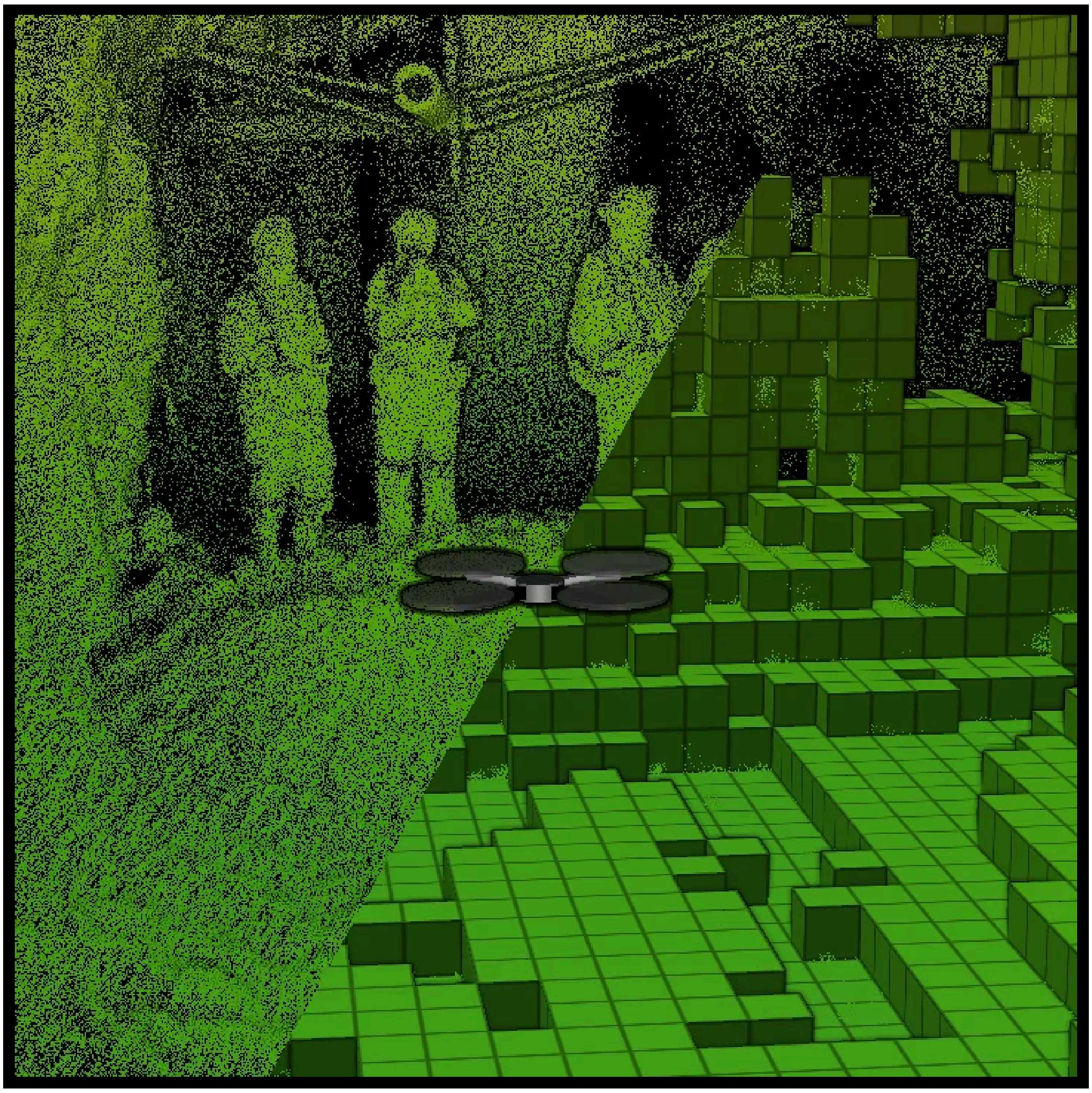
Robotics Labs Documentation



Thanondrak Arunsangsirinak

Table of Contents

Definitions, Acronyms, and Abbreviations

Dock	A dock is a piece of hardware, similar to a Roomba charging port, whose purpose is to provide drones with the ability to charge their power on the dock. Furthermore, it provides localisation of the drone by either GPS or some type of sensor. It serves as local computing and in some cases, storage system.
Drone	A drone is a piece of hardware, capable of manoeuvring in 3 dimensions, provided with Lidar, distance, and odometry sensors.

List of Actors

Guest	Anyone who is not logged into the application system by the Robotics Labs
User	Anyone who has logged into the application system by a verified user account.
Dock System	A system for processing any computation done by the drone. It could be a dock system locally or by Robotics Labs

User Requirements

Overview

In response to the increasing demand for advanced home security solutions, we introduce a state-of-the-art Household Surveillance drone System. This innovative system leverages cutting-edge robotics and artificial intelligence technologies to provide homeowners with an intelligent and proactive approach to residential security.

The 3D Household Surveillance drone is a mobile drone equipped with advanced sensors, cameras, and 3D mapping capabilities. Unlike traditional CCTV systems, this drone can autonomously navigate through indoor spaces, offering comprehensive coverage and the ability to monitor various rooms and areas of the household.

Key Features:

Autonomous Navigation:

The drone is designed to navigate intelligently throughout the house, avoiding obstacles and efficiently covering all accessible areas.

3D Mapping:

Utilizing advanced mapping algorithms, the drone creates a three-dimensional map of the environment, enabling precise navigation and location tracking.

Multi-Sensor System:

Equipped with a variety of sensors, including cameras, motion detectors, and environmental sensors, the drone can detect and respond to a range of security-related events.

Live Streaming and Recording:

The drone captures high-definition video and audio in real-time, providing live streaming capabilities for homeowners to monitor their property remotely.

Object Recognition:

Advanced AI algorithms enable the drone to recognise and distinguish between different objects and entities, helping to filter out false alarms and focus on potential security threats.

Alert System:

In the event of suspicious activity or security breaches, the system sends instant alerts to homeowners through a dedicated mobile app, allowing for quick response and intervention.

Integration with Smart Home Ecosystems:

The surveillance drone seamlessly integrates with existing smart home systems, enabling users to control and monitor the drone alongside other connected devices.

User-Friendly Interface:

The accompanying mobile app provides an intuitive interface for users to remotely control the drone, view live feeds, and customize security settings.

Functional Requirements

Functional requirements specify the features and capabilities a system must have to meet user needs. They outline the system's expected behaviour, interactions, and use cases, guiding development, testing, and validation processes.

Autonomous Navigation:

The system must enable the drone to navigate autonomously within the household, avoiding obstacles and adjusting its path in real time.

3D Mapping and Localization:

The system should generate and update a 3D map of the indoor environment, allowing the drone to accurately localize itself and navigate based on this map.

Live Streaming:

The system must support real-time high-definition video streaming from the drone's cameras to the user's mobile app, providing live monitoring capabilities.

Recording and Storage:

Users should have the ability to initiate manual recording sessions and the system must automatically record and store footage when triggered by security events.

Mobile App Control:

The mobile app should provide users with intuitive controls for remotely operating the drone, adjusting settings, and viewing live or recorded footage.

Object Recognition:

The system must employ AI algorithms for object recognition, allowing the drone to identify and classify objects within its field of view.

Alert Generation:

An alert system should be in place to notify users immediately when the drone detects suspicious activity, triggering an alert on the mobile app.

Customizable Security Settings:

Users should be able to customise security settings such as motion detection sensitivity, notification preferences, and specific areas of focus for the drone.

Smart Home Integration:

The system must seamlessly integrate with popular smart home ecosystems, allowing users to control the surveillance drone alongside other connected devices.

Privacy Controls:

The system should provide features to address privacy concerns, such as the ability to disable cameras in certain zones or turn off monitoring during specific times.

Automated Charging:

The drone should have the capability to automatically return to a charging station when its battery is low and resume surveillance after charging.

Emergency Response Mode:

The system should include an emergency response mode allowing users to remotely guide the drone to a specific location in the house during urgent situations.

Multi-drone Collaboration:

If multiple drones are deployed, the system should support collaborative efforts, ensuring efficient coverage and communication between drones.

Integration with External Sensors:

The system should allow integration with external sensors (e.g., door/window sensors) to enhance the overall security monitoring capabilities.

