CS350 Safehome Project
Software Design Specification

Team 2

20220686 Minjun Choe

20200315 Daegyu Sung

20230704 Songho Cho

20230091 Minkyeong Kim

Table of Contents

I. Overview 7

1. Introduction 7

2. Goal 8

3. How The Design Work Proceeded 9

4. Assumptions 12

Ⅱ. Architectural Structure 12

1. Overall Architecure 12

2. Architecture Change History 13

2.1 Hub-Centric Monolithic Architecture 13

2.2 Command-Oriented Modularization Trial 14

2.3 Layered Architecture and Dependency Inversion 14

Ⅲ. Class Diagram 16

01. Class Diagram – Whole System Overview [CD-01] 16

1.1. Diagram 16

02. Class Diagram – User [CD-02] 16

2.1. Diagram 16

2.2. Design Rationale and Structural Explanation 17

03. Class Diagram – Core System [CD-03] 17

3.1. Diagram 17

3.2. Design Rationale and Structural Explanation 17

04. Class Diagram – Sensor Manager [CD-04] 18

4.1. Diagram 18

4.2. Design Rationale and Structural Explanation 18

05. Class Diagram – Appliance Manager [CD-05] 19

5.1. Diagram 19

5.2. Design Rationale and Structural Explanation 19

06. Class Diagram – SecurityZone Manager [CD-06] 19

6.1. Diagram 19

6.2. Design Rationale and Structural Explanation 20

07. Class Diagram – SecurityMode Manager [CD-07] 20

7.1. Diagram 20

7.2. Design Rationale and Structural Explanation 20

08. Class Diagram – Alarm Manager [CD-08]........ 21

8.1. Diagram 21

8.2. Design Rationale and Structural Explanation 21

09. Class Diagram – Authentication Service [CD-09]........ 22

9.1. Diagram 22

9.2. Design Rationale and Structural Explanation 22

10. Class Diagram – Database Service [CD-10]........ 23

10.1. Diagram 23

10.2. Design Rationale and Structural Explanation 23

11. Class Diagram – Storage Service [CD-11]........ 23

11.1. Diagram 23

11.2. Design Rationale and Structural Explanation 24

12. Class Diagram – ExternalCall Service [CD-12]........ 25

12.1. Diagram 25

12.2. Design Rationale and Structural Explanation 25

13. Class Diagram – Messaging Service [CD-13]........ 25

13.1. Diagram 25

13.2. Design Rationale and Structural Explanation 26

14. Class Diagram – Logging Service [CD-14]........ 27

14.1. Diagram 27

14.2. Design Rationale and Structural Explanation 27

IV. CRC Card 27

01. CRC Card – User [CR-01]........ 28

02. CRC Card – Core System [CR-02]........ 29

03. CRC Card – Sensor Manager [CR-03]........ 31

04. CRC Card – Appliance Manager [CR-04]........ 34

05. CRC Card – SecurityZone Manager [CR-05]........ 38

06. CRC Card – SecurityMode Manager [CR-06]........ 39

07. CRC Card – Alarm Manager [CR-07]........ 39

08. CRC Card – Authentication Service [CR-08]........ 40

09. CRC Card – Database Service [CR-09] 41
10. CRC Card – Storage Service [CR-10] 42
11. CRC Card – ExternalCall Service [CR-11] 44
12. CRC Card – Messaging Service [CR-12] 45
13. CRC Card – Logging Service [CR-13] 46
Ⅴ. State Diagram 47
01. State Diagram – User [SD-01] 47
02. State Diagram – Core System [SD-02] 48
03. State Diagram – Sensor Manager [SD-03] 48
04. State Diagram – Appliance Manager [SD-04] 48
05. State Diagram – SecurityZone Manager [SD-05] 49
06. State Diagram – SecurityMode Manager [SD-06] 49
07. State Diagram – Alarm Manager [SD-07] 49
08. State Diagram – Authentication Service [SD-08] 49
09. State Diagram – Database Service [SD-09] 50
10. State Diagram – Storage Service [SD-10] 50
11. State Diagram – ExternalCall Service [SD-11] 50
12. State Diagram – Messaging Service [SD-12] 51
13. State Diagram – Logging Service [SD-13] 51
Ⅵ. Design Evaluation 51
1. Architecture Design Metric 51
2. CK Metrics 52
2.1. Calculation Table 52
2.2. CK Metrics 54
3. MOOD Metric 55
3.1. Calculation table 55
3.2. MOOD Metrics 57
Ⅶ. Who Did What 58
1. Minjun Choi 58
2. Daegyu Sung 58

3. Songho Cho 58
4. Minkyeong Kim 58
Ⅷ.Meeting Log 59
1. Meeting 1 [ME-01] 59
2. Meeting 2 [ME-02] 60
3. Meeting 3 [ME-03] 62
4. Meeting 4 [ME-04] 64
Appendix A. Glossary 67
System Components (Hardware) 67
SafeHome Hub 67
Control Panel 67
Sensor 67
Camera (IP Camera) 68
Smart Home Device 68
System Components (Software) 68
SafeHome Cloud Server 68
Mobile Application 69
Web Interface 69
Security Concepts 69
Security Mode 69
Alarm Condition 69
Security Zone 70
Sensor Bypass 70
Panic Button 70
Two-Factor Authentication (2FA) 70
Exit Delay / Entry Delay 71
User Roles and Access 71
Homeowner 71
Admin Homeowner 71
Guest 71
System Administrator (Software) 71
Operational Modes and States 72
Armed / Disarmed 72
Alarm Verification 72
Cooldown Period 72
Activity Log / Timeline 72
Recording and Media 73
Live View 73

Recording 73
Camera Lock / Password Protection 73
Two-Way Audio 73
Evidence Export 74

Device Management 74
Device Registration / Pairing 74
Device Status 74
Sensor Activation / Deactivation 74

Technical Terms 75
Push Notification 75
One-Time Password (OTP) 75
Session Token 75
End-to-End Encryption (E2EE) 75
Audit Log 76
Idempotent Request 76

Environmental Monitoring (Second Increment) 76
Indoor Air Quality (IAQ) 76
Smart Meter 76

Appendix B. Modification 77

B.1 Outdoor Motion Detection – related with UC1.1.3 77

B.2 Alarm Verification Step – related with UC1.2.2 77

I. Overview

1. Introduction

This Software Design Specification (SDS) concretizes the design of the SafeHome Integrated Home Automation System, in alignment with the Software Requirements Specification (SRS). It provides detailed design perspectives including architecture, class diagrams, CRC cards, and state diagrams.

SafeHome is an intelligent platform designed to address both physical intrusions and environmental risks, while enhancing user convenience through unified control and monitoring from mobile or web interfaces.

To translate high-level requirements into implementable design artifacts, this document adopts a layered architecture, event-driven processing, and object-oriented design principles (SOLID). These ensure clear responsibility boundaries, reusable interfaces, and scalability across development, testing, and operation phases.

The system is explicitly decomposed into three structural layers:
SafeHomeHub – Application and domain orchestration
Managers – Core domain logic controllers
Services – Infrastructure and external communication

The overall architecture follows a layered pattern centered around **two core entities**:

> **SafeHomeHub**: Orchestrates the Manager layer, which directly interfaces with field devices (e.g., SensorManager, ApplianceManager, SecurityZoneManager, SecurityModeManager, AlarmManager).
> **CloudServer**: Hosts the Service layer, providing shared platform-level functionalities such as AuthenticationService, DatabaseService, StorageService, ExternalCallService, MessagingService, and LoggingService.

Based on the five Major Functionalities defined in the SRS—"Intelligent Security”, “Live Surveillance”, “System & User Management”, “Remote Access & Account”, and “Indoor Monitoring & Device Control”—the design decomposes concerns into seven domains: Authentication, Data, Logging, External Communication, Alarm, Device, and Security.

These are further realized through 13 modular components, including the User, CoreSystem (SafeHomeHub/CloudServer), five Managers, and six Services. The architecture ensures separation of concerns and low coupling / high cohesion through event-based communication and well-defined interfaces.

Core design principles emphasize real-time responsiveness (low-latency streaming and alerts), reliability (high availability and fault tolerance), and security (E2EE, RBAC, 2FA) as foundational elements.

2. Goal

The goal of the SafeHome design is to establish a robust, scalable, and secure software architecture enabling unified monitoring and automated responses across diverse home security and automation contexts.

The design centers on the SafeHomeHub, which orchestrates domain logic through dedicated Managers based on user commands and system events, while the cloud-based CloudServer provides shared Services and global synchronization.

Distinct boundaries among Managers (e.g., AlarmManager, SecurityModeManager, SensorManager) enable modular handling of complex security scenarios and straightforward extensibility.

We adhere to the following core design principles:
1. **Compliance** – Fully conform to the SRS requirements and analysis models.
2. **Modularity** – Achieve low coupling, high cohesion, and high modularity across components.

3. **Quality Assurance** – Ensure testability, maintainability, integrity, efficiency, and reliability throughout the system lifecycle.
4. **Simplicity & Reusability** – Minimize design complexity while promoting reusability and flexibility.

From a user experience standpoint, remote control via CloudServer, combined with AuthenticationService and MessagingService, enables intuitive operation and transparent system visibility. Also, Security and reliability are prioritized through a high-availability design, real-time system health monitoring via LoggingService, and end-to-end encryption (E2EE) across data transmission and storage. Role-based access control (RBAC) and two-factor authentication (2FA) are treated as core architectural requirements, not optional features.

Beyond security, SafeHome evolves into a comprehensive smart-home hub through:
- A pluggable device architecture (ApplianceManager, CameraApplianceManager)
- Hybrid local/cloud storage strategies using StorageService
- Optimized live streaming via LiveStreamingService

These enable energy-efficient automation based on real-time environmental metrics (e.g., air quality, power consumption), supporting an integrated and adaptive smart-home ecosystem.

3. How The Design Work Proceeded

The design process evolved through multiple iterative and analytical stages to ensure alignment between the SRS-defined requirements and the final architecture.

1. Initial Functional Decomposition

The design process began with a noun-verb analysis of the five _Major Functionalities_ specified in the SRS. The goal was to derive five corresponding submodules reflecting each functionality. However, during the analysis, it became evident that these functionalities were not mutually exclusive—they shared overlapping responsibilities and cross-cutting concerns. As a result, achieving a clean architectural separation based strictly on these five functions proved infeasible.

2. AI-Assisted Architectural Experimentation

Next, we attempted to employ AI-driven architecture generation using _Cursor Agents_ and the _PlantUML plugin_, aiming to construct a context-based hierarchical design automatically.

However, this approach encountered significant challenges:

- The agents failed to maintain consistent abstraction levels across design layers.
- Due to the large and interdependent nature of the SRS requirements and use cases, the models frequently lost contextual grounding, resulting in semantic drift and hallucinations in the generated diagrams. Therefore, we do a next approach.

3. Manual Domain-Centric Redesign

Consequently, we transitioned to a manual and domain-driven approach by reviewing all use cases directly. This led to the formulation of a Core System-oriented abstraction, where:

- The SafeHomeHub governs and coordinates the _Manager_ layer.
- The CloudServer supervises and integrates the _Service_ layer.

Through this structure, the entire system was modularized into 13 well-defined components:

`Authentication`, `Database`, `ExternalCall`, `Storage`, `Messaging`, `Logging`, `SecurityZone`, `SecurityMode`, `Appliance`, `Sensor`, `Alarm`, `CoreSystem`, and `User`.

4. Iterative Refinement and Metric Evaluation

Using this component structure, we iteratively refined the Class Diagram while measuring software design metrics to ensure structural quality. Abstract classes and interfaces were strategically introduced to enforce low coupling and high cohesion. Repeated class methods and shared attributes were extracted into abstract superclasses, while Manager and Service layers were used to enforce separation of concerns.

Although Cursor Agents were insufficient for autonomously generating consistent architecture, they proved highly valuable in evaluating design consistency and proposing architectural improvements. The agents provided meaningful feedback during refactoring iterations—particularly in identifying potential code smells, redundant class responsibilities, and unbalanced dependencies—serving as effective AI-assisted design reviewers.

5. CRC Card Development

Based on the constructed class relationships, CRC (Class-Responsibility-Collaboration) cards were created to formalize class roles, responsibilities, and collaborations.

6. Class Diagram Revision

The CRC analysis revealed issues of mutual reference and directionality among certain classes. These relationships were carefully revised to improve dependency orientation and interaction clarity, resulting in a cleaner and more maintainable class structure.

7. State Diagram Construction

Subsequently, State Diagrams were developed to capture the dynamic behaviors of major components.

In addition to the 13 previously defined sections, several new classes requiring explicit state transitions were introduced—such as `Camera`, `ExportService`, and `StreamingService`. We highly utilize using Cursor Agents in this progress also.

8. State Diagram Review

The state models underwent internal review to ensure logical consistency, completeness, and traceability with the corresponding class behaviors.

9. Finalization

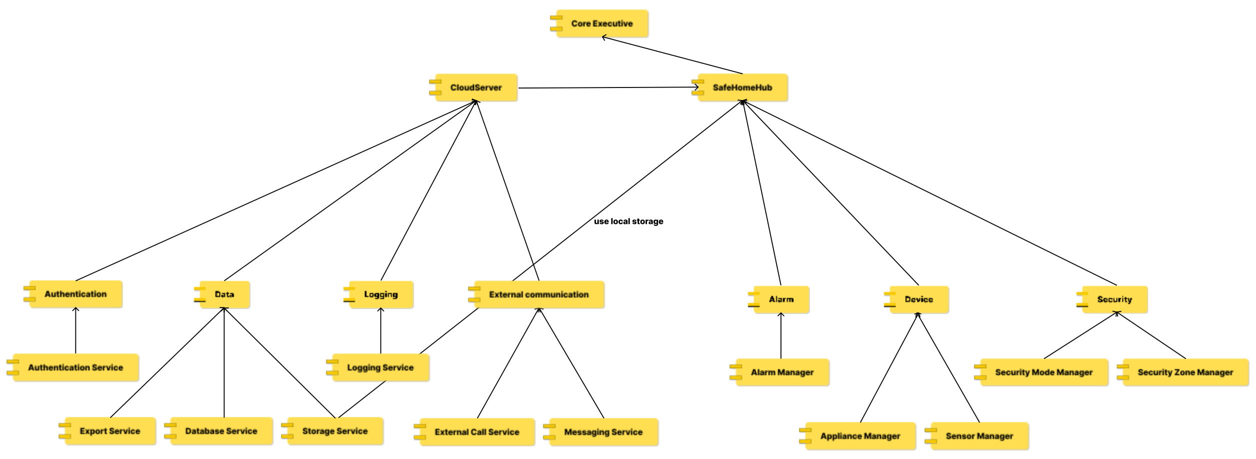
After multiple refinement cycles, both the Class Diagram and State Diagram were finalized, establishing a cohesive design foundation that connects structural and behavioral perspectives of the SafeHome system.

4. Assumptions

1. All assumptions stated in the SRS document are inherited and remain valid.
2. Any cases that differ from the original SRS assumptions are described in Appendix B. Modification. (Meeting 1 [ME-01])
3. The SafeHomeHub functions both as a physical control panel and as a mobile application. Users can interact with the system through either interface, and all user actions are processed and managed by the SafeHomeHub Core System.
4. Safety zone is set based on Safehome_design_guideline. Also, all floor are assumed to be static and predetermined, so no register/remove functionailty is needed. (Slide 2, Meeting 3 [ME-03]).
5. Sensor locations and safety zone scope is shown as coordinates. (Meeting 4 [ME-04])
6. Sensor belongs to at most one safety zone. (In class by professor, Meeting 1 [ME-01])
7. Camera will not perform barking detection; it is handled exclusively by the sound sensor. (Meeting 2 [ME-02])
8. Both Appliance and Sensor classes will include a Passive Power Consumption attribute. (Meeting 3 [ME-03])

Ⅱ. Architectural Structure

1. Overall Architecure



Reference on [ME-03]

2. Architecture Change History

The architecture of SafeHome evolved through three major iterations, each addressing key limitations identified in prior versions. This section documents the progression from an early monolithic hub-centric design to the final layered and dependency-inverted architecture.

2.1 Hub-Centric Monolithic Architecture

Key Changes

- SafeHomeHub directly referenced all major entities, including user, sensor, alarm, and external call modules.
- The CloudServer and Hub were mutually dependent, lacking clear permission boundaries.
- AlarmService consisted of concrete subclasses (Verification, Dispatch, Control) directly placed under the Hub.

Rationale (why we choose this design)

- Provided a quick and intuitive visualization of SRS-level functionalities for early prototyping.
- Established a preliminary Domain-Driven Design (DDD) package structure.
- Simplified system flow by centralizing all managers under the Hub as a single entry point—suitable for early prototypes and demos.

Problems Identified

- Circular dependency among Hub, CloudServer, and Authentication modules caused build and test difficulties.
- Overloaded responsibilities within SafeHomeHub, violating the Single Responsibility Principle (SRP).
- Absence of user permission and configuration management, limiting both security and personalization capabilities.

2.2 Command-Oriented Modularization Trial

Key Changes

- Introduced a SafeHomeCommand layer to encapsulate control of SecurityMode, Sensor, and Appliance components as command objects.
- Added Permission, ManualPermission, and UserSettings to establish a user-centric access control strategy.
- Abstracted AlarmAction and Storage as interfaces, and extended the domain model with a Point class.

Rationale (why we choose this design)

- Applied the Command Pattern to suppress the proliferation of Hub methods and secure Open/Closed Principle (OCP) compliance.
- Encapsulated the permission policy using the Strategy Pattern, enabling user-specific behavior customization.
- Delegated alarm and external integration tasks to CloudServer, allowing the Hub to focus solely on local device control and reinforcing SRP.
- Abstracted Storage and Messaging into interfaces to achieve Dependency Inversion Principle (DIP) and Interface Segregation Principle (ISP).

