SOFTWARE REQUIREMENTS SPECIFICATION for FOLLOW ME DRONES

COS301, University of Pretoria, South Africa

Five Guys, One Branch

Francois Venter (u14177553)
Brendon van Biljoen (u17011842)
Devon Petrie (u17019266)
Len Bekker (u11026953)
Gilad Tabul (u16154658)

July 19, 2019

Preface

Lorem ipsum dolor sit amet.

Structure of Specification

Each unit will focus on <SOMETHING>.

Introduction

1.1 Purpose

The purpose of this document serves to provide an in-depth analysis into the user characteristics, requirements and use cases for an autonomous drone. It will utilize the latest technologies in the industry to detect objects and navigate itself based on the detections. The document includes detailed information regarding the requirements of the project. Furthermore it explains how requirements are to be met.

The primary aim of the Follow Me Drones System (FMDS) is to provide support to a ranger in the field. The system will identify any dangers in the vicinity of the ranger by identifying that which is possibly undetectable to a human walking in the bush. It will identify the objects threat level to the ranger and notify them of any detections.

1.2 Definitions, Acronyms and Abbreviations

- FMDS - Follow Me Drones System
- OS - Operating system
- UI - User interface
- FR - Functional requirement

1.3 Project Scope

In order to avoid notifying the ranger of themselves as a detection, the software will need to be initially trained to know what the ranger looks like. This will be done before take-off. The ranger will place the drone with the camera facing them and start the drone, the drone will identify the ranger in front of it and set the ranger as its primary tracking target.

The movement of the drone will be based on the rangers current location and follow the ranger based on their movements. The ranger will make use of a wearable beacon which will communicate with the drone. This will allow the drone to continuously know where the ranger is without needing to maintain a visual line of sight. With this method, the drone is able to freely move around and patrol the area around the ranger instead of only being able to look in the ranger's direct vicinity.

The drone will fly in a pre-defined pattern around the ranger, identifying objects as it goes and performing the notifications. At points during the flight pattern, the drone will take a snapshot below it for later use in creating a 2D map of the reserve. A video of the drones surveillance will be captured during flight and saved onto the drones storage, usage of this will vary based on the users needs. Any objects (be it animals, humans or human carriable/wearable items) that are detected by the drone's object detection software will have their location recorded and a pin will be dropped on the 2D map. This can be used to detect hotspots for animals or even poachers.