User Activity Log:

The system should maintain a log of user activities, including drone movements, triggered alerts, and system changes for reference and audit purposes.

Non-Functional Requirements

Non-functional requirements define the attributes that characterize how a system should perform rather than what it should do.

Performance:

The system must achieve low-latency video streaming, with a maximum delay of X seconds, ensuring real-time monitoring capabilities.

Reliability:

The drone should operate continuously with a mean time between failures (MTBF) of at least X hours, minimizing downtime and ensuring consistent surveillance.

Scalability:

The system must support the integration of additional drones without compromising performance, allowing for scalable deployment based on user needs.

Security:

Data transmission between the drone and the mobile app must be encrypted to ensure the privacy and security of user information and surveillance footage.

Usability:

The mobile app interface should be user-friendly and intuitive, requiring minimal training for users to operate and configure the surveillance system.

Compatibility:

The system must be compatible with a wide range of smartphones and tablets, supporting both iOS and Android platforms for the mobile app.

Interoperability:

The surveillance system should be interoperable with standard smart home protocols (e.g., Zigbee, Z-Wave) and devices for seamless integration within a smart home ecosystem.

Maintainability:

The system should allow for easy software updates and maintenance, ensuring that users can benefit from new features and security patches promptly.

Power Efficiency:

The drone should optimize power consumption during normal operation and standby modes, maximizing the time between required recharging.

Environmental Adaptability:

The drone should operate effectively within a range of environmental conditions, including various lighting conditions and temperature ranges commonly found in households.

Compliance:

The system must comply with relevant privacy regulations (e.g., GDPR) and security standards to ensure legal and ethical use.

Response Time:

The mobile app should have a fast response time, allowing users to control the drone and receive alerts without significant delays.

Availability:

The surveillance system should have a high availability rate, ensuring that users can access live feeds and control features whenever needed.

Data Storage Capacity:

The system should provide ample storage capacity for recorded footage, with the ability to expand storage based on user requirements.

User Support:

The system should include a comprehensive user manual, online support resources, and a responsive customer support service to assist users with any issues or inquiries.

Constraints

Constraints represent the limitations or boundaries within which the system must operate.

Battery Life:

The drone's operational time is constrained by its battery life. Consider specifying the maximum duration the drone can operate before it needs to return for recharging.

Indoor Navigation Limitations:

The drone's autonomous navigation might be constrained by the indoor environment, including narrow spaces, low ceilings, or obstacles that might limit its movement.

Weight and Size Restrictions:

The drone must adhere to specific weight and size limitations to ensure safe and practical operation within the confined spaces of a household.

Regulatory Compliance:

The system must comply with local regulations governing the use of drones, ensuring adherence to airspace restrictions and privacy laws.

Noise Level:

The drone's noise level during operation may be constrained to avoid disturbance in a residential setting. Consider setting acceptable noise limits for the system.

Communication Range:

The drone's communication range with the mobile app and other devices may be limited by the indoor environment. Specify the acceptable range for effective operation.

Privacy Concerns:

The system must incorporate features to address privacy concerns, such as the ability to disable cameras in specific zones, and provide users control over the monitoring areas.

Cost Constraints:

The overall cost of the surveillance drone system may be constrained by budget limitations, requiring a balance between features and affordability.

Weather Conditions:

The drone's operation might be limited by adverse weather conditions. Specify the system's capability to operate in specific temperature ranges or under varying humidity levels.

Maintenance Access:

The system's design should consider the accessibility for maintenance and repairs, ensuring that users or technicians can easily access and service the drone components.

Software Compatibility:

The drone's software must be compatible with a defined set of operating systems and mobile devices, considering constraints imposed by software development resources.

Integration with Existing Infrastructure:

The system must be designed to integrate with existing smart home infrastructure and devices, but constraints may exist based on compatibility and available communication protocols.

Data Storage Limits:

Constraints on data storage capacity may exist due to cost considerations or technological limitations, requiring users to manage and archive recorded footage accordingly.

User Skill Level:

Consideration should be given to the skill level required for users to set up, operate, and troubleshoot the surveillance drone system, with constraints on complexity and technical requirements.

Legal and Ethical Considerations:

The system must operate within legal and ethical boundaries, considering constraints imposed by privacy laws, surveillance regulations, and ethical standards.

Quality Attributes

Quality attributes define the overall characteristics and qualities that the system must possess to meet the expectations and needs of its users.

Reliability:

The system should consistently perform its surveillance tasks without unexpected failures, ensuring that users can rely on it for home security.

Availability:

The surveillance drone system should be available for use whenever users need it, with minimal downtime for maintenance or charging.

Performance:

The system should provide fast response times for commands and real-time video streaming, ensuring a smooth and responsive user experience.

Scalability:

The system should be scalable to accommodate additional drones or sensors, allowing users to expand the surveillance coverage based on their evolving needs.

Security:

The entire system, including data transmission, should implement robust security measures to protect user privacy and prevent unauthorized access.

Usability:

The user interface of the mobile app should be intuitive, requiring minimal training for users to operate and configure the surveillance drone system effectively.

Maintainability:

The system should be designed for ease of maintenance, with straightforward procedures for software updates, repairs, and component replacements.

Interoperability:

The surveillance system should seamlessly integrate with existing smart home ecosystems and devices, ensuring compatibility with a variety of technologies.

Adaptability:

The system should be adaptable to different indoor environments and user preferences, allowing for customization of surveillance settings.

Efficiency:

The drone's operational efficiency should be maximized, balancing performance with energy consumption to extend battery life and reduce the need for frequent recharging.

Accuracy of Object Recognition:

The system's object recognition capabilities should be highly accurate to minimize false alarms and provide reliable detection of security-related events.

Privacy:

The system should prioritize user privacy, offering features such as camera disablement in specific areas to address privacy concerns.

User Satisfaction:

The overall user experience should contribute to high user satisfaction, ensuring that the system meets or exceeds user expectations for home surveillance.

Resilience to Environmental Conditions:

The surveillance drone should be resilient to common household environmental conditions, such as changes in lighting, temperature variations, and indoor obstacles.

Legal and Ethical Compliance:

The system should comply with all relevant laws, regulations, and ethical standards governing the use of surveillance devices in residential settings.

Cost-Effectiveness:

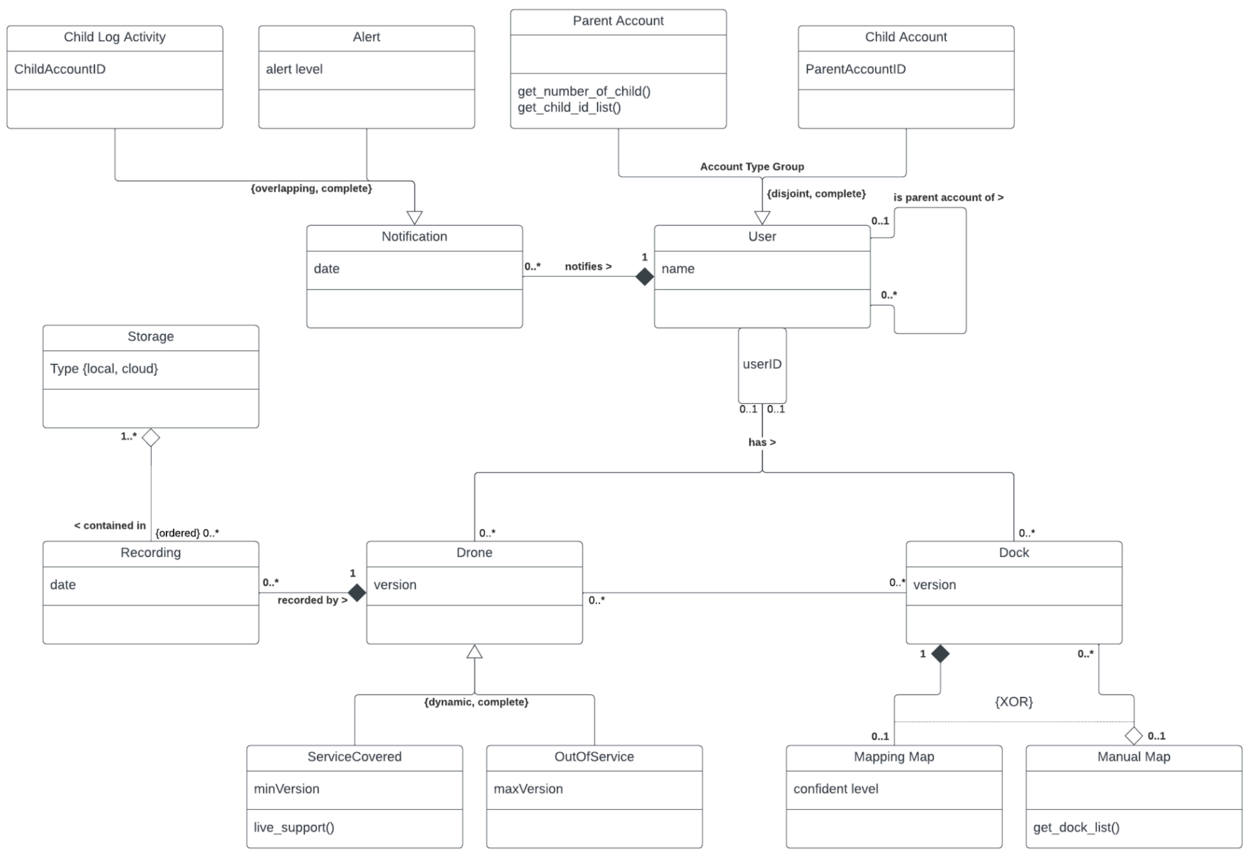
The system should provide value for money, considering the cost of the hardware, software development, and ongoing maintenance in relation to the benefits it delivers to users.