Problems Identified

- AuthenticationService remained ambiguously linked to both Hub and Cloud, blurring dependency direction.
- Multiple Messaging services existed without a unified interface, causing code duplication and test complexity.
- Although commands improved modularity, related actions such as authentication, logging, and storage remained scattered and inconsistently managed.

2.3 Layered Architecture and Dependency Inversion

Key Changes

- Relocated AuthenticationService entirely to the Cloud layer, allowing the Hub to depend only on abstractions, thereby eliminating circular dependencies via DIP.
- Defined a unified Message interface with concrete implementations (EmailMessage, SMSMessage, PushNotificationMessage), ensuring polymorphism and consistent naming across services (Manager → Service).
- Extended the Command layer to include authentication-related actions, with CloudServer orchestrating DatabaseService, ExternalCallService, and MessagingService.
- Introduced the Value Object Point to model spatial information within Sensor and SecurityZone domains.

Rationale (why we choose this design)

- Adopted a Layered Architecture separating Edge (Hub), Cloud, and Data layers, enabling local fallback operation during network failures.
- Applied the Dependency Inversion Principle (DIP) to decouple high-level modules (Hub) from low-level implementations (Cloud), simplifying unit testing and mock injection.
- Established shared interfaces (e.g., Messaging, Storage, Permission) to realize the Interface Segregation Principle (ISP) and minimize code modification when adding new channels or services.
- Strengthened domain modeling through Point Value Objects, ensuring consistency in spatial computation and user interface alignment.

Remaining Risks / Limitations

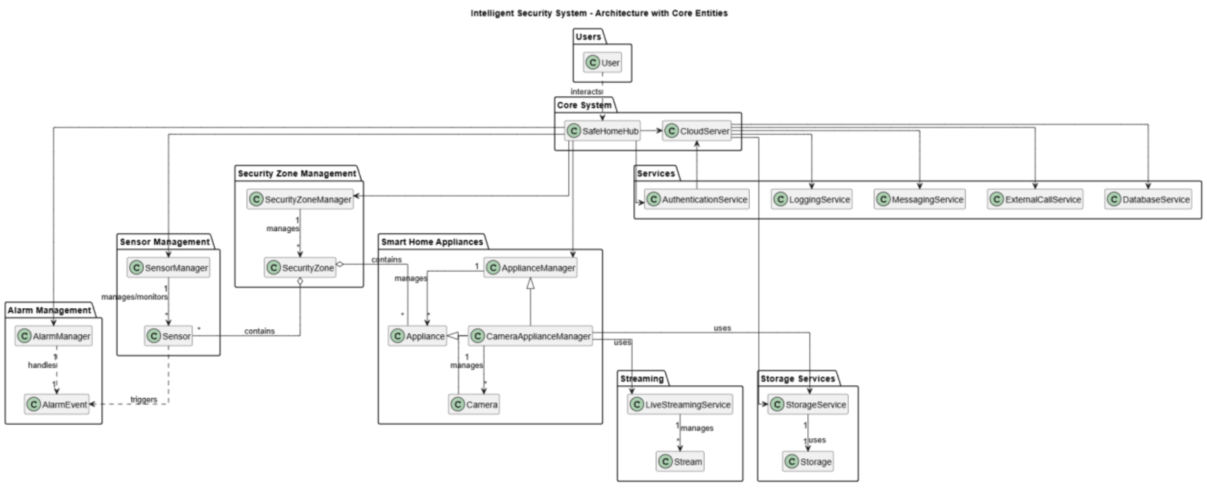
- The growing number of command objects suggests a potential need for a dedicated orchestration layer for command validation and coordination.
- The SafeHomeHub has continuously expanded in responsibility, managing an increasing number of Managers. To address this growing complexity, a sub-Hub system is planned, where related Managers are grouped under specialized sub-hubs for localized coordination and simplified maintenance.
- The CloudServer now encapsulates a large set of responsibilities, indicating a possible future transition toward microservice decomposition.

- The overall diagram has become increasingly complex, requiring further visual segmentation into function-specific subviews for maintainability.

Ⅲ. Class Diagram

01. Class Diagram – Whole System Overview [CD-01]

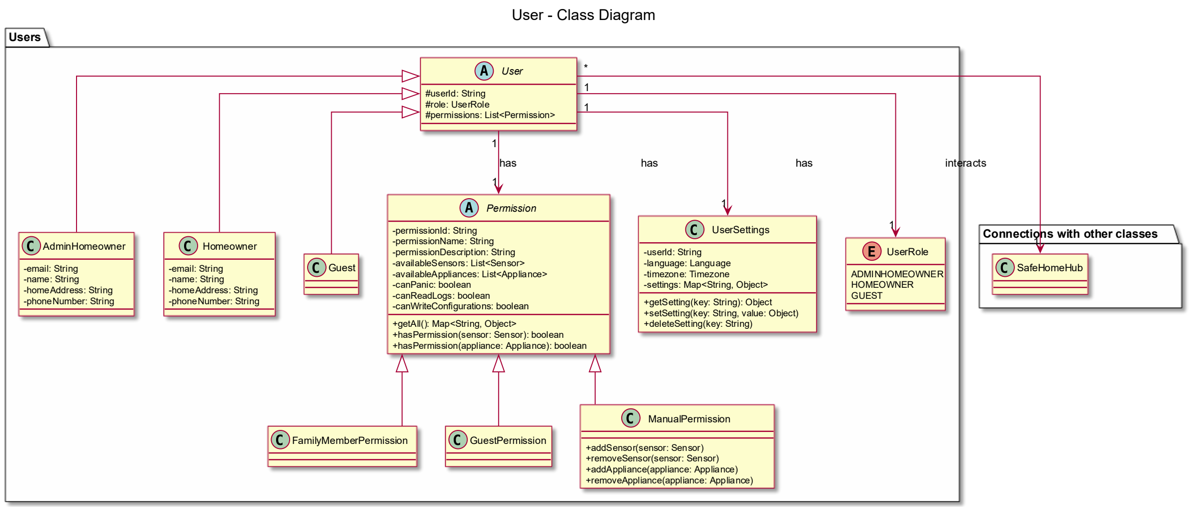
1.1. **Diagram**



Note:
Due to the large size of the overall class diagram, the complete version is provided as an external image file, available at the following location: [https://drive.google.com/drive/folders/1TreunUzTMWlf7mETwbFSw7mH9ai_5gq1?usp=sharing]

02. Class Diagram – User [CD-02]

2.1. **Diagram**


User - Class Diagram
Users
User
#userId: String
#role: UserRole
#permissions: List<Permission>
AdminHomeowner
-email: String
-name: String
-homeAddress: String
-phoneNumber: String
Homeowner
-email: String
-name: String
-homeAddress: String
-phoneNumber: String
Guest
has
Permission
-permissionId: String
-permissionName: String
-permissionDescription: String
-availableSensors: List<Sensor>
-availableAppliances: List<Appliance>
-canPanic: boolean
-canReadLogs: boolean
-canWriteConfigurations: boolean
+getAll(): Map<String, Object>
+hasPermission(sensor: Sensor): boolean
+hasPermission(appliance: Appliance): boolean
has
UserSettings
-userId: String
-language: Language
-timezone: Timezone
-settings: Map<String, Object>
+getSetting(key: String): Object
+setSetting(key: String, value: Object)
+deleteSetting(key: String)
has
UserRole
ADMINHOMEOWNER
HOMEOWNER
GUEST
interacts
Connections with other classes
SafeHomeHub
FamilyMemberPermission
GuestPermission
ManualPermission
+addSensor(sensor: Sensor)
+removeSensor(sensor: Sensor)
+addAppliance(appliance: Appliance)
+removeAppliance(appliance: Appliance)

2.2. **Design Rationale and Structural Explanation**

Structural Explanation:

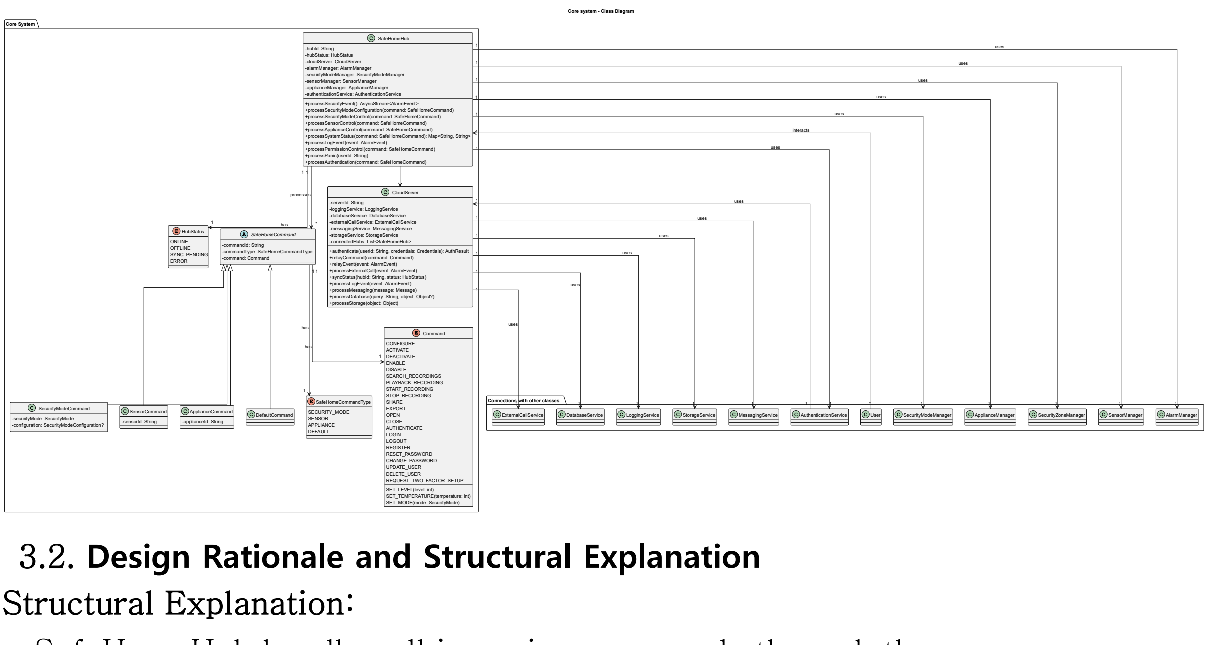
The User abstract class serves as the root entity, inherited by AdminHomeowner, Homeowner, and Guest.
User authorization is managed through the Permission hierarchy (FamilyPermission, GuestPermission, ManualPermission), while personalization and preferences are encapsulated in the UserSettings class.

Rationale:

By combining Role-Based Access Control (RBAC) with the Strategy Pattern, user-permission relationships remain loosely coupled.
The separation of UserSettings as a value object upholds Single Responsibility Principle (SRP), and the composable ManualPermission class enhances extensibility for future policy additions.

03. Class Diagram – Core System [CD-03]

3.1. **Diagram**



Core system - Class Diagram
Core System
SafeHomeHub
-hubId: String
-hubStatus: HubStatus
-cloudServer: CloudServer
-alarmManager: AlarmManager
-securityModeManager: SecurityModeManager
-sensorManager: SensorManager
-applianceManager: ApplianceManager
-authenticationService: AuthenticationService
+processSecurityEvent(): AsyncStream<AlarmEvent>
+processSecurityModeConfiguration(command: SafeHomeCommand)
+processSecurityModeControl(command: SafeHomeCommand)
+processSensorControl(command: SafeHomeCommand)
+processApplianceControl(command: SafeHomeCommand)
+processSystemStatus(command: SafeHomeCommand): Map<String, String>
+processLogEvent(event: AlarmEvent)
+processPermissionControl(command: SafeHomeCommand)
+processPanic(userId: String)
+processAuthentication(command: SafeHomeCommand)
CloudServer
-serverId: String
-loggingService: LoggingService
-databaseService: DatabaseService
-externalCallService: ExternalCallService
-messagingService: MessagingService
-storageService: StorageService
-connectedHubs: List<SafeHomeHub>
+authenticate(userId: String, credentials: Credentials): AuthResult
+relayCommand(command: Command)
+relayEvent(event: AlarmEvent)
+processExternalCall(event: AlarmEvent)
+syncStatus(hubId: String, status: HubStatus)
+processLogEvent(event: AlarmEvent)
+processMessaging(message: Message)
+processDatabase(query: String, object: Object?)
+processStorage(object: Object)
HubStatus
ONLINE
OFFLINE
SYNC_PENDING
ERROR
SafeHomeCommand
-commandId: String
-commandType: SafeHomeCommandType
-command: Command
processes
has
has
has
SecurityModeCommand
-securityMode: SecurityMode
-configuration: SecurityModeConfiguration?
SensorCommand
-sensorId: String
ApplianceCommand
-applianceId: String
DefaultCommand
SafeHomeCommandType
SECURITY_MODE
SENSOR
APPLIANCE
DEFAULT
Command
CONFIGURE
ACTIVATE
DEACTIVATE
ENABLE
DISABLE
SEARCH_RECORDINGS
PLAYBACK_RECORDING
START_RECORDING
STOP_RECORDING
SHARE
EXPORT
OPEN
CLOSE
AUTHENTICATE
LOGIN
LOGOUT
REGISTER
RESET_PASSWORD
CHANGE_PASSWORD
UPDATE_USER
DELETE_USER
REQUEST_TWO_FACTOR_SETUP
SET_LEVEL(level: int)
SET_TEMPERATURE(temperature: int)
SET_MODE(mode: SecurityMode)
uses
interacts
Connections with other classes
ExternalCallService
DatabaseService
LoggingService
StorageService
MessagingService
AuthenticationService
User
SecurityModeManager
ApplianceManager
SecurityZoneManager
SensorManager
AlarmManager

3.2. **Design Rationale and Structural Explanation**

Structural Explanation:

SafeHomeHub handles all incoming commands through the SafeHomeCommand hierarchy (SensorCommand, ApplianceCommand, SecurityModeCommand).

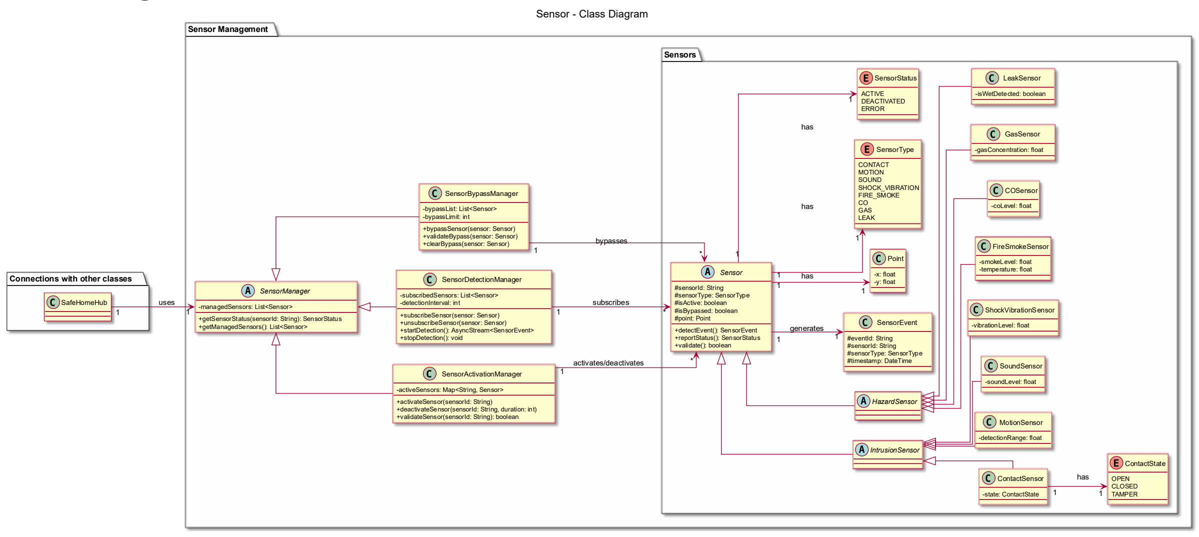
CloudServer manages supporting services such as logging, database access, messaging, and storage orchestration.

Rationale:

The Command Pattern encapsulates Hub control logic, reducing complexity and improving extensibility.
By maintaining a unidirectional dependency (Hub → Cloud), the design realizes the Dependency Inversion Principle (DIP) and upholds a clear Layered Architecture.

04. Class Diagram – Sensor Manager [CD-04]

4.1. Diagram



4.2. **Design Rationale and Structural Explanation**

Structural Explanation:

SensorManager abstracts three functional modules—ActivationManager, DetectionManager, and BypassManager.
The Sensor hierarchy distinguishes between intrusion and environmental sensors, each containing a Point value object for spatial representation.

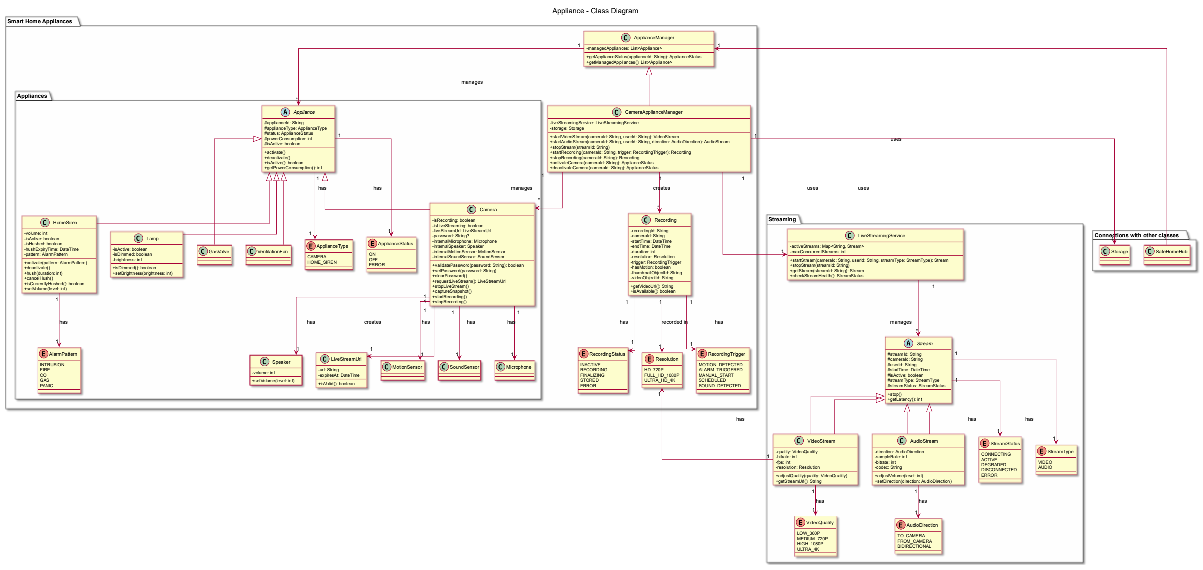
Rationale:

Functional segmentation maintains high cohesion and fulfills LSP/OCP through hierarchical design.

