacade	7	0	
CameraRegistry	5	0	
CameraController	2	0	
RecordingService	2	0	
PlaybackService	2	0	
AuthZ	2	0	
SafeHomeCamera	1	5	
Device	2	5	
DeviceManager	3	2	
Hub	3	2	
Dashboard	2	0	
UserAccount	2	3	
UserManager	6	1	
Account	2	5	
AccountManager	7	1	
AuthService	3	0	
NotificationService	2	0	
CloudServer	4	0	
Page	6	3	
DashboardPage	6	4	

RecordingsPage	7	4	
RecordingDetailPage	7	5	
EmergencyPage	7	4	
DevicesPage	7	4	
DeviceDetailPage	7	5	
PreferencesSettingsPage	7	4	
MobilePage	7	4	
MobileDashboardPage	7	6	
EmergencyTabPage	8	6	
DeviceManagementPage	9	5	
DeviceSettingPage	8	6	
CameraLiveViewPage	9	7	
SettingsPage	9	6	
IndoorMonitoring	3	2	

6.5. General Evaluation of Goal

Refer to 3rd Meeting Log.

The SafeHome system design effectively addresses the four core objectives established in the Software Requirements Specification. The architecture and functional requirements are aligned to ensure the final product delivers comprehensive security, reliability, and an exceptional user experience.

1. Proactive and Comprehensive Security Framework

   The design establishes a proactive, multi-layered security framework through integrated detection and automated response systems. It supports real-time monitoring of both physical intrusions and environmental hazards, combined with automated incident management that includes alarm verification and emergency service dispatch. This approach ensures the system actively prevents and mitigates potential threats rather than merely reacting to them.

2. Seamless and Intuitive User Experience

   Usability is prioritized through a mobile-first interface and intuitive controls. The design features one-touch security modes and a comprehensive system status dashboard that provides clear, context-aware information to users of varying technical proficiency. This focus on accessibility and clarity enables users to confidently manage and monitor their home security.

3. High System Reliability and Data Security

The system ensures operational stability and strong data protection, emphasizing trust and safety. Security is reinforced through measures such as two-factor authentication, role-based access control, and end-to-end encryption of sensitive data, including video streams. These mechanisms safeguard both system integrity and user privacy.

4. Expansion into a Healthy and Smart Living Environment

   Beyond its primary focus on home security, the system architecture is designed for future expansion into a comprehensive smart home platform. It includes provisions for indoor air quality monitoring and smart device control, enabling the system to evolve toward improving quality of life and promoting energy efficiency in future developments.

7. Who Did What

Name	**Responsibility**
Sihun Chae (20190642)	System and User Management, Remote Access and Account
Wooyoung Choi (20190659)	Intelligent Security
Donggeun Kim (20190074)	Live Surveillance, Indoor Monitoring and Device Control

8. Meeting Logs

1st Meeting	
Time	Nov. 5th 2025, 12.00PM-12.30PM
Location	E3-1
Attendees	Sihun Chae, Wooyoung Choi, Donggeun Kim
Goal	Role assignment and schedule coordination
Discussion	**Sihun**: Our goal is to evaluate the Design Metrics, but we need to complete the Architectural Structure and Class Diagram first. **Wooyoung**: Right, the metrics only make sense once the structure is ready. **Donggeun**: Then, when should we finish them? *Sihun*: Let's set the deadline to November 7 and divide the tasks. **Sihun**: I'll take System and User Management, and Remote Access and Account. **Wooyoung**: I'll handle Intelligent Security. **Donggeun**: I'll work on Live Surveillance, Indoor Monitoring, and Device Control. **Sihun**: Great, let's proceed with that plan.

Conclusion	Complete Architectural Structure and Class Diagram by November 7 Task Assignment: - Sihun: System and User Management / Remote Access and Account - Wooyoung: Intelligent Security - Donggeun: Live Surveillance / Indoor Monitoring and Device Control

2nd Meeting	
Time	Nov. 7th 2025, 4.00PM-5.30PM
Location	E3-1
Attendees	Sihun Chae, Wooyoung Choi, Donggeun Kim
Goal	To identify and define the common classes that will be shared across all modules in the Safehome system when designing the class diagram. These shared components will ensure modularity, maintainability, and consistency across different subsystems such as Security, Surveillance, and User Management.
Discussion	**Sihun**: Today's goal is to decide which classes should be shared across all modules when we build the class diagram. We need to make sure that common functions, like user management and notifications, are not duplicated in each subsystem. **Wooyoung**: The shared classes should mainly cover authentication, user management, device control, and system coordination. **Donggeun**: For user-related functionality, AuthZ, UserAccount, and AccountManager will definitely be common since all modules require authentication and account access. **Sihun**: Agreed. We should also include AuthService to handle session management and authorization checks. That ensures consistency across both local and remote access. **Wooyoung**: On the system control side, ModeController, NotificationService, and CloudGateway should be defined as shared classes. They'll manage system modes, alert distribution, and communication with the cloud server. **Donggeun**: Yes, especially for events like intrusions. When IncidentManager detects something, it will use NotificationService to alert users and send reports through CloudGateway. **Sihun**: For device management, we'll need a base Device class and a central DeviceManager that handles registration and monitoring of all connected devices.