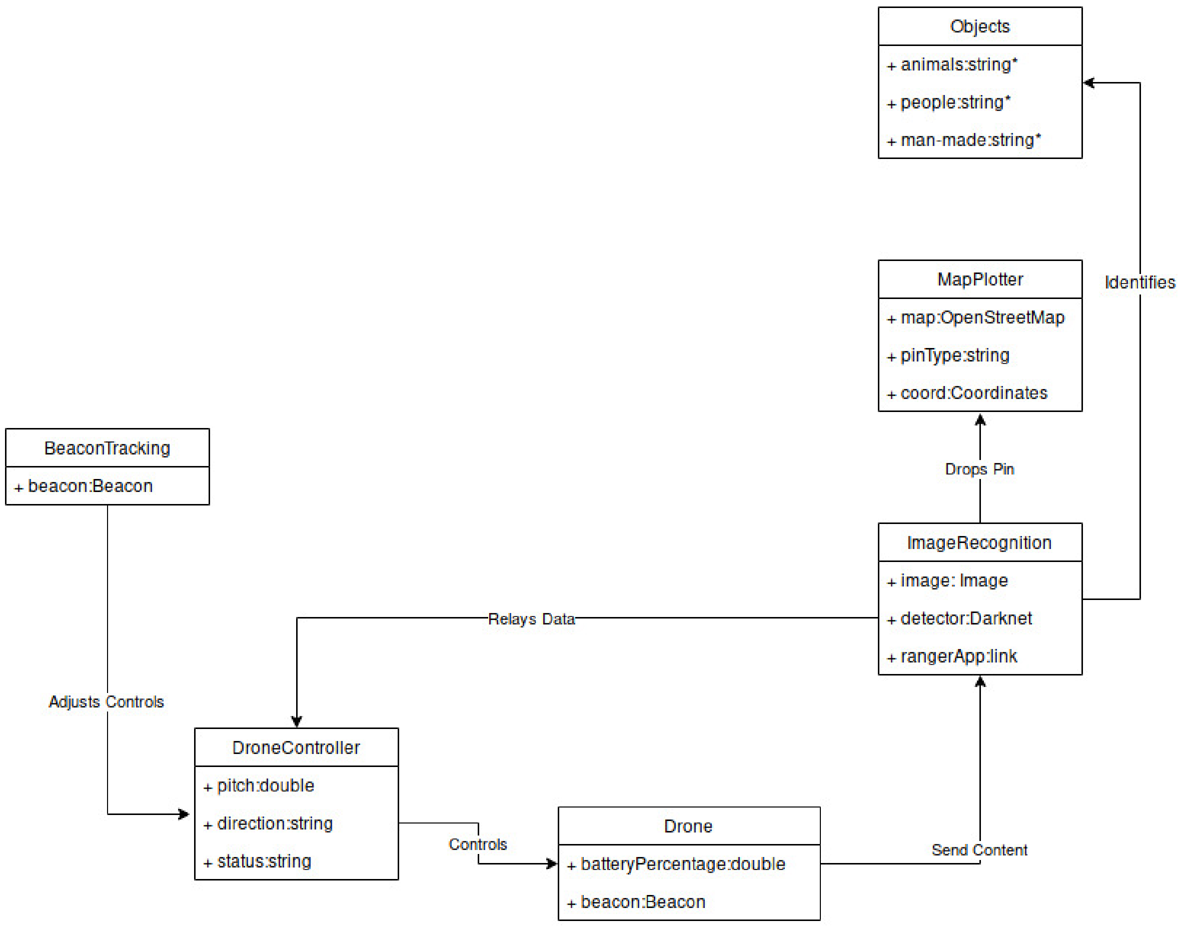
1.4 Intended Audience and Reading Suggestions

This Software Requirements document is intended for:

- The end users of this project are namely GroupElephant, EPI-USE and the COS301 lecturers.

Overall Description

2.1 Domain Model



Domain Model

2.2 User Classes and Characteristics

A typical user of the system is characterised as an employee (volunteer) of ERP. These employees can be divided into two main classes:

Ranger:
The ranger is a person entrusted with protecting and preserving the wildlife park. The FMDS will be used as an augmentation to the current ranging procedures. The ranger needs to be technology savvy to use the system. It is necessary that the ranger knows how to operate a mobile device ie. phone/tablet. This is an essential part of controlling the drone with regard to take off, initiating flight patterns and landing procedures.

Command Centre Personnel:
Command center personnel are employees with specialized skills to analyze data collected

during the time the drone is airborne. Data to be analyzed includes mapping data and new footage for further training of the object recognition.

2.3 Constraints

2.3.1 Business Constraints

2.3.2 Technical Constraints

2.4 Operating Environment

The operating environment for the FMDS is listed below:

- Device - Raspberry Model 3B+
- OS - Rasbian 9 (Stretch)
  - Linux Kernel v14.4
  - gcc v6.3
  - apt v1.4.9
- Camera - Raspberry Pi Camera Module V2.8 1080p

Architectural Design

3.1 Architectural Overview

3.1.1 Interactive subsystem (Type)

The mobile app allows the ranger to perform specific actions, such as:

- Connecting to the drone
- Giving basic commands to the drone
- Viewing the detections

This is done because the ranger may be required to connect to multiple drones and have the ability to communicate with each indepenently.