The Point value object represents spatial domain rules using DDD Value Object modeling, ensuring consistent spatial policy handling.

05. Class Diagram – Appliance Manager [CD-05]

5.1. **Diagram**



5.2. **Design Rationale and Structural Explanation**

Structural Explanation:

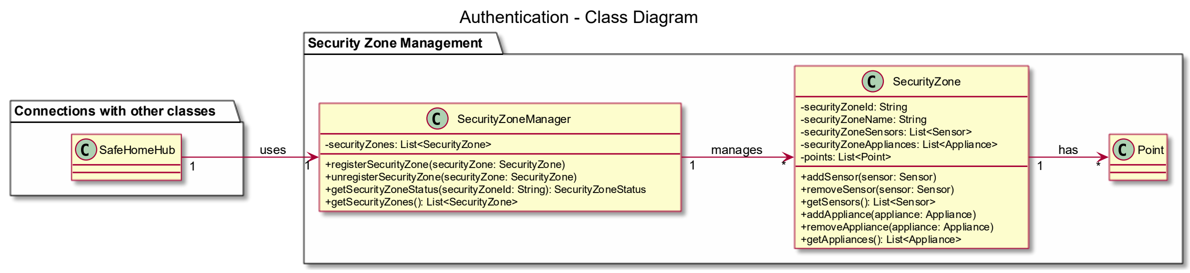
ApplianceManager controls the shared lifecycle of devices, while CameraApplianceManager extends it with streaming and recording functionalities.
The Appliance abstract class encapsulates common properties across devices such as cameras, sirens, and lamps.

Rationale:

The Template Method and inheritance structure allow per-device customization while maintaining a unified interface.
This promotes OCP compliance and consistent state management for power and operational status across all appliance types.

06. Class Diagram – SecurityZone Manager [CD-06]

6.1. **Diagram**


Authentication - Class Diagram
Security Zone Management
Connections with other classes
SafeHomeHub
uses
1
1
SecurityZoneManager
-securityZones: List<SecurityZone>
+registerSecurityZone(securityZone: SecurityZone)
+unregisterSecurityZone(securityZone: SecurityZone)
+getSecurityZoneStatus(securityZoneId: String): SecurityZoneStatus
+getSecurityZones(): List<SecurityZone>
manages
1
*
SecurityZone
-securityZoneId: String
-securityZoneName: String
-securityZoneSensors: List<Sensor>
-securityZoneAppliances: List<Appliance>
-points: List<Point>
+addSensor(sensor: Sensor)
+removeSensor(sensor: Sensor)
+getSensors(): List<Sensor>
+addAppliance(appliance: Appliance)
+removeAppliance(appliance: Appliance)
+getAppliances(): List<Appliance>
has
1
*
Point

6.2. **Design Rationale and Structural Explanation**

Structural Explanation:

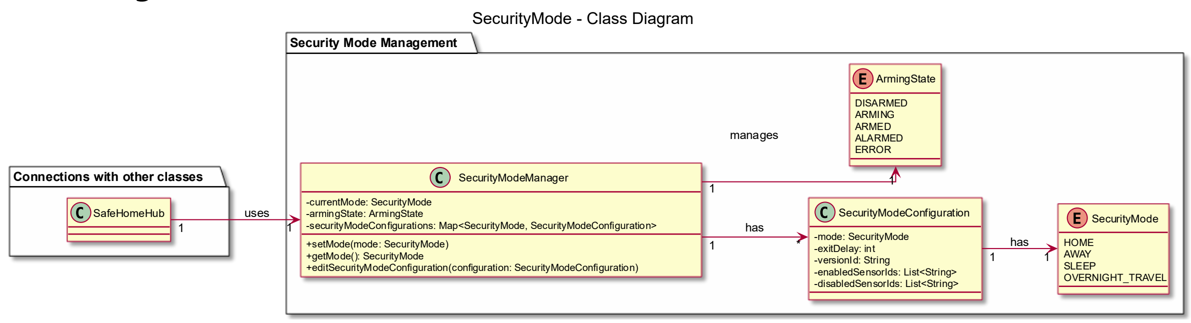
SecurityZoneManager maintains a collection of SecurityZone objects. Each zone aggregates multiple sensors, appliances, and spatial coordinates (Point).

Rationale:

By defining SecurityZone as an Aggregate Root, the design follows DDD principles, allowing grouped control of related IoT devices.
This reduces coupling by managing many-to-many relationships indirectly through zones.

07. Class Diagram – SecurityMode Manager [CD-07]

7.1. **Diagram**


SecurityMode - Class Diagram
Security Mode Management
Connections with other classes
SafeHomeHub
uses
1
1
SecurityModeManager
-currentMode: SecurityMode
-armingState: ArmingState
-securityModeConfigurations: Map<SecurityMode, SecurityModeConfiguration>
+setMode(mode: SecurityMode)
+getMode(): SecurityMode
+editSecurityModeConfiguration(configuration: SecurityModeConfiguration)
manages
1
1
ArmingState
DISARMED
ARMING
ARMED
ALARMED
ERROR
has
1
*
SecurityModeConfiguration
-mode: SecurityMode
-exitDelay: int
-versionId: String
-enabledSensorIds: List<String>
-disabledSensorIds: List<String>
has
1
1
SecurityMode
HOME
AWAY
SLEEP
OVERNIGHT_TRAVEL

7.2. **Design Rationale and Structural Explanation**

Structural Explanation:

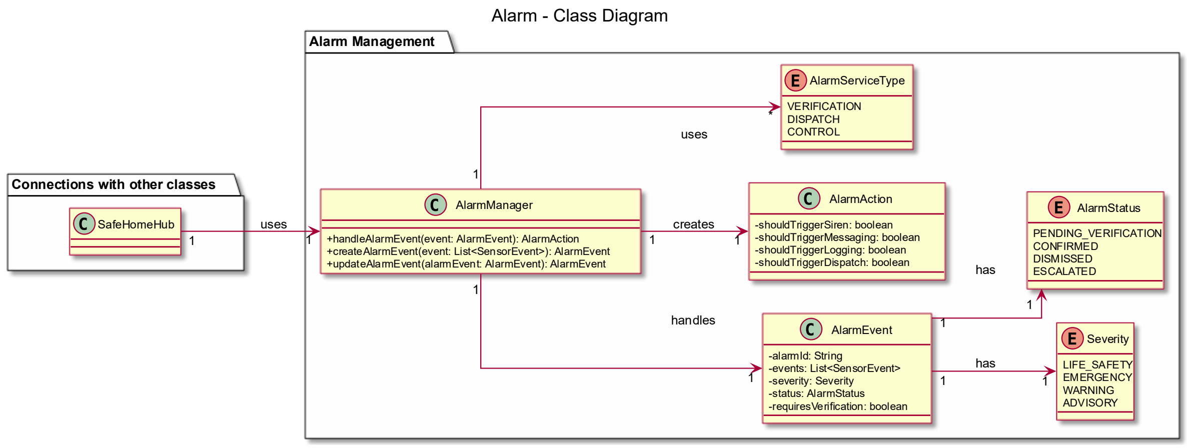
SecurityModeManager manages the active mode and ArmingState, while SecurityModeConfiguration handles versioned sensor activation profiles.

Rationale:

Combining the State and Strategy patterns, this design clarifies mode transition logic and isolates configuration changes.
Maintaining configurations as separate entities enables version tracking, rollback, and auditability.

08. Class Diagram – Alarm Manager [CD-08]

8.1. Diagram


Alarm - Class Diagram
Alarm Management
Connections with other classes
SafeHomeHub
uses
AlarmManager
+handleAlarmEvent(event: AlarmEvent): AlarmAction
+createAlarmEvent(event: List<SensorEvent>): AlarmEvent
+updateAlarmEvent(alarmEvent: AlarmEvent): AlarmEvent
uses
AlarmServiceType
VERIFICATION
DISPATCH
CONTROL
creates
AlarmAction
-shouldTriggerSiren: boolean
-shouldTriggerMessaging: boolean
-shouldTriggerLogging: boolean
-shouldTriggerDispatch: boolean
handles
AlarmEvent
-alarmId: String
-events: List<SensorEvent>
-severity: Severity
-status: AlarmStatus
-requiresVerification: boolean
has
AlarmStatus
PENDING_VERIFICATION
CONFIRMED
DISMISSED
ESCALATED
has
Severity
LIFE_SAFETY
EMERGENCY
WARNING
ADVISORY
1

8.2. **Design Rationale and Structural Explanation**

Structural Explanation:

The AlarmManager handles the creation and update of AlarmEvent instances.
During event processing, the AlarmManager determines appropriate responses—such as whether to trigger the siren or send a notification—based on the AlarmAction object.
Once the decision is made, the resulting AlarmAction is passed to the SafeHomeHub, which executes and manages the corresponding actions.
The AlarmAction class encapsulates decision outcomes as structured data, enabling consistent interpretation and processing across components.

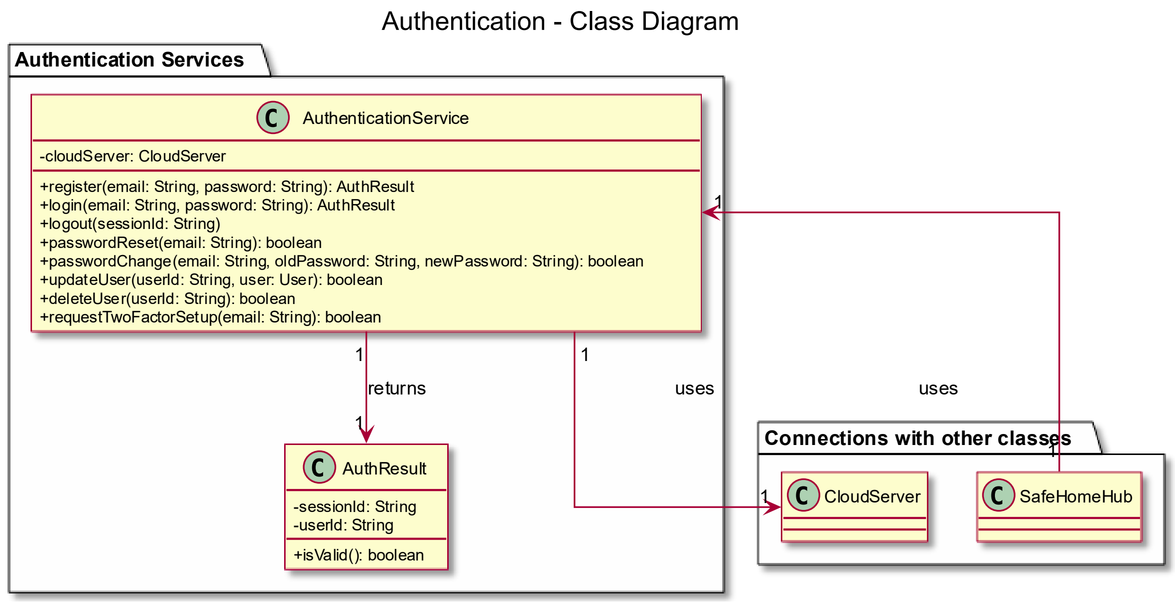
Rationale:

By applying the Chain of Responsibility, each stage of alarm handling is modularized into independent processes, ensuring compliance with the Single Responsibility Principle (SRP).

Externalizing decision logic into AlarmAction data transfer objects (DTOs) allows for flexible rule-based evaluation, easier testing, and clearer traceability of decision outcomes.

09. Class Diagram – Authentication Service [CD-09]

9.1. **Diagram**


Authentication - Class Diagram
Authentication Services
AuthenticationService
-cloudServer: CloudServer
+register(email: String, password: String): AuthResult
+login(email: String, password: String): AuthResult
+logout(sessionId: String)
+passwordReset(email: String): boolean
+passwordChange(email: String, oldPassword: String, newPassword: String): boolean
+updateUser(userId: String, user: User): boolean
+deleteUser(userId: String): boolean
+requestTwoFactorSetup(email: String): boolean
1
returns
1
AuthResult
-sessionId: String
-userId: String
+isValid(): boolean
1
uses
1
uses
1
1
Connections with other classes
CloudServer
SafeHomeHub

9.2. **Design Rationale and Structural Explanation**

Structural Explanation:

The AuthenticationService depends on the CloudServer to handle user registration, login, and two-factor authentication (2FA), and returns session details encapsulated in the AuthResult object.

Initially, the SafeHomeHub manages user authentication commands such as login and logout. These commands are passed from the Hub to the AuthenticationService, which processes them by coordinating with the CloudServer—for example, saving user credentials, validating passwords, or retrieving current authentication states.

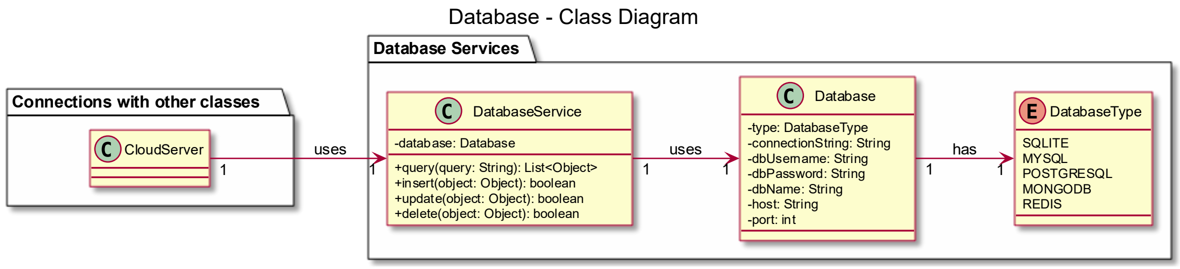
This design establishes a clear separation between user interaction (Hub layer) and security processing (Cloud layer) while maintaining a consistent command-based workflow.

Rationale:

By isolating authentication as an independent service, security responsibilities are decoupled from the Hub.
Encapsulating session results as objects ensures consistent event logging, validation, and failure handling across the system.

10. Class Diagram – Database Service [CD-10]

10.1. Diagram



10.2. Design Rationale and Structural Explanation

Structural Explanation:

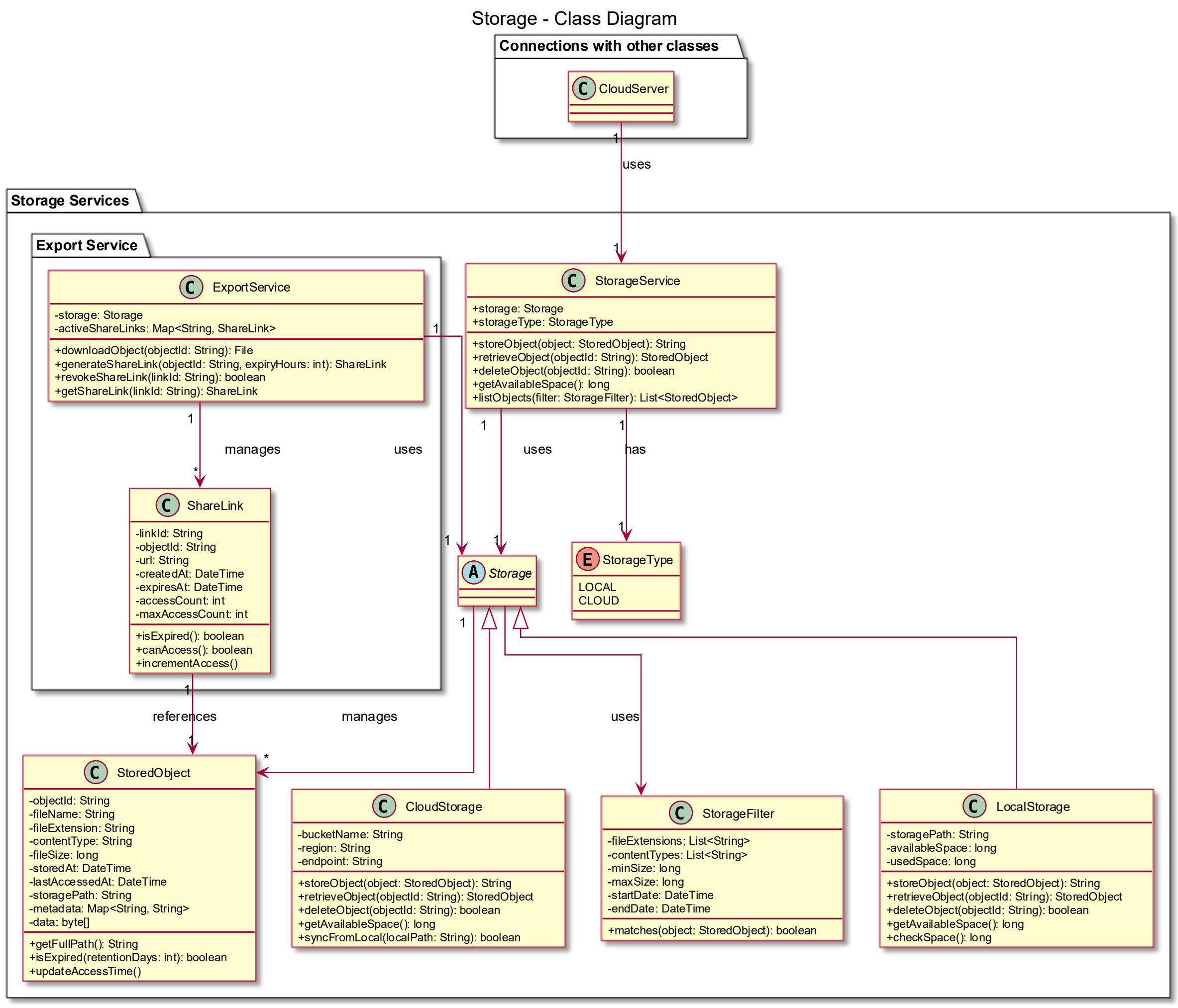
DatabaseService performs CRUD operations using a Database entity and abstracts the storage engine via the DatabaseType enumeration.