Adherence to Standards:

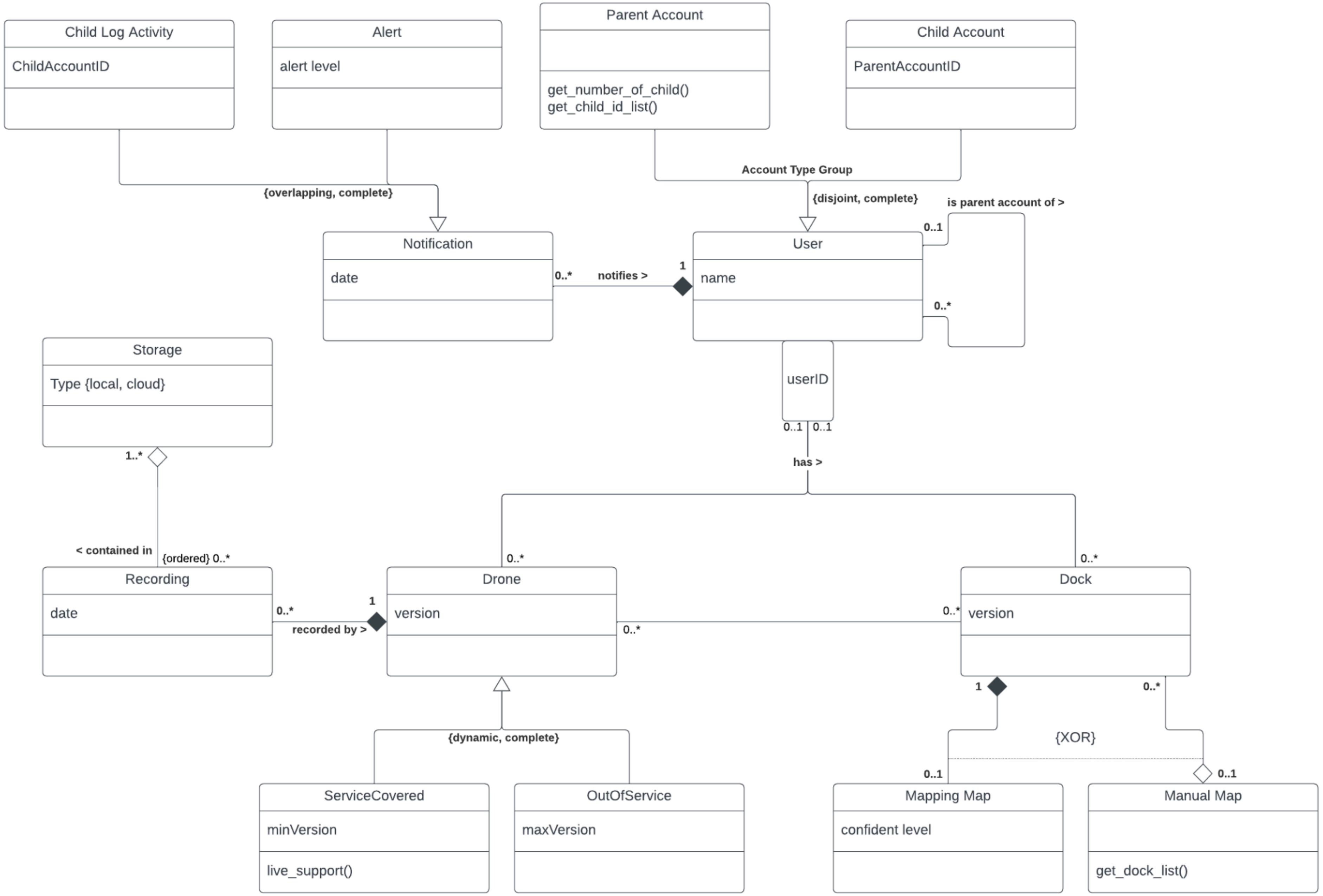
The system should adhere to industry standards for drone design, communication protocols, and data security to ensure compatibility and compliance.

Analytical Class Diagram

An Analytical Class Diagram serves as a structured visual representation of the relationships, attributes, and behaviours of classes within a system. Unlike a typical Class Diagram that outlines the static structure of a system, an Analytical Class Diagram delves deeper into the analysis of classes, emphasizing their interactions and dependencies during specific scenarios. This type of diagram is a valuable tool for system analysts, designers, and developers, as it provides insights into the dynamic aspects of class interactions, helping to refine and optimize the system's design. By illustrating how classes collaborate and exchange information in different situations, an Analytical Class Diagram aids in making informed decisions about system architecture, functionality, and data flow.



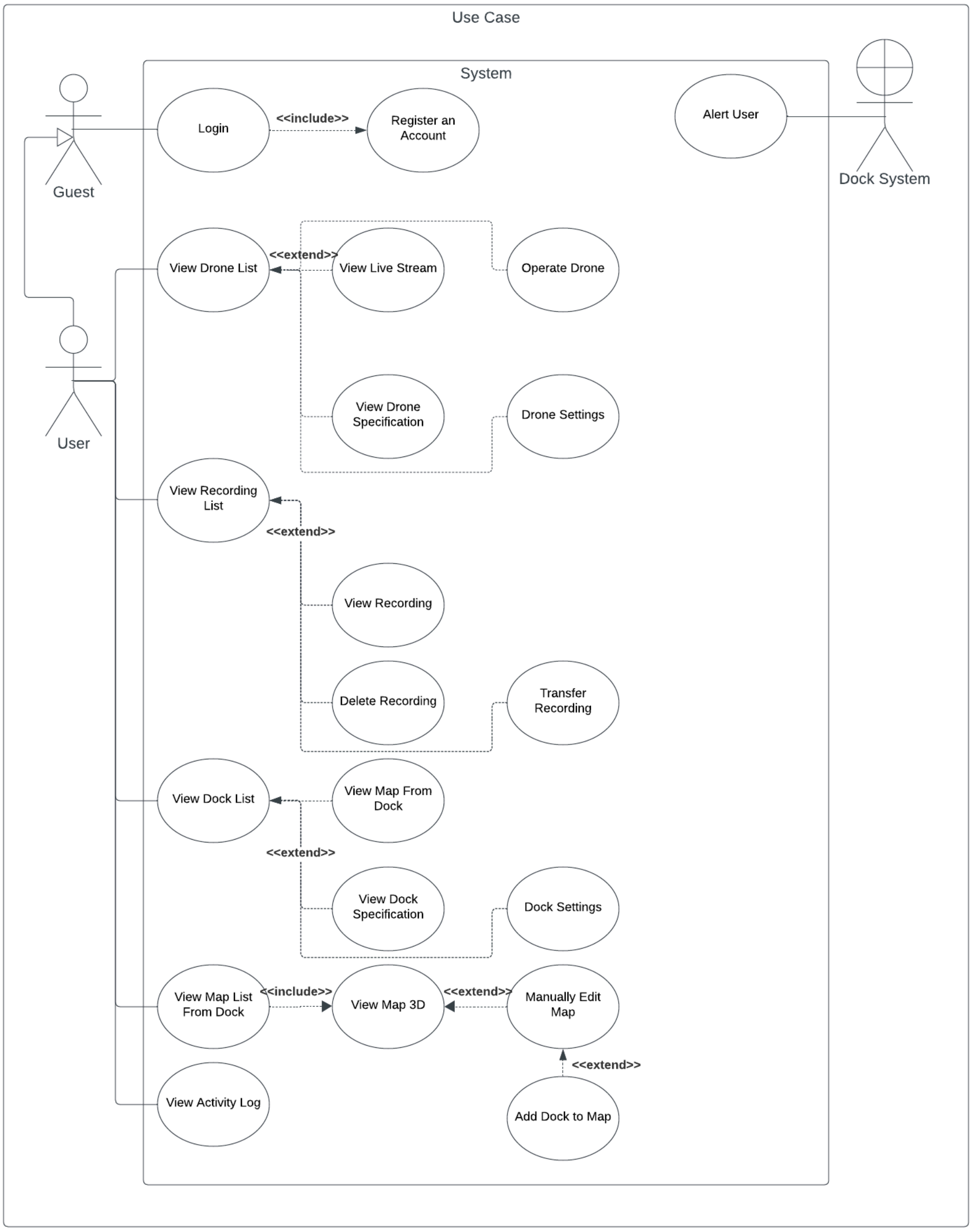
(Landscape View Below)