3.1.2 Event driven system (Type & Style)

- Object recognition notifies the server based on object detections made in real time. The server will process this information and decide whether or not an event has been triggered. If it has, then it will emit a signal to the mobile app notifying it of a detection.
- The server will be notified by the app of any ranger movements. The server will then issue commands to the drone which will cause the drone to follow the ranger.

This was done because the drone needs to change its actions based on actions. Our systems need to change their behaviour based on specific inputs and adjust accordingly.

3.1.3 Database system (Type) & persistence (Style)

- Dropped pins generated from server snapshots will contain animal snapshots, location data and drone information. These pins will be stored in a database for future use.

This is done in order to record data for any future use that may be required.

3.1.4 Client-server

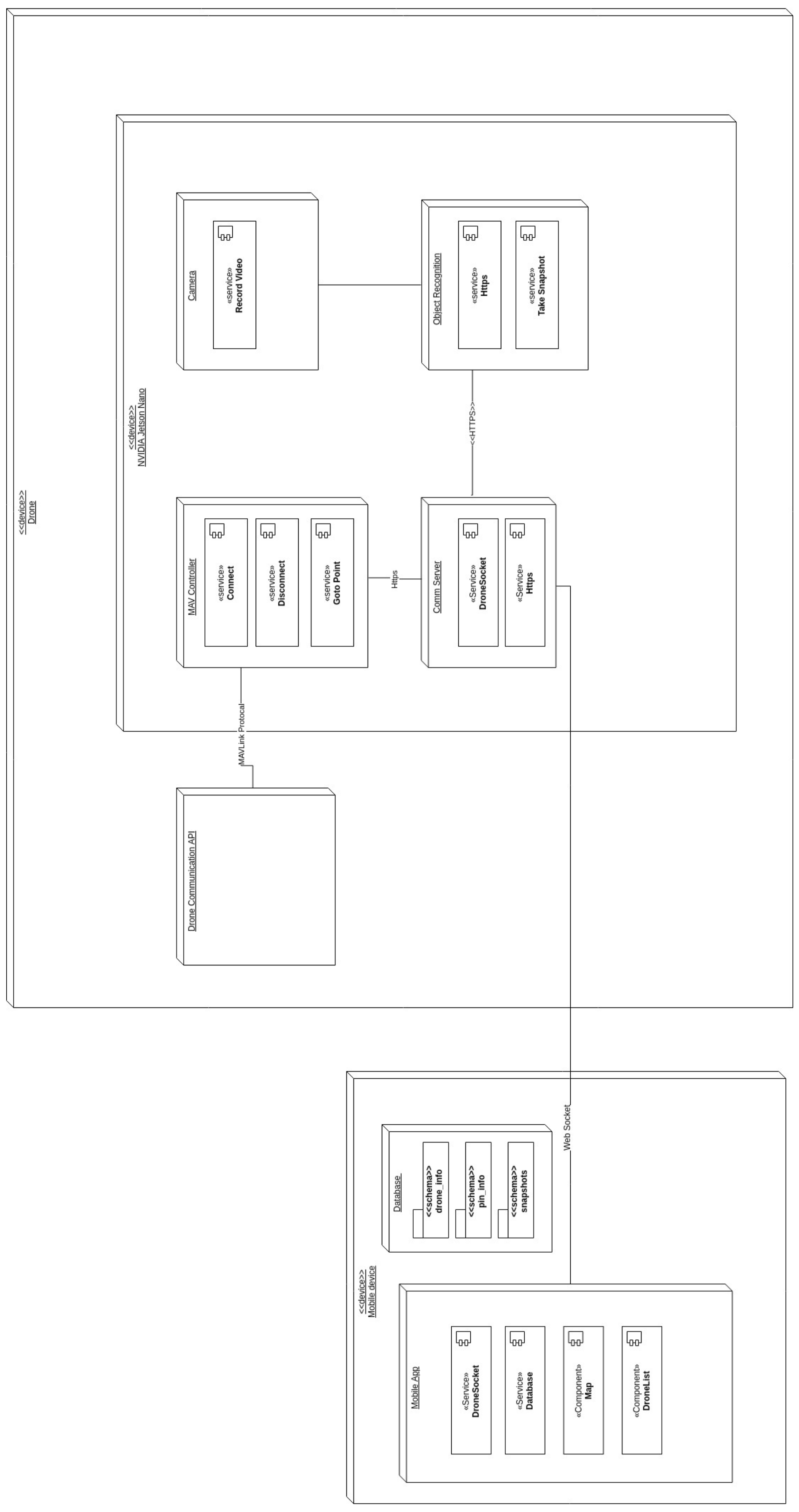
- At the centre of our system we have a server that acts as a mediator. The clients, object recognition and the mobile app, communicate back-and-forth with the server, relaying information as events are triggered.