Rationale:

Following the Repository Pattern, the design separates domain logic from data persistence, aligning with DIP.
This abstraction provides flexibility for future database engine substitution without structural modification.

11. Class Diagram – Storage Service [CD-11]

11.1. Diagram


Storage - Class Diagram
Connections with other classes
CloudServer
uses
Storage Services
Export Service
ExportService
-storage: Storage
-activeShareLinks: Map<String, ShareLink>
+downloadObject(objectId: String): File
+generateShareLink(objectId: String, expiryHours: int): ShareLink
+revokeShareLink(linkId: String): boolean
+getShareLink(linkId: String): ShareLink
StorageService
+storage: Storage
+storageType: StorageType
+storeObject(object: StoredObject): String
+retrieveObject(objectId: String): StoredObject
+deleteObject(objectId: String): boolean
+getAvailableSpace(): long
+listObjects(filter: StorageFilter): List<StoredObject>
manages
uses
uses
has
ShareLink
-linkId: String
-objectId: String
-url: String
-createdAt: DateTime
-expiresAt: DateTime
-accessCount: int
-maxAccessCount: int
+isExpired(): boolean
+canAccess(): boolean
+incrementAccess()
Storage
StorageType
LOCAL
CLOUD
references
manages
uses
StoredObject
-objectId: String
-fileName: String
-fileExtension: String
-contentType: String
-fileSize: long
-storedAt: DateTime
-lastAccessedAt: DateTime
-storagePath: String
-metadata: Map<String, String>
-data: byte[]
+getFullPath(): String
+isExpired(retentionDays: int): boolean
+updateAccessTime()
CloudStorage
-bucketName: String
-region: String
-endpoint: String
+storeObject(object: StoredObject): String
+retrieveObject(objectId: String): StoredObject
+deleteObject(objectId: String): boolean
+getAvailableSpace(): long
+syncFromLocal(localPath: String): boolean
StorageFilter
-fileExtensions: List<String>
-contentTypes: List<String>
-minSize: long
-maxSize: long
-startDate: DateTime
-endDate: DateTime
+matches(object: StoredObject): boolean
LocalStorage
-storagePath: String
-availableSpace: long
-usedSpace: long
+storeObject(object: StoredObject): String
+retrieveObject(objectId: String): StoredObject
+deleteObject(objectId: String): boolean
+getAvailableSpace(): long
+checkSpace(): long

11.2. **Design Rationale and Structural Explanation**

Structural Explanation:

The StorageService operates on the Storage abstract class, with two concrete implementations: LocalStorage and CloudStorage. Data objects are encapsulated as StoredObject instances and queried through the StorageFilter class. Since export operations are frequently executed in conjunction with storage activities, the ExportService is integrated within the StorageService layer.
ExportService manages the creation, expiration, and lifecycle of shared links, ensuring secure and efficient data sharing.

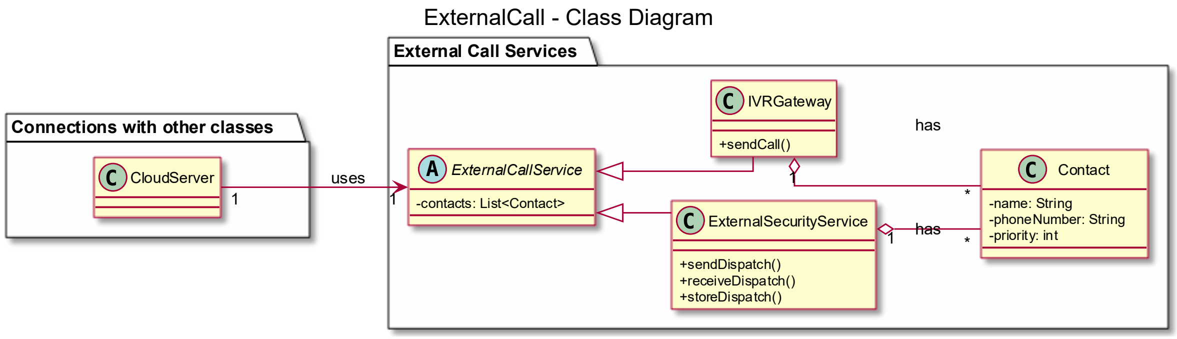
Rationale:

The combination of Interface Segregation and Strategy Pattern enables seamless switching between storage backends.

Security is reinforced by delegating export management (expiration, access count) to a distinct ExportService, maintaining SRP.

12. Class Diagram – ExternalCall Service [CD-12]

12.1. Diagram



12.2. Design Rationale and Structural Explanation

Structural Explanation:

ExternalCallService acts as an abstract base for ExternalSecurityService and IVRGateway, operating in conjunction with Contact entities.

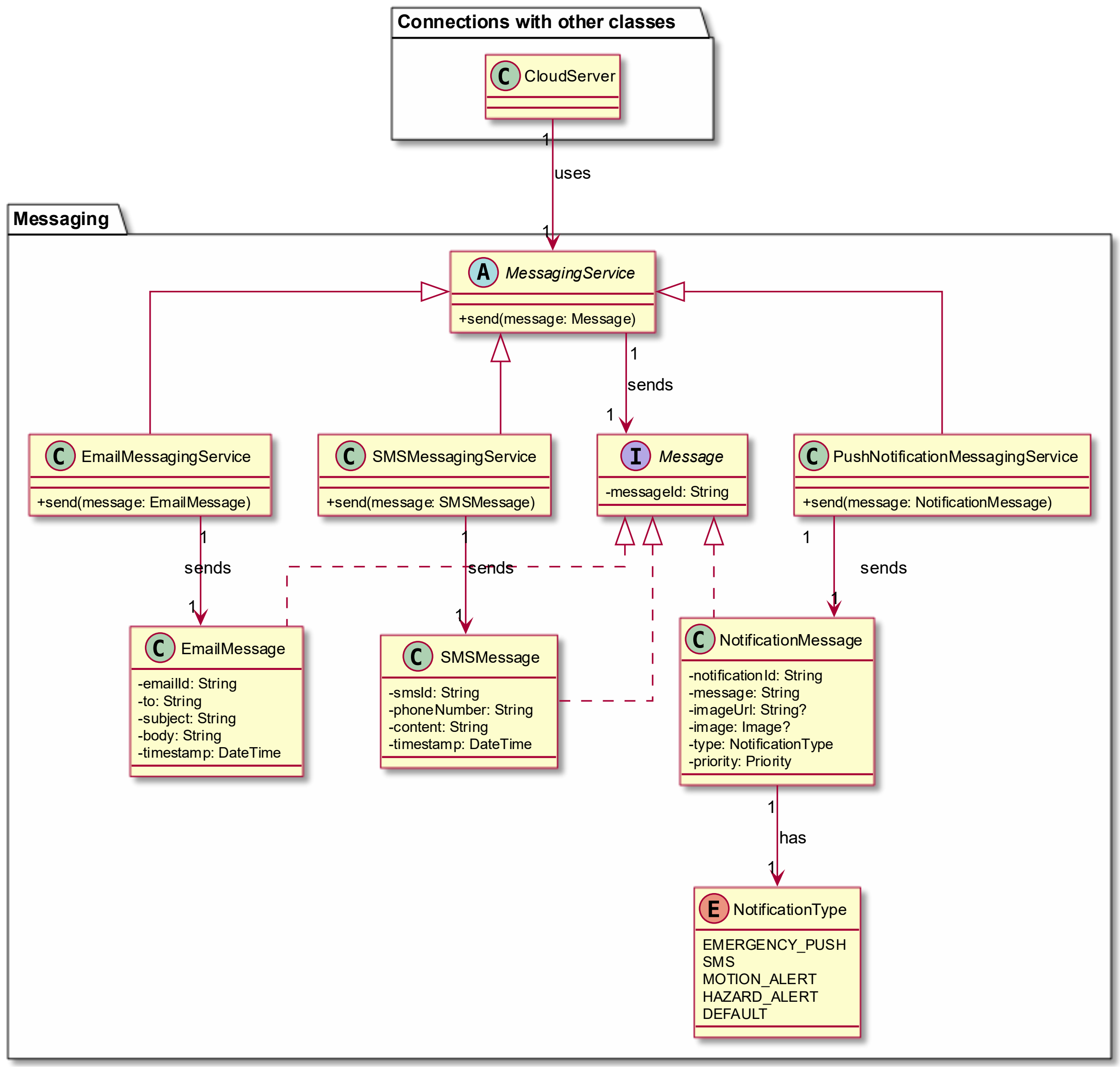
Rationale:

By defining a common interface for external communication channels, the system simplifies future integration of new services.
Contacts are modeled as value objects, streamlining priority handling and reducing coupling between communication mechanisms.

13. Class Diagram – Messaging Service [CD-13]

13.1. Diagram

Messaging - Class Diagram



13.2. **Design Rationale and Structural Explanation**

Structural Explanation:

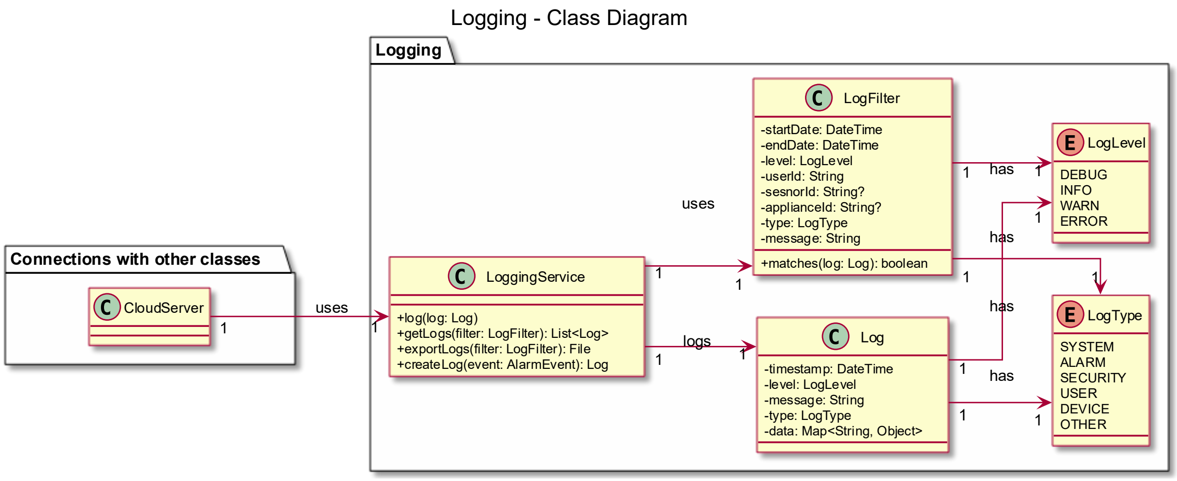
The MessagingService is separated into three specialized services — EmailMessagingService, SMSMessagingService, and PushNotificationMessagingService.
To send messages at a unified abstraction level, all services accept the common Message interface, implemented by EmailMessage, SMSMessage, and PushNotificationMessage.
Additionally, NotificationMessage extends the message model by incorporating image and priority metadata for richer content delivery.

Rationale:

By leveraging interface-based polymorphism and the Strategy Pattern, each messaging channel can be extended or modified independently without affecting others.
All message entities are designed as immutable value objects, ensuring thread safety, consistency across concurrent operations, and easy reuse within message queues and logging pipelines.

14. Class Diagram – Logging Service [CD-14]

14.1. **Diagram**



14.2. **Design Rationale and Structural Explanation**

Structural Explanation:

LoggingService provides centralized creation, retrieval, and export of Log entries.
Filtering is handled through the LogFilter object, while classification uses LogLevel and LogType enumerations.

Rationale:

Encapsulating complex filtering logic within a filter object simplifies query composition. Providing a unified logging service establishes a single, auditable entry point for system-wide traceability and diagnostics.

Ⅳ. CRC Card

01. CRC Card – User [CR-01] Class Diagram, State Diagram

Class: User (Abstract Class) [C-1.1]	
Manage the user identity, roles, and permissions across the SafeHome system.	
Responsibilites	**Collaborators**
Manage user identifier and role	
Maintain list of permissions for access control	Permission
Interact with SafeHomeHub for system operations	SafeHomeHub
Store and manage user-specific settings	UserSettings

Class: AdminHomeowner [C-1.2]	
Manage the administrative settings and high-priority security operations for the home.	
Responsibilites	**Collaborators**
Manage residential settings and advanced security configurations	SafeHomeHub
Perform special operations such as panic calls for high-risk events	SafeHomeHub
Adjust permissions for family members and guests	Permission
Configure security modes and alarm settings	SecurityModeManager
Store admin contact and profile information	UserSettings
Authenticate and manage admin account	AuthenticationService

Class: Homeowner [C-1.3]	
Manage daily security operations and device control as a regular resident.	
Responsibilites	**Collaborators**
Control sensors and devices daily as a regular resident	SafeHomeHub
Perform security mode switching and alarm verification	SecurityModeManager
Maintain user settings and contact information	UserSettings
	Permission

Class: Permission (Abstract Class) [C-1.5]	
Define which sensors and appliances a user can access.	
Responsibilites	**Collaborators**
Maintain list of sensors and devices accessible to users	Sensor
Maintain list of appliances accessible to users	Appliance

Define permission flags for panic, logs, and configurations	User
Check if user has permission for specific sensor	Sensor
Check if user has permission for specific appliance	Appliance

Class: ManualPermission [C-1.8]	
Allow administrators to manually customize sensor and appliance access.	
Responsibilites	**Collaborators**
Allow administrators to manually add or remove sensors and devices	Sensor
Allow administrators to manually add or remove appliances	Appliance
Provide methods to add/remove specific sensors and appliances	Permission
Used by admin to create custom permission configurations	AdminHomeowner

Class: UserSettings [C-1.9]	
Store per-user language, timezone, and key-value settings; provide read/write utilities.	
Responsibilites	**Collaborators**
Store user ID reference and belong to specific user	User
Store language and timezone preferences as attributes	
Store key-value settings map for custom user preferences	
Provide getSetting, setSetting, deleteSetting operations for settings management	

02. CRC Card – Core System [CR-02] Class Diagram, State Diagram

Class: SafeHomeHub [C-2.1]	
Central hub managing all security and smart home operations, coordinating between managers, CloudServer and processing commands.	
Responsibilites	**Collaborators**
Process security events and generate alarm events stream	AlarmManager
Process security mode configuration commands	SecurityModeManager
Process security mode control commands (activate/deactivate)	SecurityModeManager

Process sensor control commands	SensorManager
Process appliance control commands	ApplianceManager
Process system status requests and return status information	CloudServer
Process authentication control commands	AuthenticationService
Communicate with cloud server for remote operations	CloudServer
Process various SafeHome commands	SafeHomeCommand

Class: CloudServer [C-2.2]	
Provide cloud-based services for remote access, data synchronization, and external service integration.	
Responsibilites	**Collaborators**
Authenticate users and return authentication results	AuthenticationService
Process requests from SafeHome hubs	
Report critical events to external emergency services	ExternalCallService
Synchronize hub status with cloud storage	
Log events for audit and analysis	LoggingService
Manage connections to SafeHome hub	
Store and retrieve data from database	DatabaseService
Send notifications and messages to users	MessagingService
Store and manage recordings and files	StorageService

Abstract Class: SafeHomeCommand [C-2.3]	
Base class for all command types in the SafeHome system, providing common command structure.	
Responsibilites	**Collaborators**
Store unique command identifier	
Store command type classification	
Store specific command operation	

Class: SecurityModeCommand [C-2.4]	
Represent security mode operations with optional configuration payload.	
Responsibilites	**Collaborators**
Specify target security mode	SecurityMode
Specify target security mode configuration	SecurityModeConfiguration

Class: SensorCommand [C-2.5]

Target a specific sensor for control or query operations.	
Responsibilites	Collaborators
Carry target sensor identifier	Sensor
Extend base command for sensor operations	

Class: ApplianceCommand [C-2.6]	
Target a specific appliance for control or query operations.	
Responsibilites	Collaborators
Carry target appliance identifier	Appliance
Extend base command for appliance operations	

03. CRC Card – Sensor Manager [CR-03] Class Diagram, State Diagram

Class: SensorManager (Abstract Class) [C-3.1]	
Provide common interface for sensor list and status queries.	
Responsibilites	Collaborators
Maintain list of managed sensors	Sensor
Query sensor status by sensor ID	
Provide list of all managed sensors	Sensor
Define common interface for derived sensor management classes	
Generate and handle sensor events	SensorEvent

Class: SensorActivationManager [C-3.2]	
Activate or deactivate sensors and validate their status.	
Responsibilites	Collaborators
Extend base SensorManager functionality	SensorEvent
Maintain map of active sensors	Sensor
Activate sensors by sensor ID	Sensor
Deactivate sensors for specified duration	Sensor
Validate sensor functionality and status	Sensor

Class: SensorDetectionManager [C-3.2]	
Subscribe sensors to detection loop and generate event streams.	
Responsibilites	Collaborators
Maintain list of subscribed sensors	Sensor
Extend base SensorManager functionality	SensorEvent
Subscribe sensors to detection loop	Sensor