Child Log Activity
ChildAccountID
Alert
alert level
Parent Account
get_number_of_child()
get_child_id_list()
Child Account
ParentAccountID
{overlapping, complete}
Account Type Group
{disjoint, complete}
is parent account of >
0..1
0..*
Notification
date
0..*
notifies >
1
User
name
userID
0..1
0..1
has >
Storage
Type {local, cloud}
1..*
< contained in
{ordered} 0..*
Recording
date
0..*
recorded by >
1
Drone
version
0..*
0..*
Dock
version
0..*
0..*
1
0..*
{dynamic, complete}
{XOR}
0..1
0..1
ServiceCovered
minVersion
live_support()
OutOfService
maxVersion
Mapping Map
confident level
Manual Map
get_dock_list()

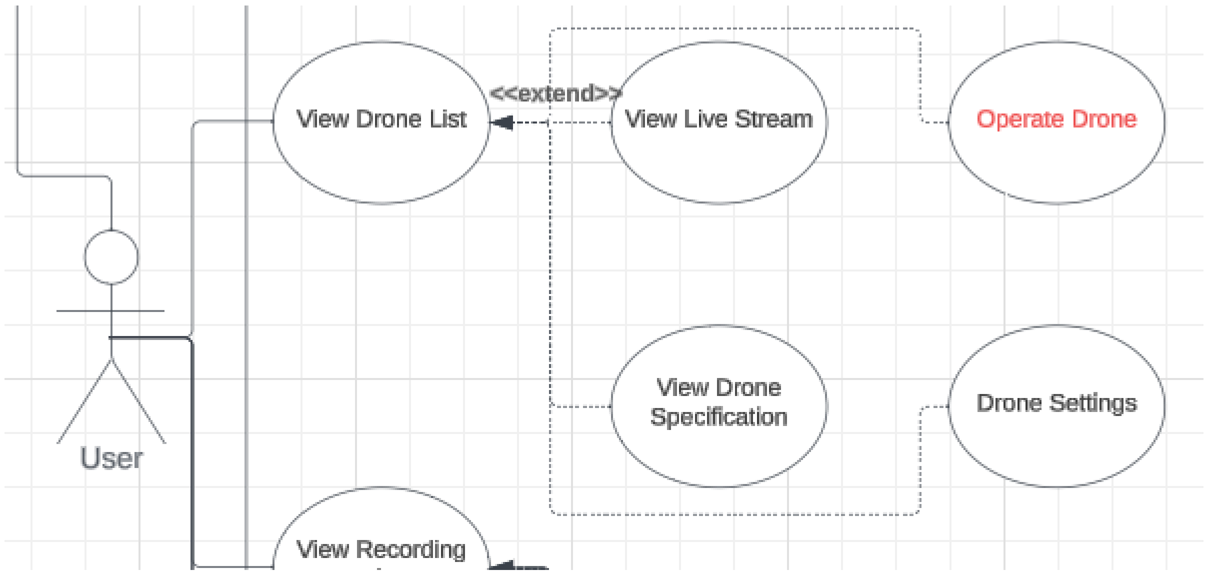
Use Case Diagram

A Use Case Diagram serves as a visual representation of the functional requirements and interactions within a system from the perspective of its users, known as actors. It provides a high-level view of how users and system components collaborate to achieve specific functionalities. The diagram illustrates various use cases, each representing a distinct interaction between actors and the system. By mapping out these use cases and their relationships, stakeholders gain a clear understanding of the system's key functionalities and how users interact with it. This graphical representation aids in communication, requirements analysis, and the visualization of system behaviour.



Use Case Scenario

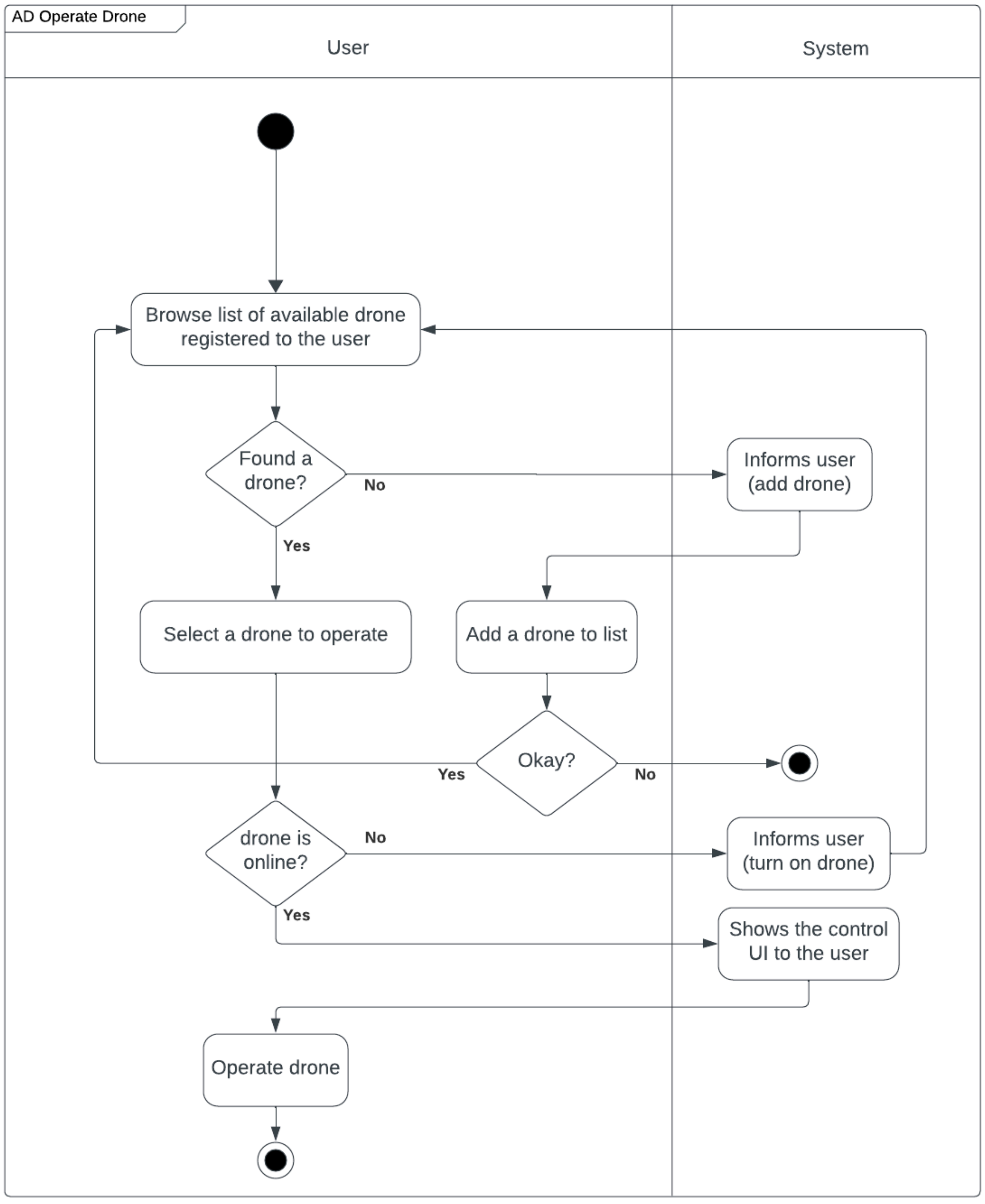
A Use Case Scenario introduces a specific situation or interaction between users and a system, highlighting the key steps and expected outcomes. It serves as a concise narrative outlining how users interact with the system to achieve a particular goal, providing a high-level view of system functionality and user engagement.