We have multiple subsystems that need to communicate with each other, thus a central server with multiple clients was the ideal solution. The server also allows us to open a socket between it and the app, allowing data to be pushed instead of needing to poll.

3.2 Design Principles

- Due to separation of concerns, high cohesion and modular design, our system is implemented in such a way as to be easily changed in the future. Thus implying a design for changeability, maintainability and upgradeability.
- The subsystems are designed to be general purpose, to allow for repurposing if need be.

3.3 Deployment Diagram


<<device>>
Drone
<<device>>
NVIDIA Jetson Nano
Camera
«service»
Record Video
Object Recognition
«service»
Https
«service»
Take Snapshot
<<HTTPS>>
MAV Controller
«service»
Connect
«service»
Disconnect
«service»
Goto Point
Https
Comm Server
«Service»
DroneSocket
«Service»
Https
MAVLink Protocal
Drone Communication API
Web Socket
<<device>>
Mobile device
Database
<<schema>>
drone_info
<<schema>>
pin_info
<<schema>>
snapshots
Mobile App
«Service»
DroneSocket
«Service»
Database
«Component»
Map
«Component»
DroneList

External Interface Requirements

4.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

4.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

4.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

4.4 Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

Subsystems

5.1 Drone Navigation

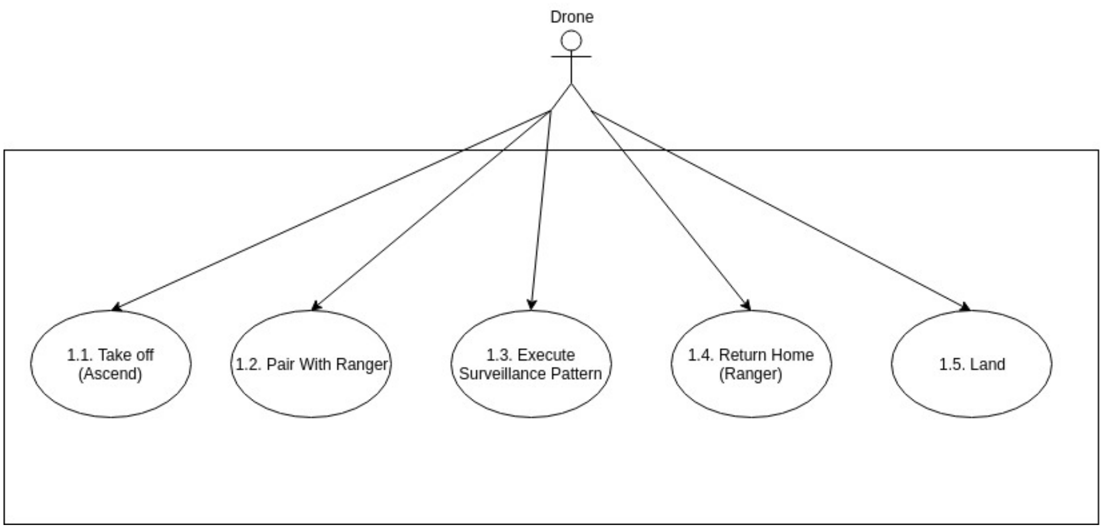
5.1.1 Functional Requirements

- **R1:** FMDS will **follow** the ranger
  - **R1.1** FMDS will allow the ranger to perform **initialisation** for the drone.
    - **R1.1.1** FMDS will allow the ranger to set himself as the validated object.
      - The validated object will not be treated as a normal object when searching the field of view of the drone. Notifications are sent to the ranger when objects are detected, however notifications will not be sent every time the ranger is identified.
    - **R1.1.2** FMDS will allow the ranger to initialise his beacon.
      - The beacon will allow the drone to know where the ranger is at any moment in time, which will assist in expanding the drone's field of view since the drone will not need to be facing the ranger.
  - **R1.2** FMDS will allow the ranger to specify a **flight pattern** for the drone
    - **R1.2.1** FMDS will detect objects while on its flight path.
      - Object recognition (as a service) will be running from the moment the ranger initialises the drone until the moment it lands. Rectangles will be drawn around identified objects along with the label for each identified object.
    - **R1.2.2** FMDS will record live video while on its flight path.
      - Stored video data will be used as further training data for the object recognition model.

- **R2:** FMDS will **patrol** the area around the ranger
  - **R2.1** FMDS will patrol by flying in the designated **flight pattern** around the ranger.
    - The desired flight pattern will be chosen upon initialisation and will remain as the default flight pattern unless updated during the drone's flight.
  - **R2.2** FMDS will perform **object detection**.
    - **R2.2.1** FMDS will drop a pin of the object detected on the 2D map.
      - The coordinates of the drone's current location will be sent to the server with a command to drop a pin at that location. In addition to the coordinates, the object's classification (elephant, rhino, person, etc) will will be sent to the server. the server will then attempt to assess the identified object's threat level. The server will then send an update to the ranger's app on their phone which will generate a pin. The pin will be coloured based on the perceived threat level.
  - **R2.3** FMDS will **record** video footage during its surveillance.
    - Stored video data will be used as further training data for the object recognition model.
  - **R2.4** FMDS will take **snapshots** of the reserve during its surveillance.

    * **R2.4.1** FMDS will build a 2D map of the reserve.
        · Snapshots will be periodically taken and stitched together at a later stage to create a virtual 2D map which is regularly updated. The updated 2D map will be useful to the rangers when wildlife, plantlife and the general health of the reserve is monitored.