Unsubscribe sensors from detection loop	Sensor
Start detection and generate asynchronous sensor event stream	SensorEvent
Stop detection process	

Class: SensorBypassManager [C-3.3]	
Maintain bypass list and validate bypass limits.	
Responsibilites	**Collaborators**
Extend base SensorManager functionality	SensorEvent
Maintain list of bypassed sensors	Sensor
Enforce bypass limit constraints	Sensor
Bypass specific sensors	Sensor
Validate bypass requests against limits	Sensor
Clear bypass status for sensors	Sensor

Class: Sensor (Abstract Class) [C-3.4]	
Manage sensor ID, type, and state; detect events and report status.	
Responsibilites	**Collaborators**
Manage unique sensor ID and sensor type	
Track active and bypassed state flags	
Detect and generate sensor events	SensorEvent
Report current sensor status	
Perform self-validation	
Manage own's position	Point

Class: ContactSensor [C-3.12]	
Track the open/closed state of doors and windows.	
Responsibilites	**Collaborators**
Extend sensor functionality	
Track open/closed/tamper state	
Report state changes as sensor events	SensorEvent

Class: MotionSensor [C-3.10]	
Detect motion within a specified range and trigger events.	
Responsibilites	**Collaborators**
Extend sensor functionality	
Maintain detection range parameter	
Detect and notify motion within detection range	SensorEvent

Class: SoundSensor [C-3.11]	
Manage the user identity, roles, and permissions across the SafeHome system.	
Responsibilites	Collaborators
Extend sensor functionality	
Measure and track sound level	
Trigger alarm on abnormal sound levels	SensorEvent

Class: ShockVibrationSensor [C-3.13]	
Detect vibration/shock levels and trigger intrusion alerts.	
Responsibilites	Collaborators
Extend sensor functionality	
Measure and track vibration level	
Report vibration as sensor events	SensorEvent

Class: FireSmokeSensor [C-3.14]	
Detect smoke and temperature changes to trigger fire alarms.	
Responsibilites	Collaborators
Extend sensor functionality	
Measure smoke level and temperature	
Detect smoke and temperature changes	SensorEvent

Class: COSensor [C-3.15]	
Measure carbon monoxide concentration and report hazardous levels.	
Responsibilites	Collaborators
Extend sensor functionality	
Measure and track carbon monoxide concentration level	s
Report warning when hazardous level is exceeded	SensorEvent

Class: GasSensor [C-3.16]	
Monitor combustible or harmful gas concentration.	
Responsibilites	Collaborators
Extend sensor functionality	
Measure and track gas concentration level	
Monitor combustible or harmful gas concentration	SensorEvent

Class: LeakSensor [C-3.17]	
Detect water leaks and provide signals for emergency alarm.	
Responsibilites	**Collaborators**
Extend sensor functionality	
Track wet detection status	
Detect leak presence and report status	SensorEvent

Class: SensorEvent [C-3.6]	
Store sensor event ID, sensor ID, type, and timestamp.	
Responsibilites	**Collaborators**
Store unique event ID and timestamp	
Store sensor ID and sensor type	Sensor

Class: Point [C-3.7]	
Store two dimensional position	
Responsibilites	**Collaborators**
Store two dimensional position	

04. CRC Card – Appliance Manager [CR-04] Class Diagram, State Diagram

Class: ApplianceManager [C-4.1]	
Maintain the list of managed appliances and provide status queries.	
Responsibilites	**Collaborators**
Maintain list of managed appliances	Appliance
Query appliance status by appliance ID	
Retrieve list of all managed appliances	Appliance
Manage appliances according to security mode	

Class: CameraApplianceManager [C-4.2]	
Control camera live streaming, recording, and storage integration.	
Responsibilites	**Collaborators**
Maintain list of managed cameras	Camera
Start video streams for specific cameras and users	VideoStream
Start audio streams with direction control	AudioStream

Stop active streams by stream ID	LiveStreamingService
Start and stop recording with trigger types	Recording
Activate and deactivate cameras	Camera
Manage camera appliances	Camera
Extend base ApplianceManager functionality	

Class: Recording [C-4.3]	
Store recording metadata and provide playback URL and availability status.	
Responsibilites	Collaborators
Store recording ID, camera ID, and time information	
Track recording duration and resolution	
Store trigger type that initiated recording	
Store if the recording detected motion	
Store thumbnail and video object IDs	
Provide video URL for playback	StorageService
Check if recording is available	StorageService

Class: Appliance [C-4.4]	
Define common attributes and control methods for appliances.	
Responsibilites	Collaborators
Store appliance ID, type, status, power consumption and activation of specific appliance	
Provide activate and deactivate methods	
Provide power consumption to ApplianceManager	

Class: Camera [C-4.5]	
Provide live streaming, recording, password management, and internal sensor control.	
Responsibilites	Collaborators
Track recording and live streaming status	
Store and validate password for camera access	
Control internal microphone for audio capture	Microphone
Control internal speaker for audio output	Speaker
Control internal motion sensor for motion detection	MotionSensor
Capture snapshots on demand	SoundSensor

Provide recording functionality with trigger type specification	
Provide streaming functionality	LiveStreamingService
Store live stream URL	LiveStreamUrl

Class: HomeSiren [C-4.6]	
Play alarm patterns and control volume; provide hush functionality.	
Responsibilites	Collaborators
Track volume level, activation, and alarm pattern	
Manage hush time	
Activate siren with specific pattern	
Deactivate siren	
Hush siren for specified duration	
Cancel hush state	
Set volume level	

Class: Lamp [C-4.7]	
Turn on/off and adjust brightness based on schedule or mode.	
Responsibilites	Collaborators
Track dimmed status and current brightness	
Control brightness level	

Class: Microphone [C-4.14]	
Capture audio input.	
Responsibilites	Collaborators
Capture audio input.	

Class: Speaker [C-4.10]	
Provide audio output, adjust volume	
Responsibilites	Collaborators
Control volume level	
Provide audio output	

Class: LiveStreamUrl [C-4.11]	
Manage stream URL and expiration information.	
Responsibilites	Collaborators
Store stream URL and expiration timestamp	
Verify URL validity based on expiration	

Class: LiveStreamingService [C-4.15]	
Manage active streams, check health status, and provide start/stop/query functions.	
Responsibilites	Collaborators
Maintain map of active streams	Stream
Track maximum concurrent streams limit	Stream
Start streams for specific camera, user, and stream type	
Stop streams by stream ID	Stream
Retrieve stream by stream ID	Stream
Check stream health status	

Class: Stream (Abstract Class) [C-4.16]	
Define stream ID, type, status, and provide latency calculation and stop actions.	
Responsibilites	Collaborators
Store stream ID, camera ID, and user ID	
Track stream start time and active status	
Store stream type and status	
Provide stop method for stream termination	
Calculate and return stream latency	

Class: VideoStream [C-4.17]	
Manage video quality, resolution, frame rate, and generate stream URLs.	
Responsibilites	Collaborators
Store video quality setting	
Track bitrate, FPS, and resolution	
Adjust quality dynamically	
Generate and return stream URL	LiveStreamUrl

Extend base Stream functionality	

Class: AudioStream [C-4.18]	
Control audio direction, sample rate, bitrate, and provide volume and direction adjustment.	
Responsibilites	**Collaborators**
Store audio direction (TO_CAMERA, FROM_CAMERA, BIDIRECTIONAL)	
Track sample rate, bitrate, and codec	
Adjust volume level	
Set and change audio direction	
Extend base Stream functionality	

05. CRC Card – SecurityZone Manager [CR-05] Class Diagram, State Diagram

Class: SecurityZoneManager [C-5.1]	
Manage registration and status of security zones.	
Responsibilites	**Collaborators**
Manage security zone registration and deletion	SecurityZone
Query current status of specific security zones	SecurityZone
Retrieve list of all security zones	SecurityZone

Class: SecurityZone [C-5.2]	
Maintain zone identifier, position, name, and associated sensors and appliances.	
Responsibilites	**Collaborators**
Maintain unique zone identifier and descriptive name	
Maintain list of sensors assigned to this zone	Sensor
Maintain list of appliances assigned to this zone	Appliance
Provide methods to add and remove sensors	
Provide methods to add and remove appliances	
Maintain list of points representing the boundary of this zone	Point

06. CRC Card – SecurityMode Manager [CR-06] Class Diagram, State Diagram

Class: SecurityModeManager [C-6.1]	
Manage the current security mode, arming state, and mode-specific sensor configurations.	
Responsibilites	**Collaborators**
Track current security mode	
Change securty mode	
Manages current arming state (DISARMING, ARMED, ARMING, ALARMED, ERROR)	SafeHomeHub
Store and manage multiple security mode configurations	SecurityModeConfiguration
Edit security mode configurations (exit delays, enabled sensors)	SecurityModeConfiguration

Class: SecurityModeConfiguration [C-6.2]	
Store configuration details for a specific security mode.	
Responsibilites	**Collaborators**
Define exit delay time for arming the system	
Store version identifier for configuration tracking	
Maintain list of enabled sensor IDs for the mode	
Maintain list of disabled sensor IDs for the mode	

07. CRC Card – Alarm Manager [CR-07] Class Diagram, State Diagram

Class: AlarmManager [C-7.1]	
Create, update, and handle alarm events, producing AlarmAction.	
Responsibilites	**Collaborators**
Create AlarmEvent from list of SensorEvent	SensorEvent
Handle AlarmEvent and return AlarmAction	AlarmEvent, AlarmAction
Update AlarmEvent (status/severity/verification)	AlarmEvent
Use AlarmServiceType to route handling	

Class: AlarmEvent [C-7.2]	
Store alarm identifier, associated sensor events, severity, and status.	
Responsibilites	Collaborators
Store unique alarm identifier	
Maintain list of triggering sensor events	SensorEvent
Track alarm severity level	
Track alarm progression status	

Class: AlarmAction [C-7.3]	
Represent the decision output for alarm handling (siren, messaging, logging, dispatch).	
Responsibilites	Collaborators
Provide boolean flag for siren activation	
Provide boolean flag for messaging	
Provide boolean flag for logging	
Provide boolean flag for dispatch	

08. CRC Card – Authentication Service [CR-08] Class Diagram, State Diagram

Class: AuthenticationService [C-8.1]	
Manage user authentication, registration, and account operations for the SafeHome system.	
Responsibilites	Collaborators
Register new users with email and password	CloudServer
Authenticate users through login process	AuthResult
Handle user logout and session management	AuthResult
Process password reset requests via email	CloudServer
Manage password change operations	CloudServer
Update user information and profile	User
Delete user accounts from the system	CloudServer
Request and setup two-factor authentication	CloudServer
Communicate with cloud server for authentication operations	CloudServer

Class: AuthResult [C-8.2]	
Store authentication session information and validate authentication results.	
Responsibilites	**Collaborators**
Store session ID for authenticated user session	
Store user ID of authenticated user	
Validate authentication result status	

09. CRC Card – Database Service [CR-09] Class Diagram, State Diagram

Class: DatabaseService [C-9.1]	
Provide database operations for storing, retrieving, updating, and deleting data in the SafeHome system.	
Responsibilites	**Collaborators**
Provide data persistence services to other system components	
Execute database queries and return results	Database
Insert new objects into the database	Database
Update existing objects in the database	Database
Delete objects from the database	Database
Manage database connection and operations	Database

Class: Database [C-9.2]	
Manage database connection configuration and connection parameters.	
Responsibilites	**Collaborators**
Store database type information	
Maintain database connection string	
Store database authentication credentials (username, password)	
Manage database name configuration	
Store database host and port information	
Provide database connection configuration to DatabaseService	

10. CRC Card – Storage Service [CR-10] Class Diagram, State Diagram

Class: StorageService [C-10.1]	
Perform object store, retrieval, deletion, and listing; report available space.	
Responsibilites	**Collaborators**
Use storage implementation for object operations	Storage
Track storage type (LOCAL, CLOUD)	
Store objects and return object IDs	StoredObject
Retrieve objects by object ID	Storage
Delete objects by object ID	Storage
Get available storage space	Storage
List objects with optional filtering	StorageFilter

Class: Storage (Abstract Class) [C-10.2]	
Provide common foundation for concrete storage implementations.	
Responsibilites	**Collaborators**
Provide common foundation for local and cloud storage	
Encapsulate storage-specific resource management	StoredObject
Return queried objects by storage filter	StorageFilter

Class: LocalStorage [C-10.5]	
Store, retrieve, and delete objects on local paths; monitor available space.	
Responsibilites	**Collaborators**
Extend base Storage functionality	
Store objects to local storage	StoredObject
Return queried objects by storage filter	StorageFilter
Manage local storage path	
Track available and used space	
Retrieve objects from local storage	
Delete objects from local storage	
Monitor and report available space	

Class: CloudStorage [C-10.6]
Store and sync objects in cloud buckets; manage endpoint and region settings.

Responsibilites	Collaborators
Extend base Storage functionality	
Store objects to cloud storage	StoredObject
Synchronize from local storage to cloud	LocalStorage
Manage cloud bucket name	
Store region and endpoint settings	
Retrieve objects from cloud storage	
Delete objects from cloud storage	
Get available cloud space	

Class: StoredObject [C-10.3]	
Store object metadata and data; compute full path and expiration; provide access time update.	
Responsibilites	Collaborators
Store object ID, filename, and file extension	
Store content type. file size, storage path and metadata map	
Track stored timestamp and last accessed timestamp	
Store data in byte array	
Check if object is expired based on retention days	
Update access time when accessed	

Class: StorageFilter [C-10.4]	
Define filtering conditions for stored object queries and evaluate matches.	
Responsibilites	Collaborators
Define file extension filters	
Define content type filters	
Define file size range (min and max)	
Define date range (start and end)	
Evaluate whether objects match all filter conditions	StoredObject

Class: ExportService [C-10.7]

Manage storage object downloads and share link generation; track link lifecycle and access counts.	
Responsibilites	**Collaborators**
Use storage for object access	Storage
Maintain map of active share links	ShareLink
Download objects and return as files	StoredObject
Generate share links with expiry hours	ShareLink
Revoke share links by link ID	ShareLink
Retrieve share links by link ID	ShareLink

Class: ShareLink [C-10.8]	
Store share link metadata; determine expiration and accessibility; increment and limit access counts.	
Responsibilites	**Collaborators**
Store link ID, object ID, and URL	StoredObject
Reference stored object for export	StoredObject
Determine creation and expiration timestamps	
Determine access count and maximum access limit	
Check if link is expired, or can be accessed	
Increment access count when used	

11. CRC Card – ExternalCall Service [CR-11] Class Diagram, State Diagram

Class: ExternalCallService (Abstract Class) [C-11.1]	
Maintain contact list and provide foundation for external communication.	
Responsibilites	**Collaborators**
Maintain external contact list	Contact
Provide foundation for derived services to implement external communication functions	

Class: ExternalSecurityService [C-11.2]	
Send emergency dispatch requests to external security agencies.	
Responsibilites	**Collaborators**
Maintain list of security services	Contact
Send dispatch requests to external security agencies	
Receive and store dispatch responses	

Class: IVRGateway [C-11.3]	
Initiate automated phone calls to contacts based on priority.	
Responsibilites	Collaborators
Maintain contact list for IVR calls	Contact
Initiate automated IVR guidance phone calls	
Perform calls according to contact priority	

Class: Contact [C-11.4]	
Store contact information including name, phone number, and priority.	
Responsibilites	Collaborators
Store contact name and corresponding phone number	
Store priority of contact	

12. CRC Card – Messaging Service [CR-12] Class Diagram, State Diagram

Class: MessagingService (Abstract Class) [C-12.1]	
Define common interface for message transmission.	
Responsibilites	Collaborators
Define common send method for message objects	
Sends messages	Message

Class: SMSMessagingService [C-12.2]	
Define common interface for message transmission.	
Responsibilites	Collaborators
Sends SMS messages	SMSMessage

Class: PushNotificationMessagingService [C-12.3]	
Define common interface for message transmission.	
Responsibilites	Collaborators
Sends notification messages	NotificationMessage

Class: EmailMessagingService [C-12.4]
Define common interface for message transmission.