	Donggeun: Then, module-specific devices like SafeHomeCamera or sensors can inherit from the base Device class. We should also keep a SensorRegistry for referencing all sensors in the system. **Wooyoung**: Don't forget ActivityLog. All system actions, alerts, and changes should be recorded centrally for traceability and auditing. Sihun: That makes sense. We'll include it under the shared system management components. **Wooyoung**: So to summarize, shared classes will handle user access, communication, and device coordination, while specialized modules can extend them as needed. **Sihun**: Correct. We'll apply these shared classes to the class diagram to maintain architectural consistency across all subsystems.
Conclusion	User & Access Management: AuthZ, AuthService, UserAccount, AccountManager System Control & Communication: ModeController, NotificationService, CloudGateway, IncidentManager, ActivityLog Device Management: Device, DeviceManager, SensorRegistry, SafeHomeCamera These common classes will form the core architecture, enabling consistent operation and integration among all Safehome modules.

3rd Meeting	
Time	Nov. 12th 2025, 3.00PM-4.30PM
Location	E3-1
Attendees	Sihun Chae, Wooyoung Choi, Donggeun Kim
Goal	To evaluate whether the current Safehome system design meets the four primary objectives outlined in the Software Requirements Specification (SRS): 1. Comprehensive Security 2. User Experience 3. System Reliability and Data Protection 4. Future Expansion toward Smart Living
Discussion	**Sihun**: Today we'll evaluate our system design based on the goals in the SRS. We need to confirm if the current architecture effectively supports all four objectives.

	Wooyoung: Let’s start with security. Our design already integrates multi-layered detection and automated response, right? **Donggeun**: Yes, the IncidentManager, AlarmManager, and SensorRegistry work together for real-time event detection and response. We also planned automated alerts through NotificationService and emergency dispatch via CloudGateway. **Sihun**: That aligns well with the proactive and comprehensive security framework requirement. The system prevents and responds to threats automatically rather than relying solely on user actions. **Wooyoung**: Moving to user experience — the design includes a mobile-first dashboard and one-touch control modes. The Dashboard class supports real-time monitoring, and ModeController manages user-friendly interactions. **Donggeun**: The structure allows simple transitions between modes like “Home,” “Away,” or “Sleep.” Even users without technical knowledge can easily operate the system. **Sihun**: That satisfies the “Seamless and Intuitive User Experience” objective in the SRS. **Wooyoung**: For reliability and data security, we use two-factor authentication through AuthService, access control via AuthZ, and encrypted communication managed by CloudGateway. **Donggeun**: The design also ensures redundancy for critical processes like alarm triggering and video recording. That supports operational stability. **Sihun**: So the system fulfills both reliability and data security requirements by combining access management and encryption. **Wooyoung**: Lastly, regarding future expansion — we already designed modular components such as DeviceManager, SensorRegistry, and CloudServer. Those can easily integrate new smart home features like air quality monitoring or energy management. **Donggeun**: The architecture supports scalability through independent subsystems, so adding new smart devices won’t require major redesigns. **Sihun**: That addresses the goal of evolving into a broader smart living platform.
Conclusion	**Comprehensive Security**: Achieved through IncidentManager, AlarmManager, SensorRegistry, NotificationService, and CloudGateway integration. **User Experience**: Ensured via mobile-first design, intuitive controls, Dashboard, and ModeController. **Reliability & Data Security**: Supported by AuthService, AuthZ, CloudGateway, encryption, and redundancy mechanisms.

	Future Expansion: Enabled by modular architecture (DeviceManager, SensorRegistry, CloudServer) for smart living integration. **Overall Evaluation**: The Safehome system design effectively meets the four key objectives of the SRS. Its architecture provides a strong balance of security, usability, reliability, and scalability for future smart home development.

9. Appendix

    9.1. Glossary

Term	Description
Administrator	The person who sets up the SafeHome system, configures system settings, lays out the floor plan, and places the cameras.
Camera View	The live or recorded visual field captured by a specific surveillance camera.
Control panel	A small gadget to display basic information and receive commands.
Floor plan	A map showing the homeowner's security and surveillance layout.
Guest	A person who temporarily enters the home, such as a housekeeper or repair worker.
Homeowner	The primary user who installs and manages SafeHome security and surveillance features in their home.
Safety Zone	A designated area within or around the home that is continuously monitored for security and safety purposes.
Two-factor authentication	A security mechanism that requires the user to provide two forms of verification before gaining access to the system.