5.1.2 Use Case Diagram



Drone Navigation Use Case Diagram

5.1.3 Technical Requirements

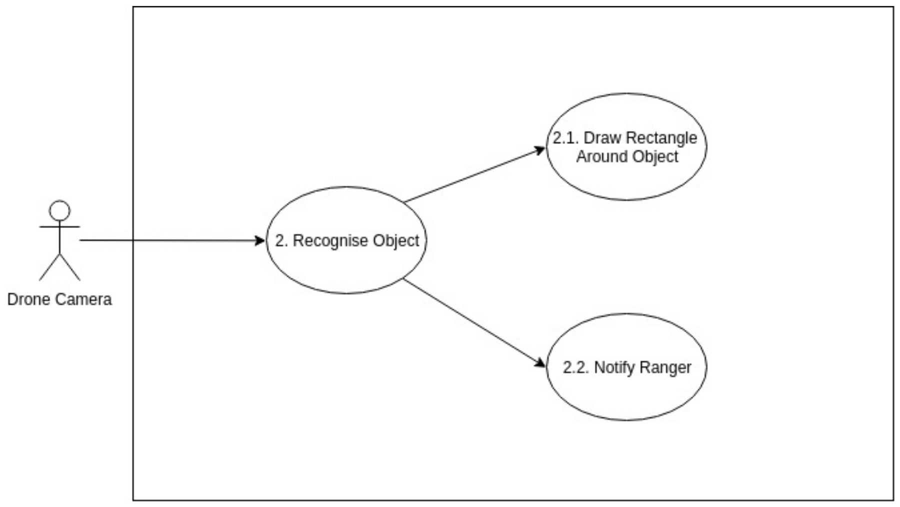
- Python v2.7

5.2 Object Recognition

5.2.1 Functional Requirements

- **R3:** FMDS will **detect objects** in the reserve
    - **R3.1** FMDS will allow the ranger to see a list of **objects** and their **locations**.
        * If an object is selected, the ranger will be pointed towards the detected object. Individual estimated threat levels will be displayed per object along with the object's classification.
    - **R3.2** FMDS will draw a rectangle around the objects that it detects.
        * Rectangles will be coloured and a label will be placed above the rectangle with the object's classification.
    - **R3.3** FMDS will notify the ranger of the objects that it detects.
        * The ranger will be notified of newly detected objects, their estimated threat level, classification and relative direction from the ranger.

5.2.2 Use Case Diagram



Object Recognition Use Case Diagram

5.2.3 Technical Requirements

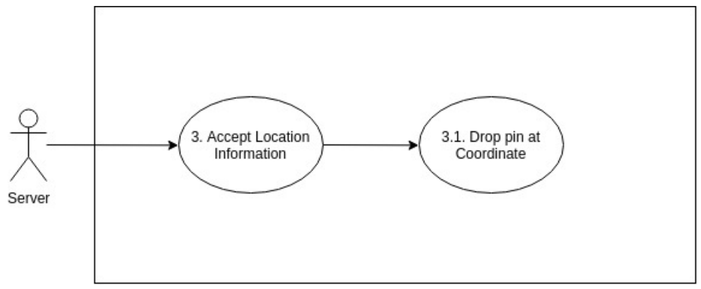
- gcc v8.0+
- cuda v10.0+
- opencv v3.0+
- darknet Yolo v3 Framework

5.3 Mapping

5.3.1 Functional Requirements

- **R4:** FMDS will build a **2D map** of the reserve
  - **R4.1** FMDS will use the snapshots taken to build a 2D map of where it has flown.
    - Snapshots will be periodically taken and stitched together at a later stage to create a virtual 2D map which is regularly updated. The updated 2D map will be useful to the rangers when wildlife, plantlife and the general health of the reserve is monitored.