Responsibilites	Collaborators
Sends email messages	EmailMessage

Interface: Message [C-12.5]	
Define the contract for message identification.	
Responsibilites	Collaborators
Define common attributes of message objects	
Store unique message identifier	

Class: EmailMessage [C-12.8]	
Store email recipient, subject, body, and timestamp.	
Responsibilites	Collaborators
Store email ID, recipient address, subject, body, and timestamp	

Class: SMSMessage [C-12.6]	
Maintain SMS recipient number, content, and timestamp.	
Responsibilites	Collaborators
Store message ID, phone number, content, and timestamp	

Class: NotificationMessage [C-12.7]	
Encapsulate notification content, type, and priority.	
Responsibilites	Collaborators
Store notification ID and message content	
Store optional image URL or image data	
Store the prioirity of notification	

13. CRC Card – Logging Service [CR-13] Class Diagram, State Diagram

Class: LoggingService [C-13.1]	
Record and query system events; provide log filtering and export functionality.	
Responsibilites	Collaborators
Record log entries	Log

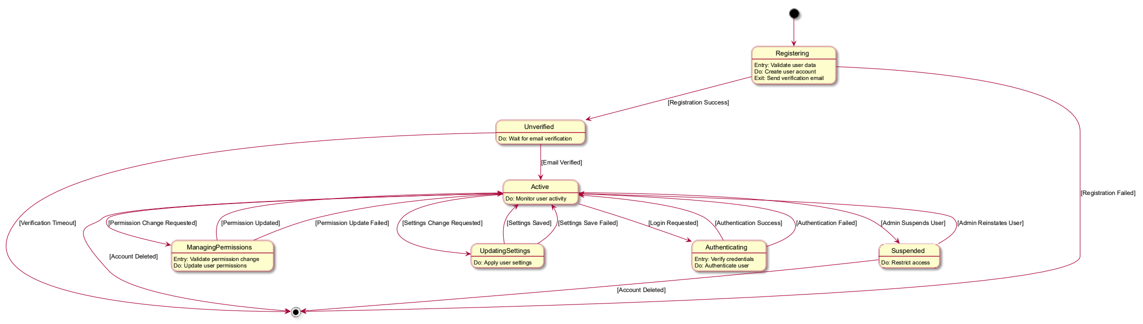
Query logs with filters	LogFilter
Export logs to file format	LogFilter
Support logging operations	

Class: LogFilter [C-13.2]	
Define log query conditions and determine if logs match criteria.	
Responsibilites	Collaborators
Create a filter with start and end date range	
Create a filter with log level range	
Create a filter with log type restrictions	
Create a filter with user, sensor, applicance ID restrictions	
Create a filter with attachted message	
Determine if logs match filter criteria	Log

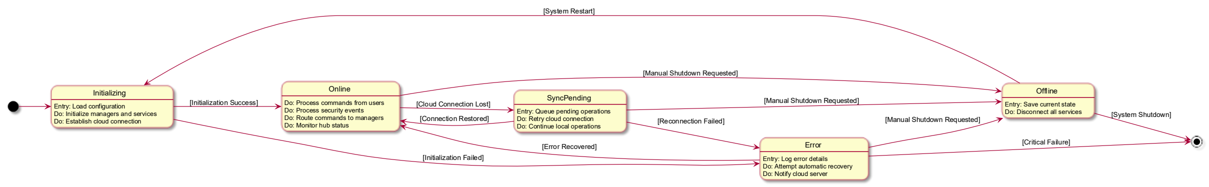
Class: Log [C-13.3]	
Store log message, level, type, and data map.	
Responsibilites	Collaborators
Store timestamp of log entry	
Store log level (severity)	
Store log message text	
Store log type (category)	
Store additional data as key-value map	

Ⅴ. State Diagram

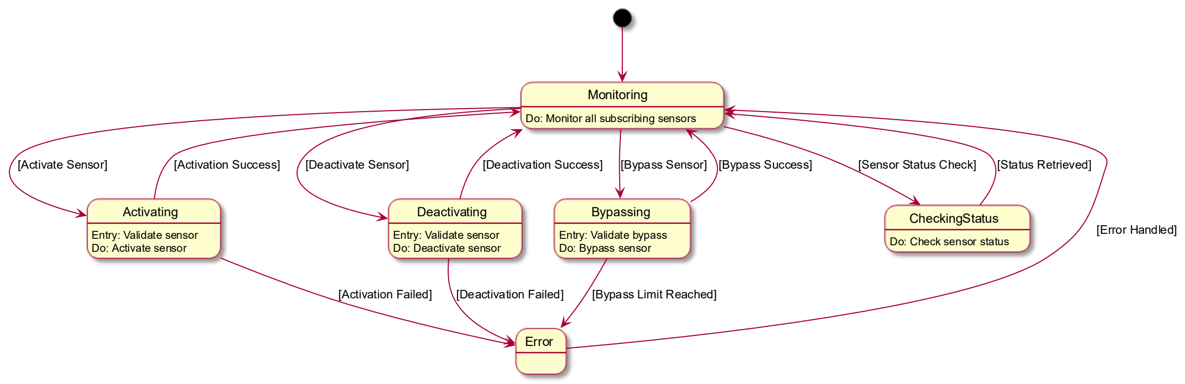
01. State Diagram – User [SD-01] Class Diagram, CRC Card



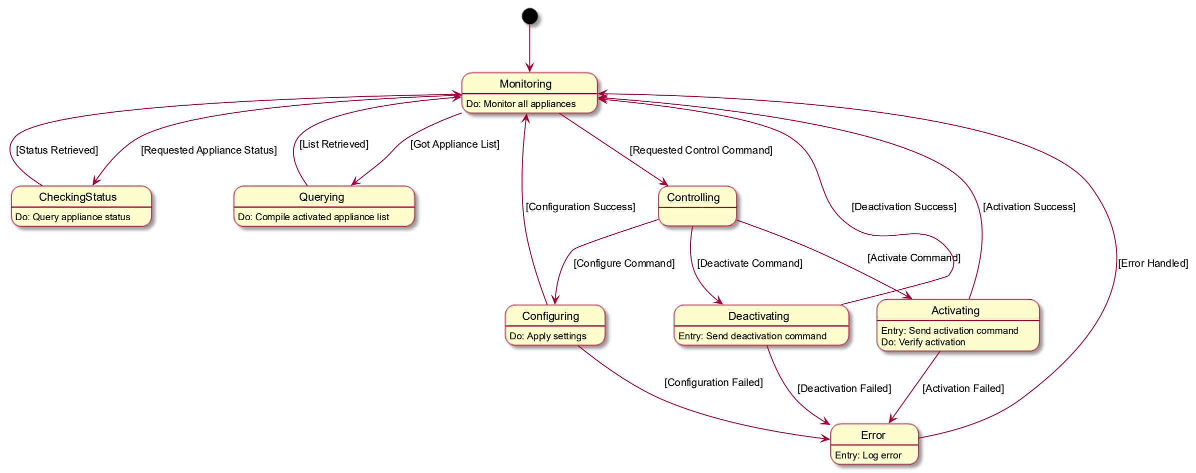
02. State Diagram – Core System [SD-02] Class Diagram, CRC Card


[System Restart]
Initializing
Entry: Load configuration
Do: Initialize managers and services
Do: Establish cloud connection
[Initialization Success]
Online
Do: Process commands from users
Do: Process security events
Do: Route commands to managers
Do: Monitor hub status
[Manual Shutdown Requested]
[Cloud Connection Lost]
[Connection Restored]
SyncPending
Entry: Queue pending operations
Do: Retry cloud connection
Do: Continue local operations
[Manual Shutdown Requested]
[Reconnection Failed]
[Error Recovered]
[Initialization Failed]
Error
Entry: Log error details
Do: Attempt automatic recovery
Do: Notify cloud server
[Manual Shutdown Requested]
Offline
Entry: Save current state
Do: Disconnect all services
[System Shutdown]
[Critical Failure]

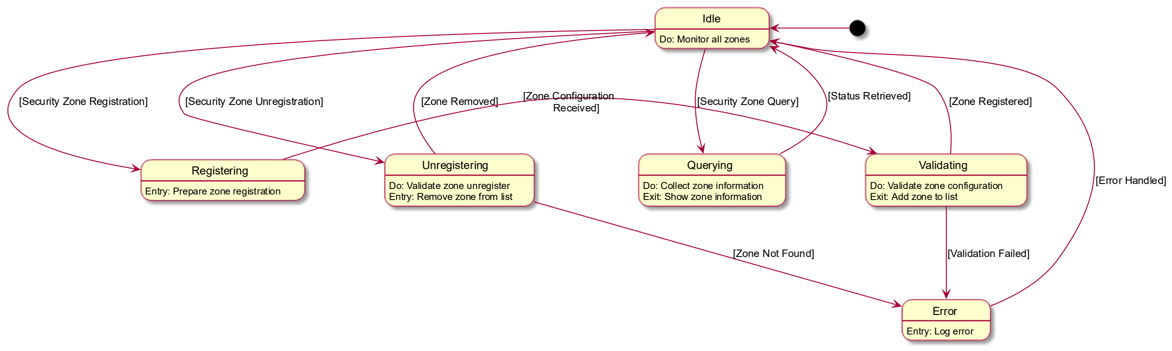
03. State Diagram – Sensor Manager [SD-03] Class Diagram, CRC Card



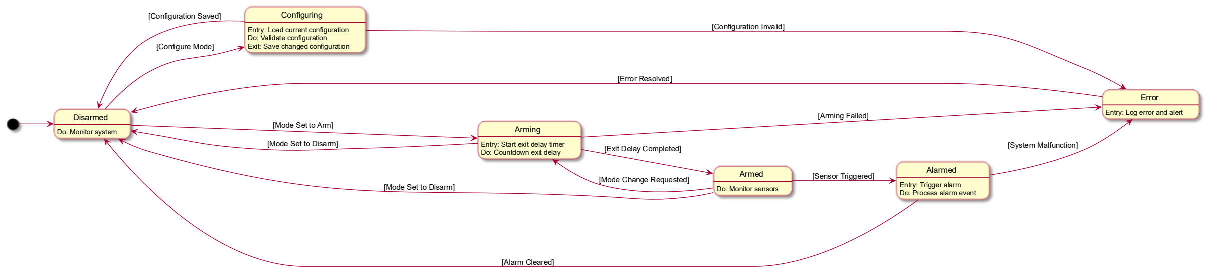
04. State Diagram – Appliance Manager [SD-04] Class Diagram, CRC Card



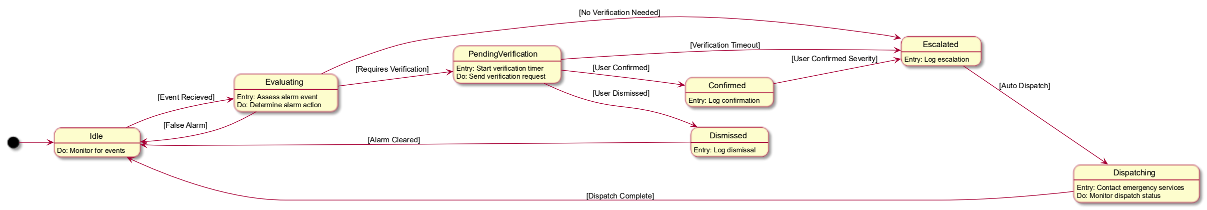
05. State Diagram – SecurityZone Manager [SD-05] Class Diagram, CRC Card


Idle
Do: Monitor all zones
[Security Zone Registration]
[Security Zone Unregistration]
[Zone Removed]
[Zone Configuration Received]
[Security Zone Query]
[Status Retrieved]
[Zone Registered]
[Error Handled]
Registering
Entry: Prepare zone registration
Unregistering
Do: Validate zone unregister
Entry: Remove zone from list
Querying
Do: Collect zone information
Exit: Show zone information
Validating
Do: Validate zone configuration
Exit: Add zone to list
[Zone Not Found]
[Validation Failed]
Error
Entry: Log error

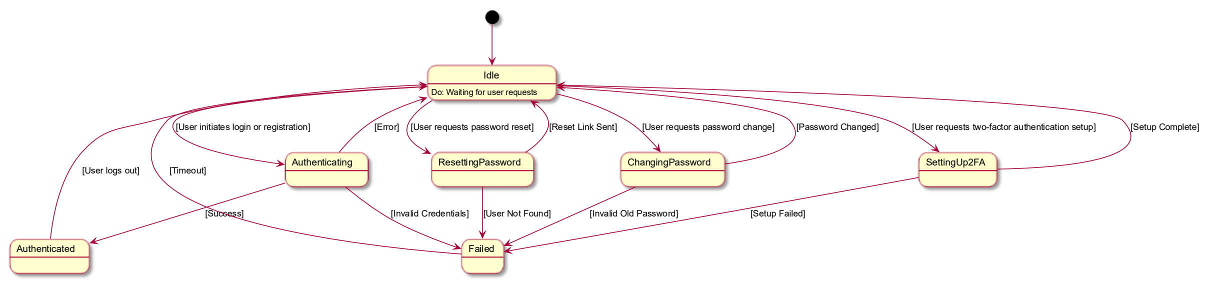
06. State Diagram – SecurityMode Manager [SD-06] Class Diagram, CRC Card



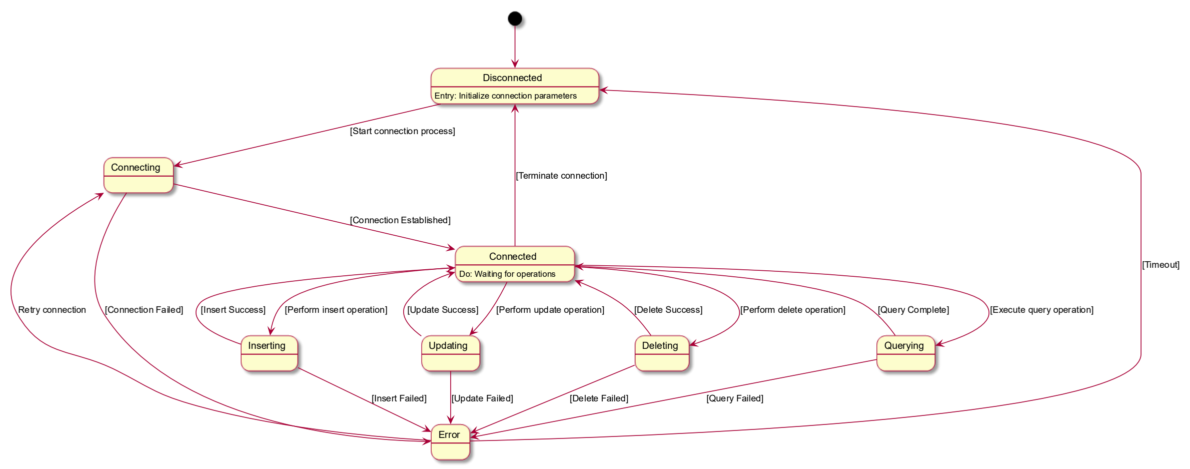
07. State Diagram – Alarm Manager [SD-07] Class Diagram, CRC Card



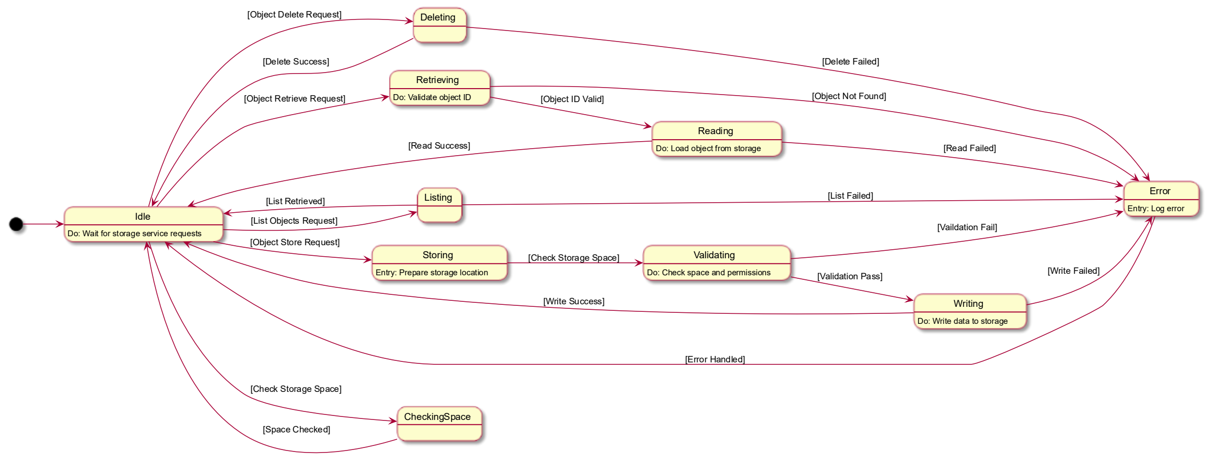
08. State Diagram – Authentication Service [SD-08] Class Diagram, CRC Card



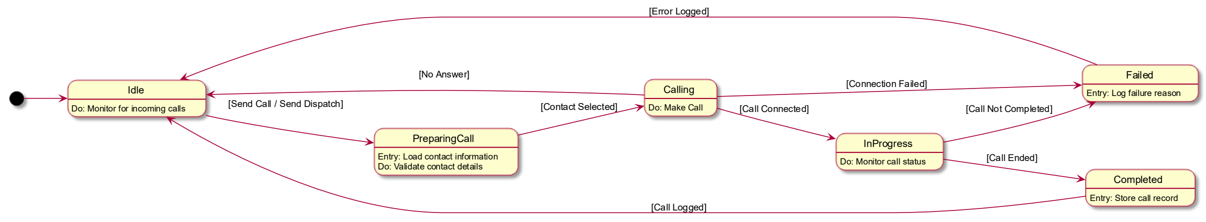
09. State Diagram – Database Service [SD-09] Class Diagram, CRC Card



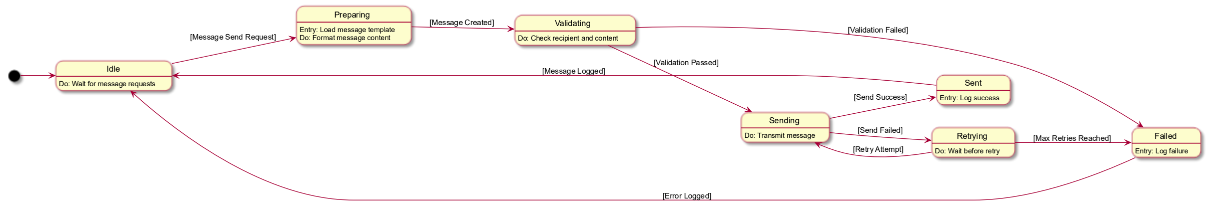
10. State Diagram – Storage Service [SD-10] Class Diagram, CRC Card



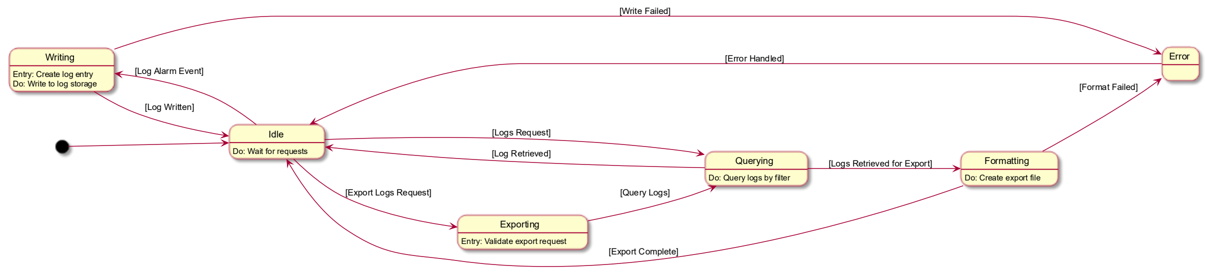
11. State Diagram – ExternalCall Service [SD-11] Class Diagram, CRC Card