Use Case Name	Operate Drone
Priority	High
Type	User Goal
Actors	1. User
Description	Allows the user to control the drone manually
Preconditions	1. The user is logged into the System 2. The user has selected a drone from the list
Postconditions	The control UI interface is shown to the user with the drone's live camera
Included Use Cases	None
Extended Use Cases	None
Initiating the Scenario	The user chooses to operate a drone
Main Success Scenario	1. The user clicks "Operate Drone" 2. The system verifies that the drone is online
Alternative And Optional Scenario	Step 2a: The drone is offline 1. The system alerts the user that the drone is offline 2. The system returns the user to the drone list page

Activity Diagram

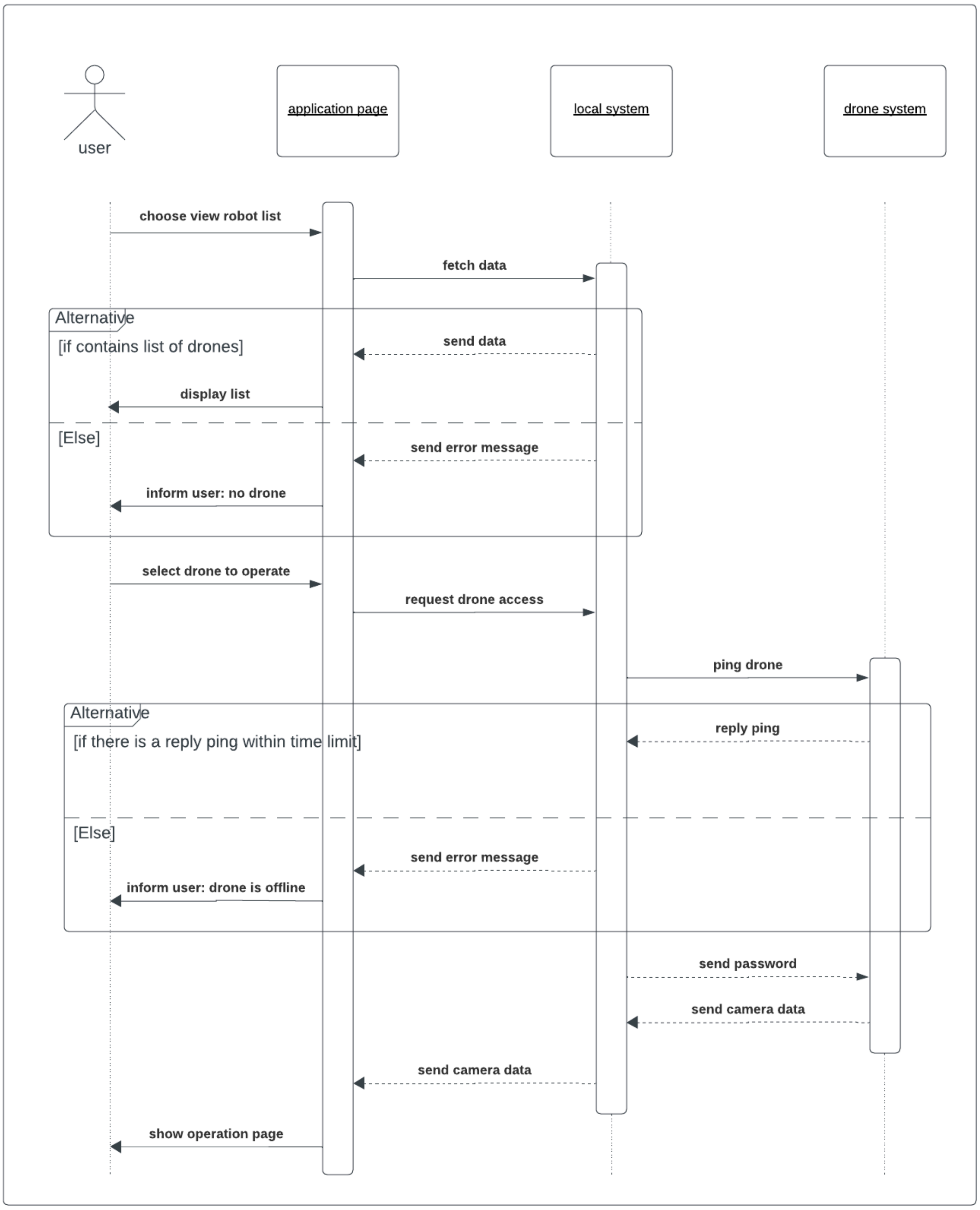
An Activity Diagram visually represents the flow of activities and actions within a system or process, emphasizing the sequence, decisions, and parallel activities. It provides a dynamic view, aiding in the understanding and modelling of business processes and software workflows.



AD Operate Drone
User
System
Browse list of available drone registered to the user
Found a drone?
No
Yes
Informs user (add drone)
Select a drone to operate
Add a drone to list
Okay?
Yes
No
drone is online?
No
Yes
Informs user (turn on drone)
Shows the control UI to the user
Operate drone