5.3.2 Use Case Diagram



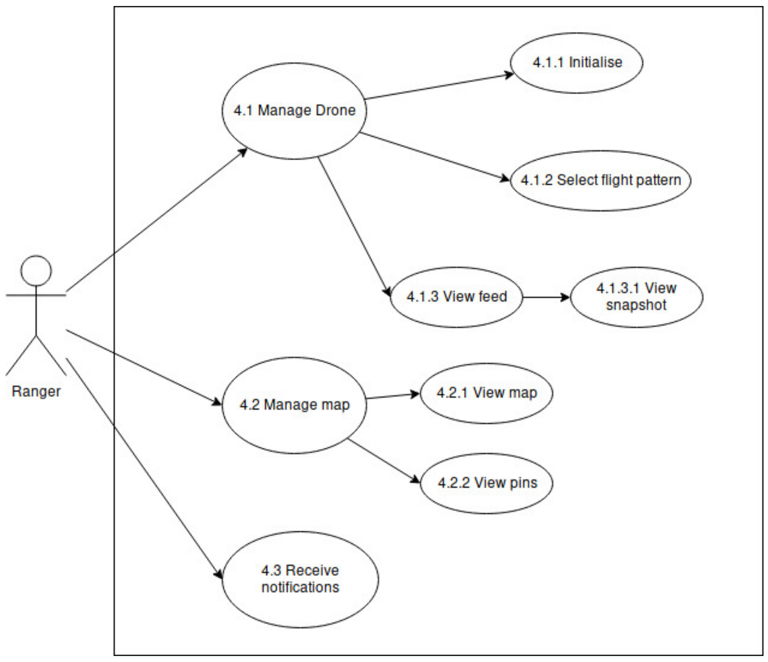
Mapping Use Case Diagram

5.4 User Application

5.4.1 Functional Requirements

- **R5:** FMDS will provide an interface for the ranger to initialise the drone.
  - Discussed in previous functional requirement (R1.1.2).
- **R6:** FMDS will provide an interface to display snapshots of the video feed.
  - The periodic snapshots (which will be used to generate the virtual 2D map) will be accessible via the application.
- **R7:** FMDS will provide an interface to display the live video feed.
  - The live video feed coming from the drone will be accessible via the application.
- **R8:** FMDS will retrieve map and pin data from the server.
  - Discussed in previous functional requirement (R2.2.1).
- **R9:** FMDS will provide an interface for the ranger to select a predefined flight pattern.
  - Discussed in previous functional requirement (R2.1).
- **R10:** FMDS will provide a user interface for the notification received from the other subsystems.
  - The notification received will include the relative position to the detected object from the drone's location.

5.4.2 Use Case Diagram



User Application Use Case Diagram

5.4.3 Technical Requirements

- Ionic cross platform framework.
- Cordova compilation suite.
- Angular Framework.

Nonfunctional Requirements

6.1 Performance Requirements

- The drones will be able to communicate with the ranger in less than 3 seconds with notifications or status updates.
- The system will be able to identify objects in less than 2 seconds and notify the ranger if needed.
- The drones will be able to change flight paths before within 2 seconds of spotting an object that it will collide with.
- The drone will be able to process the information and pin it on the map within a 3 second standstill period.

6.2 Safety Requirements

- The drone will take off and land by automated mechanisms to ensure ranger safety.
- The drone will keep a safe distance from the ranger at all times while in the air.
- The drone will ignore further commands and return to the landing pad if the battery falls below a certain level.
- The drone will return to the landing pad if it loses a connection to the ranger to prevent damage to the drone.