12. State Diagram – Messaging Service [SD-12] Class Diagram, CRC Card



13. State Diagram – Logging Service [SD-13] Class Diagram, CRC Card



Ⅵ. Design Evaluation

1. Architecture Design Metric

Fenton's simple morphology metrics	
node	22
arc	22
size	44
depth	4
width	9
arc-to-node ratio	1.0

Table 1: Fenton's simple morphology metrics

2. CK Metrics

2.1. Calculation Table

Calculation of DIT, NOC, CBO in CK Metrics					
ID	**Packge**	**Class**	**DIT**	**NOC**	**CBO**
C-1.1	User	User	1	3	3
C-1.2		AdminHomeowner	2	0	3
C-1.3		Homeowner	2	0	3
C-1.4		Guest	2	0	3
C-1.5		Permission	1	3	2
C-1.6		FamilyMemberPermision	2	0	2
C-1.7		GuestPermission	2	0	2
C-1.8		ManualPermission	2	0	2
C-1.9		UserSettings	1	0	0
C-2.1	Core System	SafeHomeHub	1	0	9
C-2.2		CloudServer	1	0	6
C-2.3		SafeHomeCommand	1	4	0
C-2.4		SecurityModeCommand	2	0	2
C-2.5		SensorCommand	2	0	1
C-2.6		ApplianceCommand	2	0	1
C-2.7		DefaultCommand	2	0	0
C-3.1	Sensor	SensorManager	1	3	2
C-3.2		SensorActivationManager	2	0	2
C-3.3		SensorDetectionManager	2	0	2
C-3.4		SensorBypassManager	2	0	2
C-3.5		Sensor	1	2	2
C-3.6		SensorEvent	1	0	1
C-3.7		Point	1	0	0
C-3.8		IntrusionSensor	2	4	2
C-3.9		HazardSenor	2	4	2
C-3.10		MotionSensor	3	0	2
C-3.11		SoundSensor	3	0	2
C-3.12		ContactSensor	3	0	2

C-3.13		ShockVibrationSensor	3	0	2
C-3.14		FireSmokeSensor	3	0	2
C-3.15		COSensor	3	0	2
C-3.16		GasSensor	3	0	2
C-3.17		LeakSensor	3	0	2
C-4.1	Appliance	ApplianceManager	1	1	1
C-4.2		CameraApplianceManager	2	0	5
C-4.3		Recording	1	0	1
C-4.4		Appliance	1	5	0
C-4.5		Camera	2	0	6
C-4.6		HomeSiren	2	0	0
C-4.7		Lamp	2	0	0
C-4.8		GasValve	2	0	0
C-4.9		VentilationFan	2	0	0
C-4.10		Speaker	1	0	0
C-4.11		LiveStreamUrl	1	0	0
C-4.12		MotionSensor	1	0	0
C-4.13		SoundSensor	1	0	0
C-4.14		Microphone	1	0	0
C-4.15	Stream	LiveStreamingService	1	0	1
C-4.16		Stream	1	2	0
C-4.17		VideoStream	2	0	1
C-4.18		AudioStream	2	0	0
C-5.1	Security Zone	SecurityZoneManager	1	0	1
C-5.2		SecurityZone	1	0	3
C-6.1	Security Mode	SecurityModeManager	1	0	1
C-6.2		SecurityModeConfiguration	1	0	0
C-7.1	Alarm	AlarmManager	1	0	3
C-7.2		AlarmEvent	1	0	1
C-7.3		AlarmAction	1	0	0
C-8.1	Authenti-cation	AuthenticationService	1	0	3
C-8.2		AuthResult	1	0	0
C-9.1	Database	DatabaseService	1	0	1
C-9.2		Database	1	0	0

C-10.1	Storage	StorageService	1	0	3
C-10.2		Storage	1	2	2
C-10.3		StoredObject	1	0	0
C-10.4		StorageFilter	1	0	1
C-10.5		LocalStorage	2	0	2
C-10.6		CloudStorage	2	0	3
C-10.7	Export	ExportService	1	0	3
C-10.8		ShareLink	1	0	1
C-11.1	External Call	ExternalCallService	1	2	1
C-11.2		ExternalSecurityService	2	0	1
C-11.3		IVRGateway	2	0	1
C-11.4		Contact	1	0	0
C-12.1	Message	MessagingService	1	3	1
C-12.2		SMSManagingService	2	0	1
C-12.3		PushNotificationMessagingService	2	0	1
C-12.4		EmailMessagingService	2	0	1
C-12.5		Message	1	3	0
C-12.6		SMSMessage	2	0	0
C-12.7		NotificationMessage	2	0	0
C-12.8		EmailMessage	2	0	0
C-13.1	Log	LoggingService	1	0	2
C-13.2		LogFilter	1	0	1
C-13.3		Log	1	0	0
Total		**85**	**134**	**41**	**123**
Avg			**1.5765**	**0.4824**	**1.4471**
Max			**3**	**5**	**9**

Table 2: Calculation of DIT, NOC, CBO in CK Metrics

2.2. CK Metrics

CK metrics		
Metric	**Average**	**Max**
DIT (Depth of Inheritence Tree)	1.5765	3
NOC (Number Of Children)	0.4824	5
CBO (Coupling Between Object classes)	1.4471	9

Table 3: CK metrics

3. MOOD Metric

3.1. Calculation table

Calculation of MIF, CF in MOOD Metrics					
ID	**Packge**	**Class**	**M_d**	**M_i**	**M_a**
C-1.1	User	User	0	0	0
C-1.2		AdminHomeowner	0	0	0
C-1.3		Homeowner	0	0	0
C-1.4		Guest	0	0	0
C-1.5		Permission	3	0	3
C-1.6		FamilyMemberPermision	0	3	3
C-1.7		GuestPermission	0	3	3
C-1.8		ManualPermission	4	3	7
C-1.9		UserSettings	3	0	3
C-2.1	Core System	SafeHomeHub	9	0	9
C-2.2		CloudServer	9	0	9
C-2.3		SafeHomeCommand	0	0	0
C-2.4		SecurityModeCommand	0	0	0
C-2.5		SensorCommand	0	0	0
C-2.6		ApplianceCommand	0	0	0
C-2.7		DefaultCommand	0	0	0
C-3.1	Sensors	SensorManager	2	0	2
C-3.2		SensorActivationManager	3	2	5
C-3.3		SensorDetectionManager	4	2	6
C-3.4		SensorBypassManager	3	2	5
C-3.5		Sensor	3	0	3
C-3.6		SensorEvent	0	0	0
C-3.7		Point	0	0	0
C-3.8		IntrusionSensor	0	3	3
C-3.9		HazardSenor	0	3	3
C-3.10		MotionSensor	0	3	3
C-3.11		SoundSensor	0	3	3
C-3.12		ContactSensor	0	3	3

C-3.13		ShockVibrationSensor	0	3	3
C-3.14		FireSmokeSensor	0	3	3
C-3.15		COSensor	0	3	3
C-3.16		GasSensor	0	3	3
C-3.17		LeakSensor	0	3	3
C-4.1	Appliance	ApplianceManager	2	0	2
C-4.2		CameraApplianceManager	7	2	9
C-4.3		Recording	2	0	2
C-4.4		Appliance	4	0	4
C-4.5		Camera	8	4	12
C-4.6		HomeSiren	6	4	10
C-4.7		Lamp	2	4	6
C-4.8		GasValve	0	4	4
C-4.9		VentilationFan	0	4	4
C-4.10		Speaker	0	0	0
C-4.11		LiveStreamUrl	1	0	1
C-4.12		MotionSensor	0	0	0
C-4.13		SoundSensor	0	0	0
C-4.14		Microphone	0	0	0
C-4.15	Stream	LiveStreamingService	4	0	4
C-4.16		Stream	2	0	2
C-4.17		VideoStream	2	2	4
C-4.18		AudioStream	2	2	4
C-5.1	Security Zone	SecurityZoneManager	4	0	4
C-5.2		SecurityZone	6	0	6
C-6.1	Security Mode	SecurityModeManager	3	0	3
C-6.2		SecurityModeConfiguration	0	0	0
C-7.1	Alarm	AlarmManager	3	0	3
C-7.2		AlarmEvent	0	0	0
C-7.3		AlarmAction	0	0	0
C-8.1	Authenti-cation	AuthenticationService	8	0	8
C-8.2		AuthResult	1	0	1
C-9.1	Database	DatabaseService	4	0	4
C-9.2		Database	0	0	0

C-10.1	Storage	StorageService	5	0	5
C-10.2		Storage	0	0	0
C-10.3		StoredObject	3	0	3
C-10.4		StorageFilter	1	0	1
C-10.5		LocalStorage	5	0	5
C-10.6		CloudStorage	5	0	5
C-10.7	Export	ExportService	4	0	4
C-10.8		ShareLink	3	0	3
C-11.1	External Call	ExternalCallService	0	0	0
C-11.2		ExternalSecurityService	3	0	3
C-11.3		IVRGateway	1	0	1
C-11.4		Contact	0	0	0
C-12.1	Message	MessagingService	1	0	1
C-12.2		SMSManagingService	1	0	1
C-12.3		PushNotificationMessagingService	1	0	1
C-12.4		EmailMessagingService	1	0	1
C-12.5		Message	0	0	0
C-12.6		SMSMessage	0	0	0
C-12.7		NotificationMessage	0	0	0
C-12.8		EmailMessage	0	0	0
C-13.1	Log	LoggingService	4	0	4
C-13.2		LogFilter	1	0	1
C-13.3		Log	0	0	0
Total	**15**	**85**	**153**	**71**	**224**

Table 4: Calculation of MIF, CF in MOOD Metrics

3.2. MOOD Metrics

MOOD metrics		
Metric	**Calculation**	**Value**
MIF	$MIF = \frac{\sum M_i}{\sum M_a} = \frac{71}{224} = 0.3170$	0.3170
CF	$CF = \frac{\sum CBO_i}{N(N-1)} = \frac{123}{85 \times 84} = 0.01723$	0.01723

Table 5: MOOD metrics

Ⅶ. Who Did What

1. Minjun Choe

Minjun Choe
- Refine overall class diagram with team members. - CRC cards ■ Edit CRC diagram (User, Alarm, Sensor, ExternalCallService) ■ Check all CRC cards are matched with the overall system diagram. ■ Reduce redundancy between parent and child classes and clarify their relationships. ■ Establish and continuously refining CRC card specific criteria through ongoing discussions with team members. - Make Simple overall class diagram from overall class diagram.

2. Daegyu Sung

Daegyu Sung
- Wrote key sections of the SDS document, including Overview, Introduction (Goal, Design Process), Architectural Structure, Class Diagram, selected CRC Cards (~10), and the State Diagram. - Led the overall architecture design and creation of the comprehensive class diagram, continuously refining the design through team discussions and feedback. - Managed the GitHub repository, ensuring seamless integration, version control, and collaborative workflow. - Introduced and Utilized Cursor Agent to efficiently generate and maintain PlantUML (.puml) diagrams. [ME-01]

3. Songho Cho

Songho Cho
- Suggested first iteration of architectural design (hub-centric). [ME-01] - Revised overall class diagram according to team 1's SRS document. [ME-02] - Wrote the draft of state diagram with Minkyeong. [ME-03] - Revised the state diagram to ensure it matches with the class diagram - Revised CRC diagram to ensure it matches with the class diagram and state diagram. - Calculated metric of our design according to CRC cards and class diagrms. Wrote the Design Evaluation part of this document.

4. Minkyeong Kim

Minkyeong Kim
1. Class diagram a. Revising overall class diagram (done by all members) 2. CRC card

a. CRC card (all) editing based on changed class diagram

b. CRC card refinement (editing responsibilities, deleting unneeded collaborators) for storage, messaging, logging services.

3. State diagram

a. Revising state diagram (3,4,7,8,10,11)

4. Document writing

a. Writing assumptions

b. Writing CRC cards 11~13

c. Editing glossary based on Team 1 SRS

d. Writing meeting logs 1~4, uploading full meeting logs

Ⅷ.Meeting Log

This section provides a summary of team meetings conducted throughout the project lifecycle, including major design discussions, decision records, and issue resolutions. For complete meeting transcripts and detailed discussion notes, please refer to the external document available at the following location: [https://drive.google.com/drive/folders/1TreunUzTMWlf7mETwbFSw7mH9ai_5gq1?usp=sharing]

1. Meeting 1 [ME-01]

Meeting 1	
Date/Time	2025/11/06, 19:00 ~ 21:00
Place	문화관 콜라보레이션 룸 4J
Participants	Minjun Choi, Daegyu Sung, Minkyeong Kim, Songho Cho
Objective	Establish an overall plan for DDS
Discussion note	1. Tools - Use of large language models (LLMs) to obtain initial drafts of the class diagram and state diagram. - Use of PlantUML (VS Code extension) for diagram creation. - Investigation of available tools for calculating software metrics. 2. Re-configuring SRS based on Team1 document

	- Use Case 1.2.2 updated so that cancelling from either the control panel or the web interface cancels the alarm. - Identification of mismatches between Team 1's SRS assumptions and the SafeHome dialog. – Decision to implement the SDS solely based on Team 1's SRS, ignoring the SafeHome dialog. - Clarification that each sensor belongs to at most one safety zone, represented as 0..1 to 0..* in the class diagram. - Modify only the conflicting requirements with Team1, make new document with conflicts modified. • Sensor cool down (UC 1.1.3) • Alarm verification (notifications sent both inside the house and to the panel for 60 seconds) (UC 1.2.2) - For document, include a dedicated section listing modifications from the SRS (Appendix B)
Decisions	- LLMs will be used to generate draft class diagrams, with reduction of classes than first version draft. - SDS will be based only on Team 1's SRS, excluding the SafeHome dialog. Edits limited to conflicts.
What to do	**1. Before next meeting** - Prepare class diagrams for Major Functionality 1 individually and ensure understanding by the assigned deadline. - Align on the abstraction level for the class diagram; generate the complete diagram after alignment. - Understand evaluation metrics. - Calculate metrics for the generated diagrams and refine them until satisfactory values are achieved. **2. After next meeting** - Review the finalized diagram collectively – Make sure nothing is left out. - – Begin DDS documentation tasks promptly.
Next Meeting	11/8 (Sat), 20:00 via Zoom

2. Meeting 2 [ME-02]

Meeting 1

Date/Time	2025/11/06, 19:00 ~ 21:00
Place	문화관 콜라보레이션 룸 4J
Participants	Minjun Choi, Daegyu Sung, Minkyeong Kim, Songho Cho
Objective	Establish an overall plan for DDS
Discussion note	**1. Tools** - Use of large language models (LLMs) to obtain initial drafts of the class diagram and state diagram. - Use of PlantUML (VS Code extension) for diagram creation. - Investigation of available tools for calculating software metrics. **2. Re-configuring SRS based on Team1 document** - Use Case 1.2.2 updated so that cancelling from either the control panel or the web interface cancels the alarm. - Identification of mismatches between Team 1's SRS assumptions and the SafeHome dialog. – Decision to implement the SDS solely based on Team 1's SRS, ignoring the SafeHome dialog. - Clarification that each sensor belongs to at most one safety zone, represented as 0..1 to 0..* in the class diagram. - Modify only the conflicting requirements with Team1, make new document with conflicts modified. • Sensor cool down (UC 1.1.3) • Alarm verification (notifications sent both inside the house and to the panel for 60 seconds) (UC 1.2.2) - For document, include a dedicated section listing modifications from the SRS (Appendix B)
Decisions	- LLMs will be used to generate draft class diagrams, with reduction of classes than first version draft. - SDS will be based only on Team 1's SRS, excluding the SafeHome dialog. Edits limited to conflicts.
What to do	**1. Before next meeting** - Prepare class diagrams for Major Functionality 1 individually and ensure understanding by the assigned deadline.

	- Align on the abstraction level for the class diagram; generate the complete diagram after alignment. - Understand evaluation metrics. - Calculate metrics for the generated diagrams and refine them until satisfactory values are achieved. **2. After next meeting** - Review the finalized diagram collectively – Make sure nothing is left out. - – Begin DDS documentation tasks promptly.
Next Meeting	11/8 (Sat), 20:00 via Zoom

3. Meeting 3 [ME-03]

Meeting 3	
Date/Time	2025/11/10, 14:30-17:00
Place	도서관 그룹스터디룸 3C
Participants	Minjun Choi, Daegyu Sung, Minkyeong Kim, Songho Cho
Objective	- Complete class diagrams for Major Functionality 2-5 - Establish criteria for state diagram development
Discussion note	Couldn't finish revision for draft class diagrams (functionality 2-5) before meeting. Started from revision of functionalities and making overall diagram. **1. Class & Manager structure** - Security Zone and Security Zone Manager classes were introduced to handle all security-zone-related operations (not in SRS) - For UC 3.2.1 System Status Dashboard, no new class will be created; the Hub will directly access Sensor Manager and Appliance Manager. - Logging Manager will manage UC 3.2.2 Activity Log, and both Export and Filter functionalities will be added. - Appliance Manager will reflect newly added appliances: Lamp, Smart Gas Valve, Ventilation for UC 5. Indoor Monitoring and Device Control. - UC 3.1 Register/Remove functionality is removed from Appliance Manager, with the assumption that floor plan is consistent.