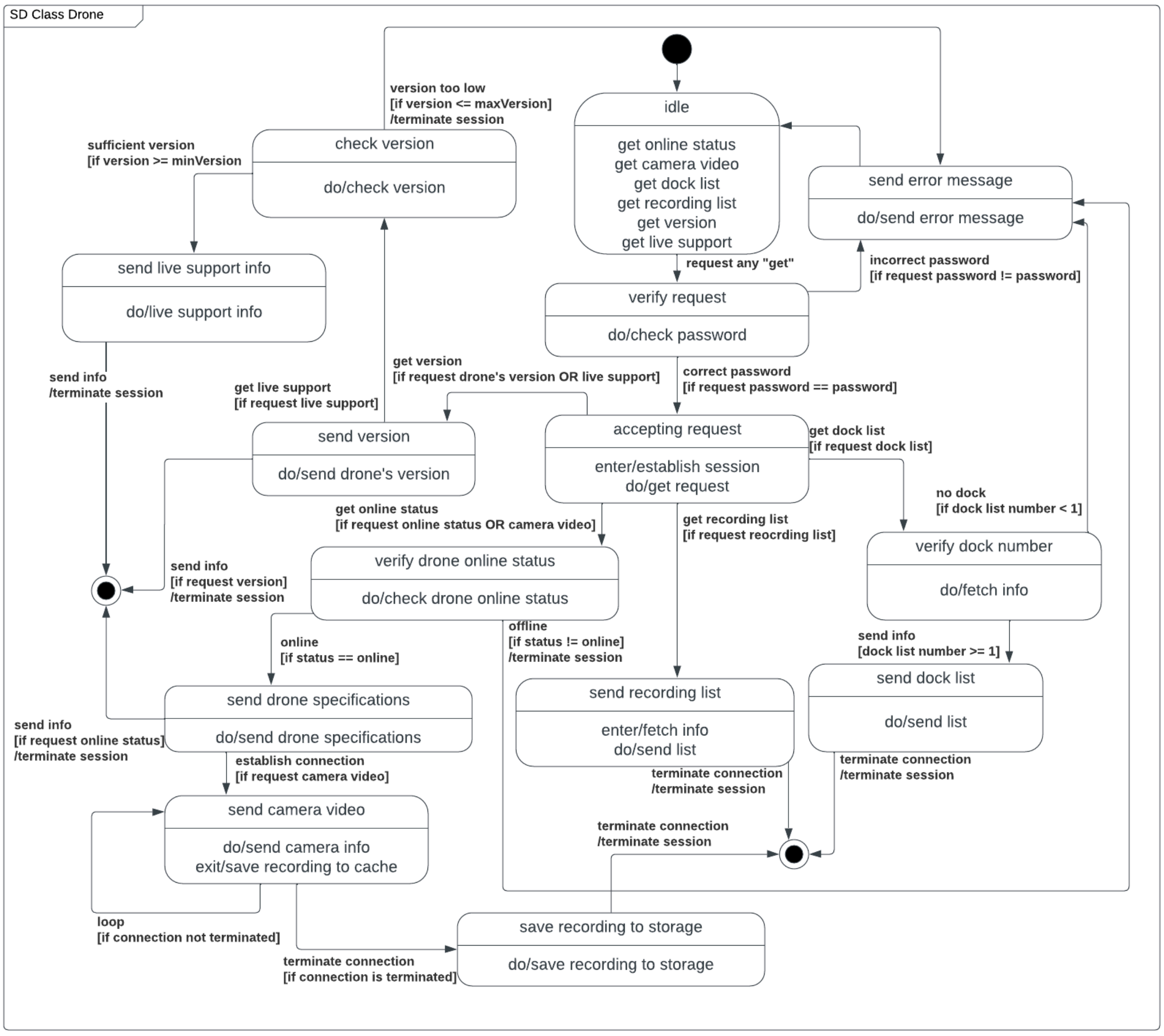
Interaction diagram

An Interaction Diagram visually depicts how objects interact and communicate in a system during a specific scenario. It includes Sequence Diagrams for chronological message exchange and Communication Diagrams for emphasizing object relationships. These diagrams enhance understanding, communication, and design in dynamic aspects of a system.



State Diagram

A State Diagram for the "Drone" class visually illustrates the various states and transitions the drone undergoes during its lifecycle. It captures the dynamic behaviour, showcasing how the drone responds to external events. This diagram is crucial for system analysis, design, and understanding the operational logic of the drone.



Dynamic Analysis

Dynamic analysis involves studying a system's behaviour and performance during runtime. It assesses interactions, identifies issues, and optimizes software for real-world efficiency and robustness.

Analysis From Use Case Diagram:

1. The diagram shows the usage of the system for real-world cases, especially from these use cases
    a. Manually Edit Map
    b. Add dock to Map
    c. View Map 3D
    d. Add Dock
2. The diagram also shows the connection between Extended usages. However, this is mostly covered in the previous Class Diagram.

Analysis From Activity Diagram:

1. The diagram shows the internal states between use cases, this gives some insights into which class is missing some components or functions, for example,
    a. The drone should be checked if it is online before the user can operate it, therefore, a function that checks the status of a drone should be included

Analysis From Interaction Diagram:

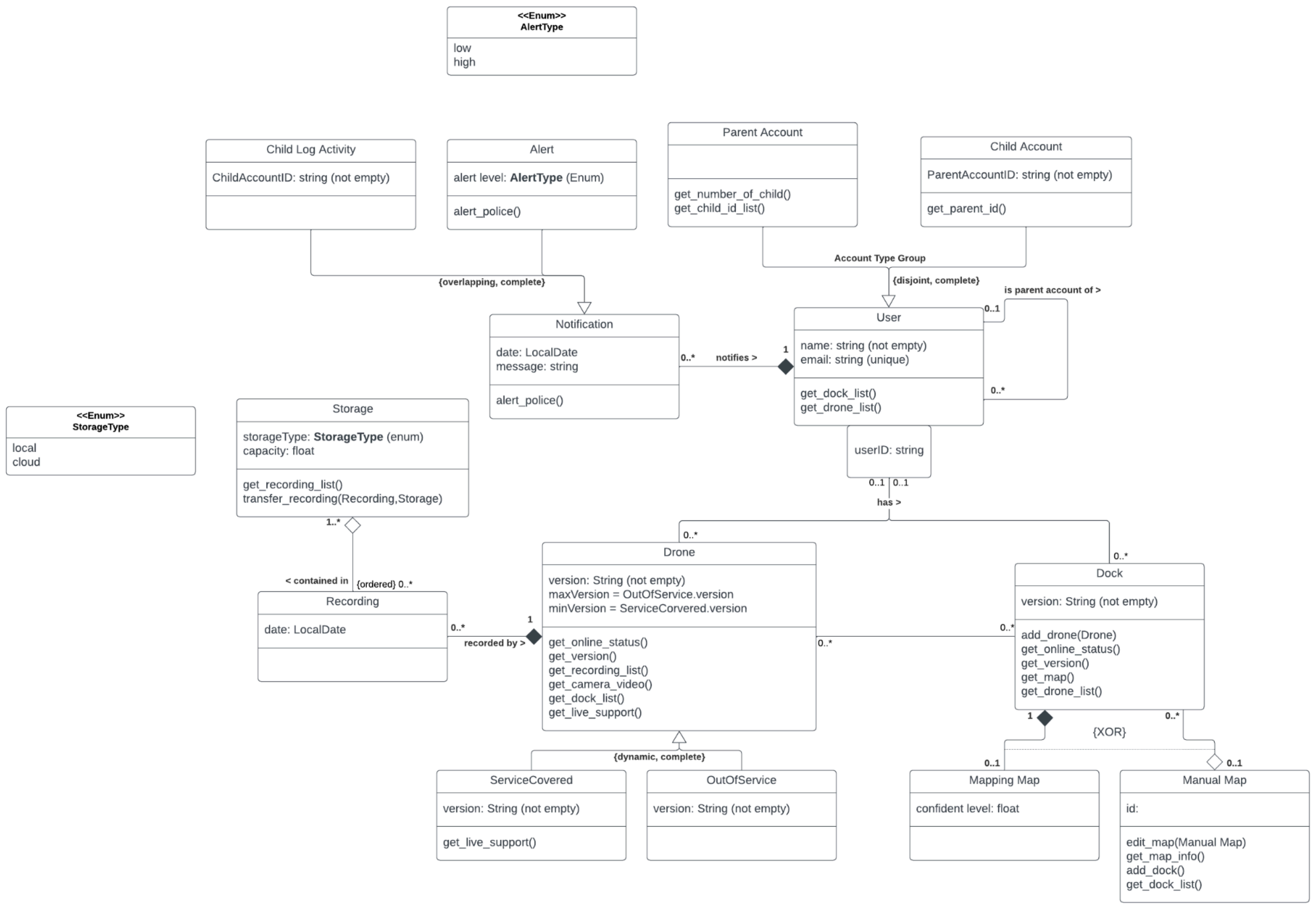
1. The diagram shows the interaction about the use case, specifically between the local system and the drone system. It shows that the drone system should not handle too many interactions and therefore can be kept lightweight with most computing done on the local system. Although this does not change anything on the class diagram, it is a valuable insight.

Analysis From State Diagram:

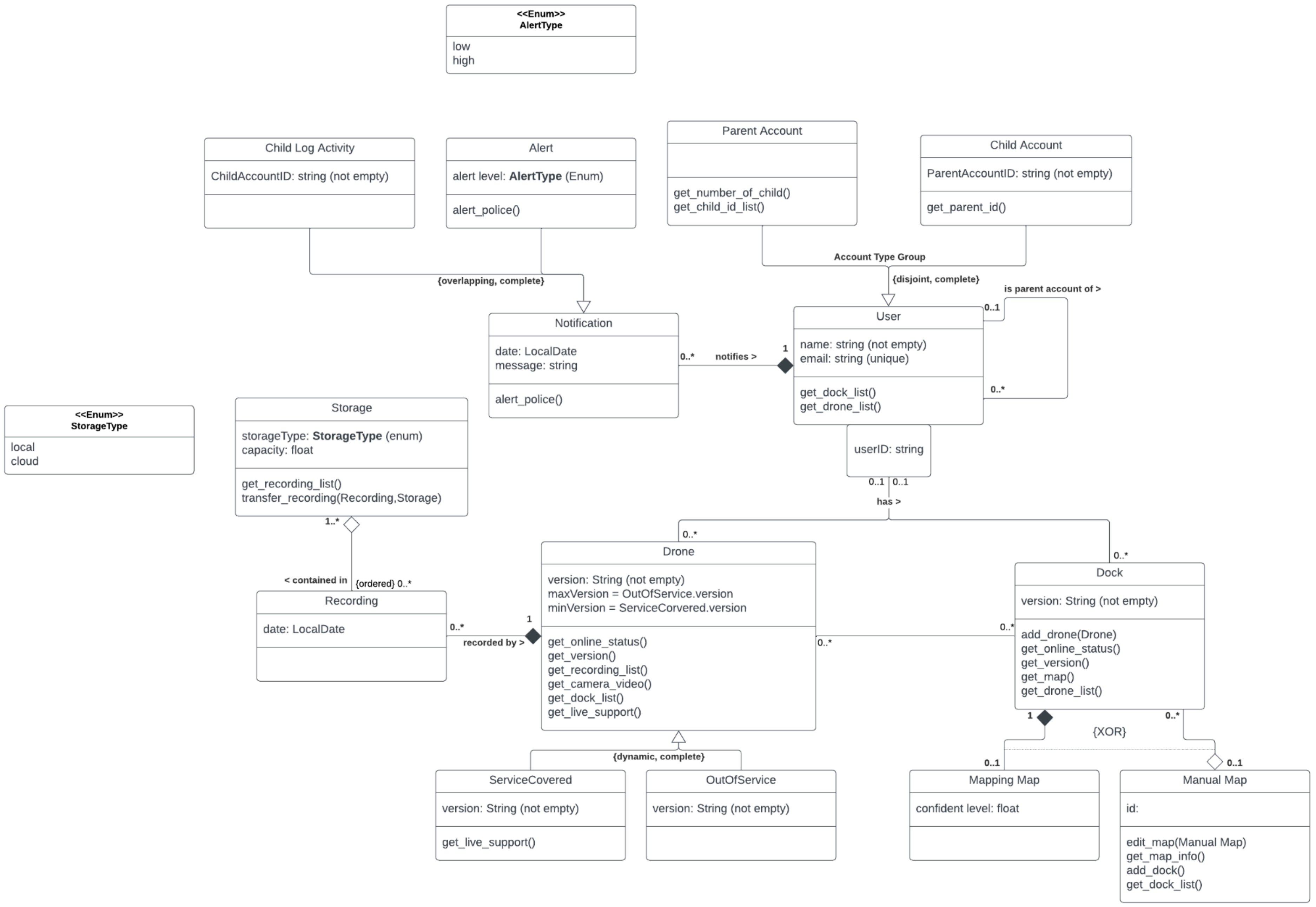
1. The diagram shows the state of the Drone class, which should have multiple missing functions from the first Class Diagram, specifically,
    a. Get online status
    b. Get drone version
    c. Get dock list
    d. Get recording list
    e. And Get live support
2. Moreover, the diagram also reminds us that another similar class, the Dock class, is also missing similar functionalities.

Implementation Class Diagram

An Implementation Class Diagram details class attributes, methods, and relationships in a system, guiding developers in translating design into executable code during software implementation. It serves as a coding blueprint in the development phase.



(Landscape View Below)



<<Enum>>
AlertType
low
high
Child Log Activity
ChildAccountID: string (not empty)
Alert
alert level: AlertType (Enum)
alert_police()
Parent Account
get_number_of_child()
get_child_id_list()
Child Account
ParentAccountID: string (not empty)
get_parent_id()
Account Type Group
{disjoint, complete}
is parent account of >
0..1
0..*
{overlapping, complete}
Notification
date: LocalDate
message: string
alert_police()
0..*
notifies >
1
User
name: string (not empty)
email: string (unique)
get_dock_list()
get_drone_list()
userID: string
0..1
0..1
has >
<<Enum>>
StorageType
local
cloud
Storage
storageType: StorageType (enum)
capacity: float
get_recording_list()
transfer_recording(Recording,Storage)
1..*
< contained in
{ordered} 0..*
Recording
date: LocalDate
0..*
recorded by >
1
0..*
Drone
version: String (not empty)
maxVersion = OutOfService.version
minVersion = ServiceCorvered.version
get_online_status()
get_version()
get_recording_list()
get_camera_video()
get_dock_list()
get_live_support()
0..*
0..*
0..*
Dock
version: String (not empty)
add_drone(Drone)
get_online_status()
get_version()
get_map()
get_drone_list()
1
0..*
{XOR}
{dynamic, complete}
ServiceCovered
version: String (not empty)
get_live_support()
OutOfService
version: String (not empty)
0..1
Mapping Map
confident level: float
0..1
Manual Map
id:
edit_map(Manual Map)
get_map_info()
add_dock()
get_dock_list()

UI Design

Introduction to Page Flow: Home-to-Drone-to-Operate Page

As users navigate through our system, we have designed a seamless and intuitive page flow to enhance their experience from the "Home Page" to the "Drone Page" and finally to the "Operate Page." Each step in this journey is purposefully crafted to provide users with easy access to information, control, and interaction with their 3D Household Surveillance Drone System.

Button:

Buttons are highlighted in colour distinct for the user, ensuring that every user can see the button easily, in this case, it is highlighted in **blue** (specific of the colour may change when implementing).

Selected Button:

Buttons are highlighted in colour distinct for the user, ensuring that every user can see that button easily when it is selected, in this case, it is highlighted in **green** (specific of the colour may change when implementing).

Specific Button:

Exit, Close, and Cancel buttons:

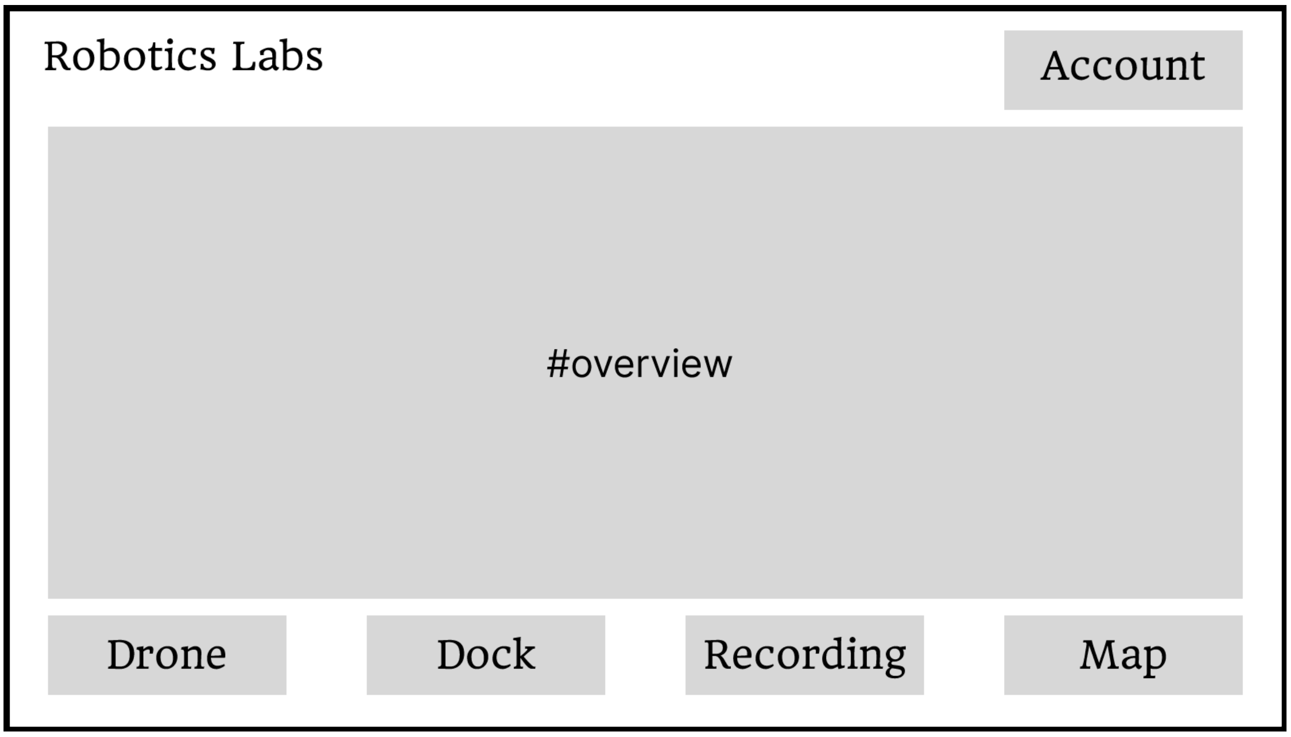
These buttons are highlighted in a colour familiar to the user that signifies exit, in this case, it is **red.**

Scroll Bar:

Scroll bars are designed in a way most users should be able to recognise, it stems from the design on most PC operating systems, Windows, macOS, and Linux.