6.3 Security Requirements

- The drone will encrypt its stored data to ensure that if a drone is lost or stolen no sensitive information can be leaked.
- The drone will only accept commands from an authenticated entity to minimize the risk of hijacking or malicious actions, safety procedure will be followed in the event that the drone is unable to receive any commands after a set period (I.e. 30 seconds).
- The ranger will constantly be sending and receiving communication from the drone across a wireless signal, therefore all transmitted information will require at least 256-bit encryption to ensure no sensitive information is leaked to potential evildoers.

6.4 Software Quality Attributes

6.4.1 Reliability

- The drone will act out its set programming in an accurate and consistent manner and revert to default behaviour patterns when no instructions received.
- Ranger inputs received will be processed correctly and then execute in a manner which is expected. Unrecognised inputs or corrupt data received shall be ignored as only valid input is accepted.

6.4.2 Availability

- The drone will actively listen to inputs from the ranger even while performing other tasks as commands from the ranger are prioritised.
- The drone will not be impeded or overloaded by its subsystem's performance demands thus maintaining availability.
- The drone will drop a pin if the drone gets damaged.

6.4.3 Usability

- The application will have an intuitive UI with basic commands and modes of operation. Different modes of operations can be activated by the ranger without requiring further inputs as its executed autonomously

Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

7.1 Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

7.2 Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

7.3 Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

System Trace-ability Matrix

FR	Object Recognition Subsystem	User Application Subsystem	Drone Navigation Subsystem	Mapping Subsystem
R1				
R1.1				
R1.1.1		X	X	
R1.1.2		X	X	
R1.2				
R1.2.1			X	
R1.2.2			X	
R2				
R2.1		X	X	
R2.2				
R2.2.1			X	X
R2.3			X	
R2.4				
R2.4.1		X	X	
R3				
R3.1	X	X		
R3.2	X			
R3.3		X		
R4				
R4.1				X
R5		X		
R6		X		
R7		X		
R8		X		X
R9		X	X	
R10		X		