	2. Authentication & Permission - A new Authentication Service Class is created to handle UC 4.1 Login/Auth, without modifying existing Hub/Server, enabling a clear Layer Design. - UC 4.1.1 Register functionality will be internally implemented within Authentication Service; no separate class for Unauthenticated User will be created. - An Auth Database Class will be placed next to the Auth class to store and manage authentication data such as user login information, mentioned in 4.1.2 - UC 4.1.7 (2FA authentication) will be implemented as part of the Auth Class and integrated with Messaging Service. - A new Permission Class for UC 3.3 User and permission management will be created and associated with each User, editable by Admin and Homeowner only. **3. Device & Sensor design** - Both Appliance and Sensor classes will include a Passive Power Consumption attribute. **4. System architecture & Service layer assumption** - Services referenced globally (e.g., LoggingService, DatabaseService) may not appear directly connected to leaf classes in diagrams but are assumed to be accessible via Hub and Cloud. **5. Diagram development process** - All changes were incorporated into the full system Class Diagram, and the corresponding PUML file was updated. - Work included alignment across Managers and Services, appliance additions, authentication components, and updated device/attribute definitions.
Decisions	- Functions of security zone were added from class

	- Class design was done in a way to include minor functions, such as export or filter, in existing classes. - An overall diagram was made first, and it will be divided for presentation/documentation according to functionalities.
What to do	**1. Revision of current work** - The overall class diagram will be revised by each member until 11/10, through review of SRS and comparison of each class with use cases. - The final version will be produced by 11/11. **2. Before next meeting** - Overall class diagram will be divided in manager and service levels (Tue) - Make CRC Card drafts (Tue) - Calculate metrics (Tue) - Make state diagram drafts (Wed) **3. At/after next meeting** - Consider the level of state diagram - Start writing documents (Thu) - Final revision (Fri) · Meeting Log revision · Filling out traceability · Writing how the design work proceeded · Filling out Glossary
Next Meeting	11/12 (Wed), 14:30 at library

4. Meeting 4 [ME-04]

Meeting 4	
Date/Time	2025/11/12, 14:30~15:30
Place	도서관 그룹스터디룸 3D
Participants	Minjun Choi, Daegyu Sung, Minkyeong Kim, Songho Cho
Objective	- Define scope and granularity for state diagram development - Review CRC cards and establish revision guidelines - Review overall architecture diagram
Discussion note	**1. Overall system architecture**

	- A Point class (x, y coordinates) was added to Sensor and Security Zone for representing spatial positioning. - The architectural diagram will be drawn directly using Figma. **2. Missing Relationships** - Missing relationships in the simple overall diagram were identified; leaf nodes connected to other class packages must be added manually. - Currently identified missing relations: • SecurityZone -> Sensor • SecurityZone -> Appliance • ApplianceManager → Appliance • Camera → Appliance (inheritance) • SecurityZoneManager → Sensor • SecurityZoneManager → Appliance • MotionSensor → Sensor (inheritance) • SoundSensor → Sensor (inheritance) • Camera → MotionSensor • Camera → SoundSensor • SensorEvent → AlarmEvent (inheritance) • AlarmManager → AlarmEvent • Sensor → SensorEvent - ExportService is tightly integrated with StorageService, so will be drawn together in a divided class diagram. **3. State Diagram Scope & Approach** - A simple and format-aligned version will be generated using LLM based on clarified guidelines. - Final scope for state diagrams includes 17 components: • Managers (5): SecurityZone, SecurityMode, Appliance, Sensor, Alarm • Services (8): Authentication, Database, ExternalCall, Export, Storage,Messaging, Logging, LiveStreamingService • Additional classes (4): Core system, Users, camera, sensor

	- This classification will also be used for class diagram presentation and CRC cards (CRC cards discluding few additional classes) **4. CRC Card Guidelines & Revisions** - General CRC Rules · No CRC cards will be created for enum types. · If a CRC card exists for an abstract class, a card for abstract class is always made. Child class cards are only made if their function vary. · - Only include classes that are directly connected with the class as collaborators.
Decisions	- Classification for CRC cards, state diagrams, class diagrams were made; 5 managers, 8 services, 3 other main classes. - Make state diagrams with simple, important states only. Start with 'idle' state normally. Do not use function names for state changes.
What to do	- Architectural diagram (Wed) – Daegyu - Class diagram (Wed) Name/divide files according to CRC numbering - Minjun - State diagram review (Thu) -Minkyeong, Songho - CRC review (Thu) · User – Minjun · Core system – Daegyu · Sensor Manager – Minjun · Appliance Manager – Songho · SecurityZone Manager – Songho · SecurityMode Manager – Songho · Alarm Manager – Minjun · Authentication Service – Daegyu · Database Service – Daegyu · Storage Service (with ExportService) – Minkyeong · ExternalCall Service – Minjun · Messaging Service – Minkyeong · Logging Service – Minkyeong - Document (Thu-Fri)

	- Formatting, putting together by Daegyu - Final review (Fri) · Meeting logs (Minkyeong) · Traceability check () · Write “How the design work proceeded” · Organize glossary, hyperlinks
Next Meeting	N/A

Appendix A. Glossary

This glossary defines key terminology used throughout the SafeHome Software Requirements Specification. Terms are organized by category for easy reference.

System Components (Hardware)

SafeHome Hub

The central control unit that manages all connected devices, processes sensor data, executes automation rules, and communicates with the cloud server. Located within the home, it operates as the primary coordinator for all system functions.

- Related Use Cases: All system functions
- Reference: Introduction, Section II

Control Panel

A physical touchscreen device installed in the home that provides local access to system functions. Users can arm/disarm the system, view camera feeds, and manage basic settings without requiring internet connectivity.

- Related Use Cases: UC 1.3.1 (Set System Mode), UC 4.1.2 (Log In)
- Reference: Fig. 1, Page 6

Sensor

A wireless device that monitors specific conditions or events. SafeHome supports multiple sensor types:

- Contact Sensor: Detects opening/closing of doors and windows
- Motion Sensor: Detects movement in designated areas
- Environmental Sensor: Monitors fire, smoke, carbon monoxide, gas leaks, or water leaks
- Air Quality Sensor: Measures indoor pollutants (CO_2 , VOCs)
- Sound Sensor: Detects specific audio patterns (e.g., dog barking, glass breaking)
- Related Use Cases: UC 1.1 (Sensor Monitoring)
- Reference: Section 1.1, Major Functionalities

Camera (IP Camera)

A network-connected video camera that provides live streaming, recording, and two-way audio capabilities. Cameras can be placed indoors or outdoors and support features such as pan, tilt, zoom (PTZ), night vision, and motion-triggered recording.

- Related Use Cases: UC 2.1 (Camera Viewing and Control)
- Reference: Section 2.1, Major Functionalities

Smart Home Device

IoT devices that can be controlled by the SafeHome system, including smart lights, smart thermostats, smart locks, and ventilation systems. Integration enables automation based on security modes or environmental conditions.

- Related Use Cases: UC 5.1.1 (Indoor Device Control), UC 5.2.1 (Air Quality Monitoring)
- Reference: Section 5, Major Functionalities (Second Increment)

System Components (Software)

SafeHome Cloud Server

The cloud-based backend infrastructure that enables remote access, stores user data, manages authentication, delivers push notifications, and synchronizes system state across multiple devices.

- Related Use Cases: All remote access functions, UC 4.1 (Account Management)
- Reference: Architecture overview in all Sequence Diagrams

Mobile Application

The primary user interface for SafeHome, available on iOS and Android platforms. Provides comprehensive access to all system functions including live camera viewing, security control, device management, and notifications.

- Related Use Cases: All user-facing use cases
- Reference: Section III (Prototype UI/UX)

Web Interface

A browser-based application that offers functionality equivalent to the mobile app, accessible from desktop or laptop computers. Useful for detailed configuration and system administration tasks.

- Related Use Cases: All user-facing use cases
- Reference: UC 4.1.2 (Log In)

Security Concepts

Security Mode

A predefined system configuration that determines which sensors are active and how the system responds to events. SafeHome supports the following modes:

- Home Mode: Interior motion sensors disabled, perimeter sensors active
- Away Mode: All sensors active, full security monitoring
- Sleep Mode: Perimeter sensors active, interior sensors partially active
- Overnight Travel / Extended Travel: Enhanced security with simulated occupancy features
- Related Use Cases: UC 1.3.1 (Set System Mode)
- Reference: Dialog slide 9, UC 1.3.1

Alarm Condition

An event detected by a sensor that requires immediate attention, such as unauthorized entry, environmental hazard, or system tampering. Alarm conditions trigger notifications, sirens, and potentially emergency service dispatch.

- Related Use Cases: UC 1.2.1 (Configure Alarm Conditions), UC 1.2.3 (Emergency Service Integration)
- Reference: UC 1.1, UC 1.2

Security Zone

A defined zone within a home that groups sensor and appliances together for coordinated monitoring, control, and security management. Each zone maintains its own configuration, status, and associated devices. Each zone's security mode can be managed seperately.

- Reference: HW2 Safehome_design_guideline

Sensor Bypass

A temporary exclusion of a faulted sensor from monitoring to allow system arming when that sensor cannot be secured (e.g., an open window). Bypassed sensors are excluded only for the current armed session and automatically re-enabled upon disarming.

- Related Use Cases: UC 1.3.2 (Sensor Bypass)
- Reference: Dialog slide 10, UC 1.3.2

Panic Button

An emergency feature accessible from the mobile app that immediately activates the highest-priority alarm, bypassing all verification steps. Used by homeowners in perceived emergency situations.

- Related Use Cases: UC 1.2.4 (Panic Button)
- Reference: UC 1.2.4

Two-Factor Authentication (2FA)

An additional security layer requiring users to provide two forms of verification during login: their password and a time-limited code from an authenticator app or SMS. Significantly reduces the risk of unauthorized account access.

- Related Use Cases: UC 4.1.7 (Two-Factor Authentication Management), UC 4.1.2 (Log In)
- Reference: UC 4.1.7

Exit Delay / Entry Delay

A configurable time window that allows users to leave or enter the home without triggering an alarm after arming the system. Typically 30-90 seconds.

- Related Use Cases: UC 1.3.1 (Set System Mode)
- Reference: UC 1.3.1, Scenario step 3a

User Roles and Access

Homeowner

The primary user of the SafeHome system with full access to all functions. Homeowners can arm/disarm the system, view all cameras, manage devices, configure settings, and invite other users.

- Related Use Cases: All use cases
- Reference: All use case Primary Actor fields

Admin Homeowner

A homeowner with elevated privileges who can modify system settings, manage other user accounts, add or remove devices, and configure security policies. At least one admin account must exist per system.

- Related Use Cases: UC 3.1.1 (Add and Configure New Devices), UC 3.3.1 (User Role and Access Control), UC 4.1.6 (Change Password)
- Reference: UC 3.3.1, UC 4.1.7

Guest

A limited-privilege user granted temporary or restricted access to specific system functions. Guests may be able to arm/disarm the system or view certain cameras but cannot modify settings or manage devices.

- Related Use Cases: UC 1.3.1 (Set System Mode), UC 4.1.2 (Log In)
- Reference: UC 3.3.1 (permission templates)

System Administrator (Software)

Not a human role, but the automated SafeHome system software that manages device communication, executes automation rules, and maintains system integrity.

- Related Use Cases: Referenced as secondary actor in multiple use cases
- Reference: Assumptions section

Operational Modes and States

Armed / Disarmed

The security state of the system:

- Armed: Sensors are actively monitoring for intrusion events
- Disarmed: Sensors continue operating but do not trigger alarms
- Related Use Cases: UC 1.3.1 (Set System Mode)
- Reference: UC 1.3.1

Alarm Verification

A workflow that requires homeowner confirmation before full alarm escalation. When a non-critical sensor event occurs, the system captures visual evidence and prompts the homeowner to confirm whether it is a genuine threat or false alarm.

- Related Use Cases: UC 1.2.2 (Alarm Verification Step)
- Reference: UC 1.2.2

Cooldown Period

A time interval after an event notification during which additional notifications of the same type are suppressed to prevent alert fatigue. For example, a motion-detected camera may only send one notification every 5 minutes.

- Related Use Cases: UC 1.1.3 (Outdoor Motion Detection), UC 2.3.2 (Notification Policy and Cooldown)
- Reference: UC 2.3.2

Activity Log / Timeline

A chronological record of all system events, including sensor activations, user actions, device state changes, and system alerts. Provides an audit trail for security review and incident investigation.

- Related Use Cases: UC 3.2.2 (Activity Logs and Timeline)
- Reference: Dialog slide 39, UC 3.2.2

Recording and Media

Live View

Real-time video streaming from a camera to the user's device with minimal latency (typically < 5 seconds). Enables users to observe their home remotely.

- Related Use Cases: UC 2.1.1 (Single Camera Live View)
- Reference: Dialog slides 29-31, UC 2.1.1

Recording

Captured video footage stored locally on the hub or in cloud storage. SafeHome supports:

- Continuous Recording: 24/7 video capture
- Event Recording: Motion-triggered or alarm-triggered capture with pre-roll and post-roll buffers
- Related Use Cases: UC 2.3.1 (Recording Settings), UC 2.2.1 (Search and Playback Recordings)
- Reference: UC 2.3.1

Camera Lock / Password Protection

A security feature that requires a specific password to access a camera's live view or recordings, providing additional privacy for sensitive areas.

- Related Use Cases: UC 2.1.3 (Protect Sensitive Camera Feed with a Password)
- Reference: Dialog slides 19-31, UC 2.1.3

Two-Way Audio

Bidirectional voice communication through a camera's built-in speaker and microphone, allowing the homeowner to speak to people at the camera's location in real-time.

- Related Use Cases: UC 2.1.2 (Two-Way Audio)
- Reference: Dialog slide 16, UC 2.1.2

Evidence Export

The ability to download or securely share recorded video clips or snapshots for purposes such as reporting to authorities, insurance claims, or personal backup.

- Related Use Cases: UC 2.2.2 (Evidence Sharing and Export)
- Reference: UC 2.2.2

Device Management

Device Registration / Pairing

The process of adding a new sensor or camera to the SafeHome system. Involves discovery, secure authentication, and initial configuration.

- Related Use Cases: UC 3.1.1 (Add and Configure New Devices)
- Reference: Dialog slide 58, UC 3.1.1

Device Status

Real-time information about a device's operational health, including:

- Online / Offline: Network connectivity state
- Battery Level: Remaining power for battery-operated devices
- Signal Strength: Quality of wireless connection to the hub
- Related Use Cases: UC 3.2.1 (System Status Dashboard)
- Reference: UC 3.2.1

Sensor Activation / Deactivation

The ability to enable or disable a sensor's monitoring function for a specified duration or indefinitely. Deactivated sensors remain registered but do not trigger events.

- Related Use Cases: UC 1.3.3 (Sensor Activation and Deactivation), UC 2.1.4 (Camera Activation and Deactivation)
- Reference: Meeting 2025.10.26/29, UC 1.3.3, UC 2.1.4

Technical Terms

Push Notification

An instant alert message sent to the user's mobile device by the SafeHome cloud server. Used for security events, system status updates, and reminders.

- Related Use Cases: UC 1.2.1 (Configure Alarm Conditions), UC 1.1.3 (Outdoor Motion Detection)
- Reference: Multiple use cases in Section 1 and 2

One-Time Password (OTP)

A time-limited verification code sent via SMS or email, used for account verification, password reset, and two-factor authentication.

- Related Use Cases: UC 4.1.1 (Sign Up), UC 4.1.4 (Reset Password), UC 4.1.7 (2FA Management)
- Reference: UC 4.1.1, UC 4.1.7

Session Token

A secure credential issued by the cloud server after successful authentication that grants the user access to the system for a limited time without requiring re-login.

- Related Use Cases: UC 4.1.2 (Log In), UC 4.1.3 (Log Out)
- Reference: UC 4.1.2

End-to-End Encryption (E2EE)

A security protocol that encrypts data at the source (e.g., camera) and decrypts it only at the destination (e.g., user's app), preventing intermediate parties from accessing the content.

- Related Use Cases: UC 2.1.1 (Single Camera Live View)
- Reference: Goal section, Non-functional requirements

Audit Log

A detailed, tamper-evident record of all administrative actions and security-sensitive operations, including who performed the action, when, and from which device. Used for security compliance and forensic analysis.

- Related Use Cases: UC 3.2.2 (Activity Logs and Timeline), UC 4.1.1 (Sign Up)
- Reference: Multiple use cases across all sections

Idempotent Request

A request that can be safely repeated multiple times without causing duplicate or unintended effects. SafeHome uses idempotent requests with unique IDs to prevent double-processing during network retries.

- Related Use Cases: UC 4.1.1 (Sign Up), UC 1.2.1 (Configure Alarm Conditions)
- Reference: UC 4.1.1, UC 1.2.1 Exception handling

Environmental Monitoring (Second Increment)

Indoor Air Quality (IAQ)

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

- Related Use Cases: UC 5.2.1 (Indoor Air Quality Monitoring)
- Reference: Dialog slide 27, UC 5.2.1 (Second Increment)

Smart Meter

A device that monitors real-time electrical power consumption, enabling energy usage tracking and analysis.

- Related Use Cases: UC 5.2.2 (Real-Time Power Consumption Monitoring)
- Reference: Dialog slide 29, UC 5.2.2 (Second Increment)

Appendix B. Modification

This appendix documents all design modifications and requirement deviations made after the publication of the SRS. Each subsection describes the corresponding use case, the reason for change, and the impact on system behavior. These updates were introduced during the detailed design phase to improve security, reliability, and usability while maintaining consistency with the original project objectives.

B.1 Outdoor Motion Detection – related with UC1.1.3

The cooldown feature was removed due to its significant drawback: it created a critical security vulnerability with an all-too-obvious potential for abuse.
For example, an intruder could trigger the sensor using an ordinary object like a rock, then exploit the cooldown period to break in immediately after the user dismissed it as nothing serious.

References on [ME-01]

B.2 Alarm Verification Step – related with UC1.2.2

The fact that alarm verification cannot be performed when the user is not accessing the web browser, coupled with the existence of a 60-second grace period even when inside the home, appears to be a critical security flaw. Therefore, even during the 60-second verification request period, a Level 1 alarm should sound inside the home to alert the user to the trigger situation.

References on [ME-01]