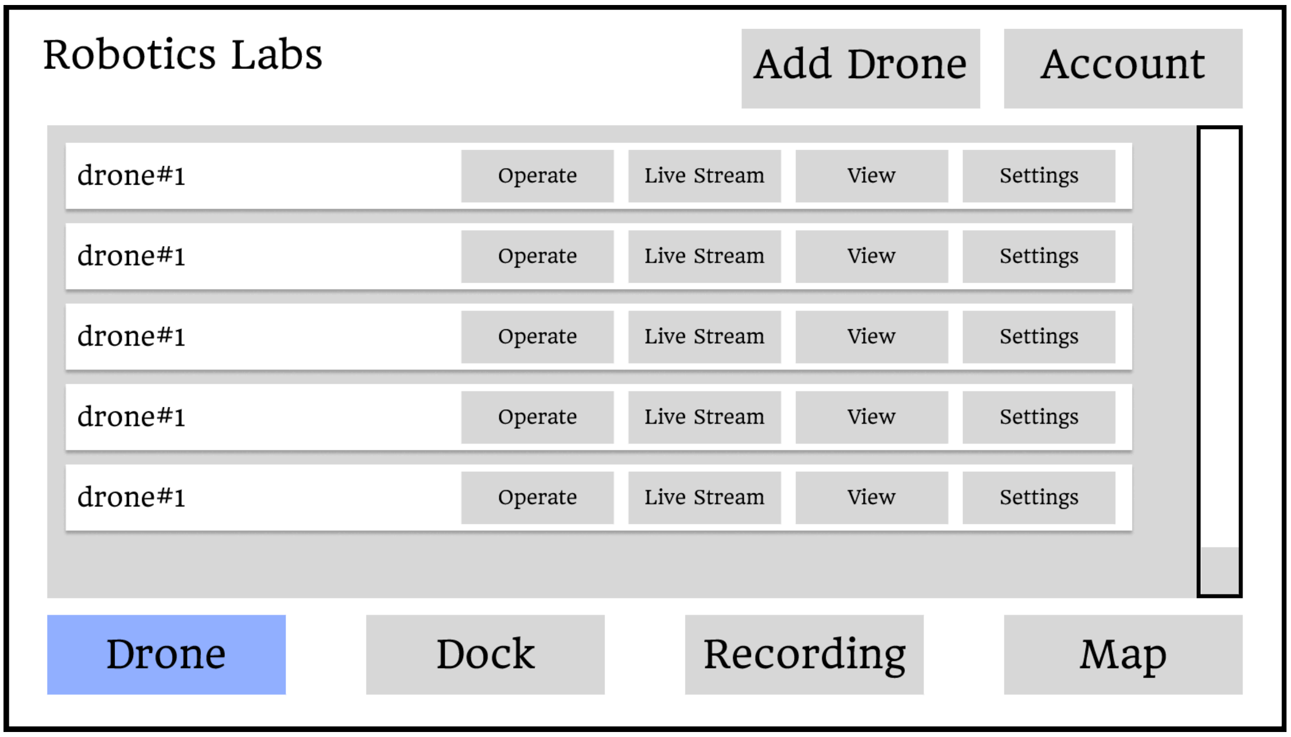
1. Home Page:

The journey begins on the "Home Page," where users are greeted with an overview of the system's status and essential information. Here, users can quickly assess the overall security of their household and access key features.



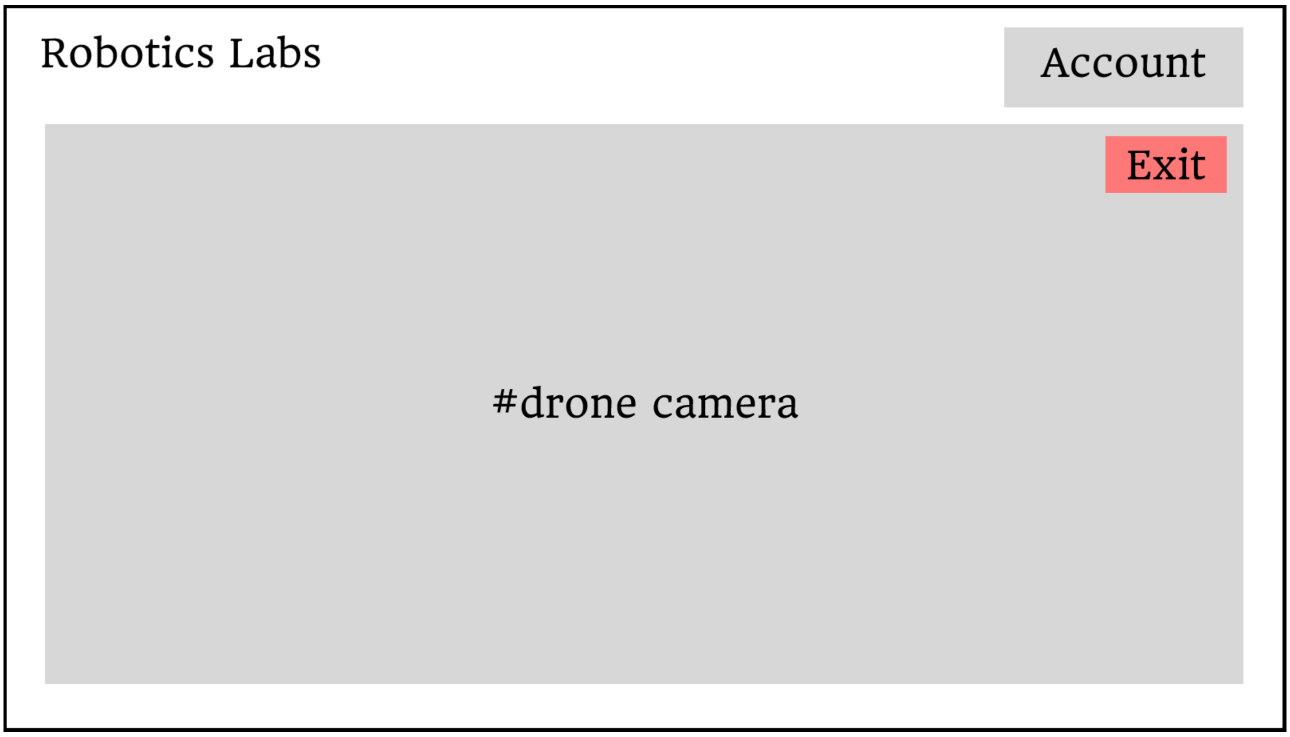
2. Drone Page:

From the "Home Page," users can seamlessly transition to the "Drone Page" to delve deeper into the specifics of their surveillance drone. This page offers more into the drone's options.



3. Operate Page:

The final step in this journey leads users to the "Operate Page," where they have direct control over the surveillance drone, empowering users with real-time control and monitoring capabilities. The user can change the control settings in the settings section from the previous page.



Design Decisions

Persistence:

Every class must be persistent, therefore, we save their information in storage. In our case, we have 2 types of storage

1. Robotics Labs system, which is a storage that stores class objects in the cloud or in the Robotics Labs side, to ensure their privacy

Class Extent:

Every class extent should be a separate class.

1. Classes that have extent stored in the Robotics Labs system, these classes' information should be private to the users
    a. User class
    b. Recording class: cloud storage type
    c. Notification class: Alert Inheritance
2. Classes that have extent stored in the local system of the user, these classes should be accessible offline and is not necessary for the Robotics Labs to have information about them
    a. Drone class
    b. Dock class
    c. Map class
    d. Recording class: local storage type
    e. Notification class

Association:

Some classes have information about another class by their association, however, a link between their objects depends on the classes since not every class should have the same type of link

1. Associations that use the object ID to store information about their links, since most of these associations are stored in the Robotics Labs system.
    a. User - Drone
    b. User - Dock
    c. User - Notification
    d. User - Recording
2. Associations that use the object to store information about their links. Since most of these associations are stored in the local system, using the object ID may reduce their performance.
    a. Dock - Drone
    b. Drone - Dock
    c. Map - Dock