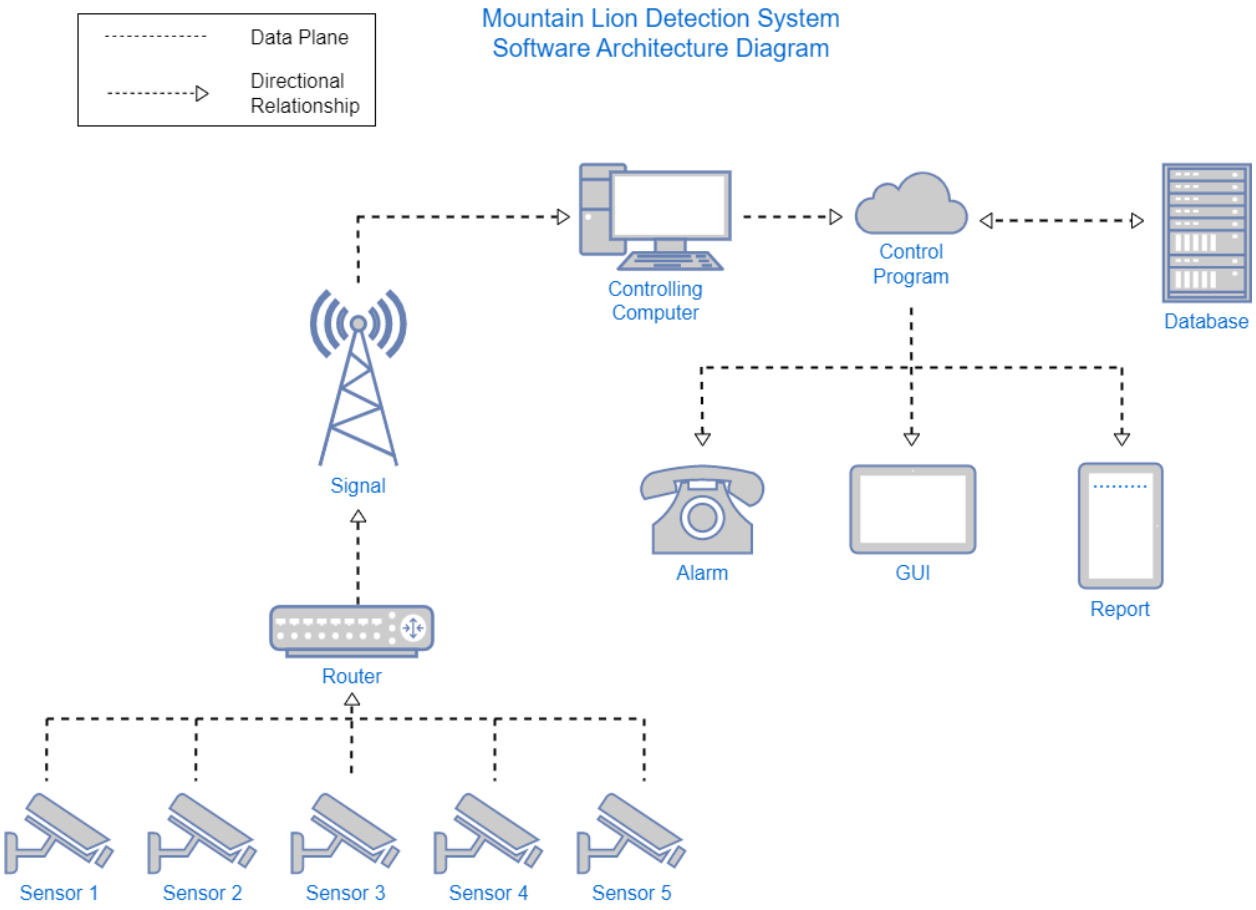
Mountain Lion Detection System

Daniel Aguilar, Yulianna Izaguirre, Conner Sommerfield

1. System Description

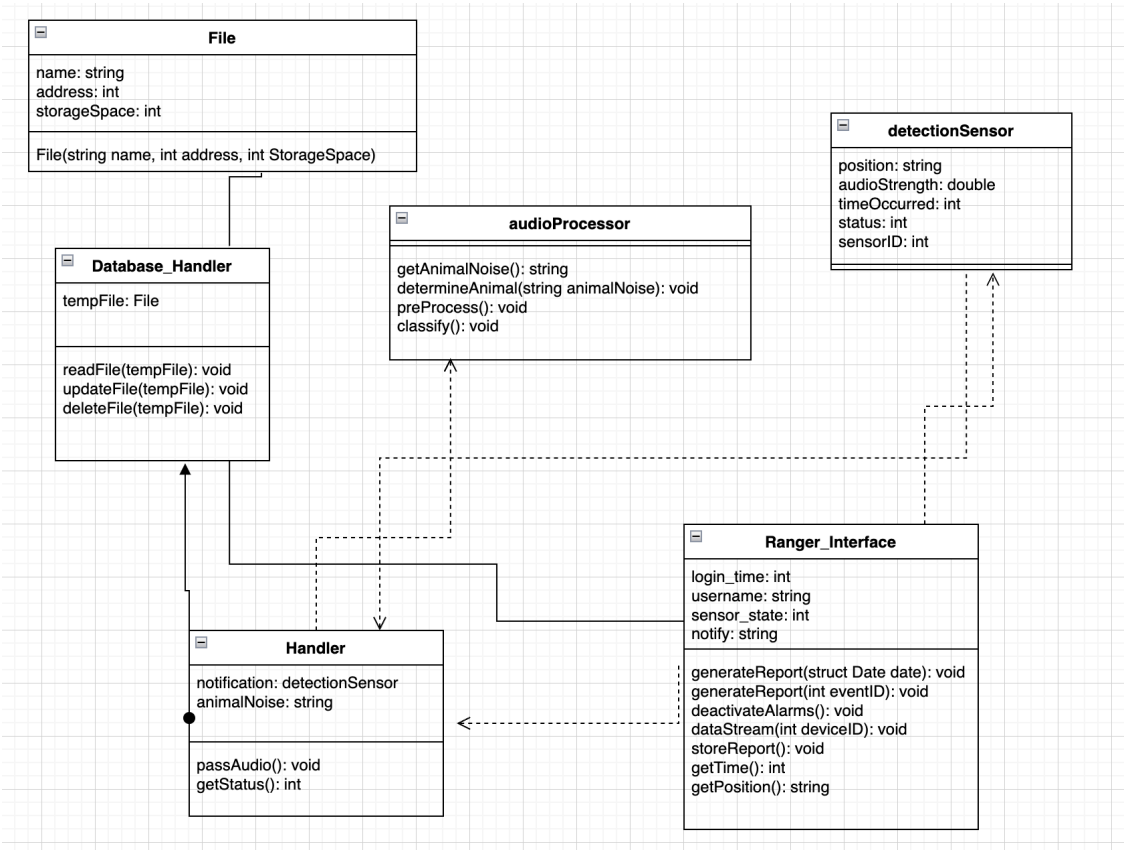
This is a mountain lion detection system, where the purpose is to detect any mountain lions nearby and if detected an alarm is sounded. This is needed to protect our civilians in parks all over California and allow our rangers to be more aware of what can possibly be out there in the parks. Though this system is for mountain lions it will also be able to detect different types of noises and animals that will directly notify the rangers to do what they need to do. This document will discuss precisely what requirements this system will need and how this system will be able to run smoothly.

2. Software Architecture Overview



The software system includes two main components which are the animal detection system and the controlling computer system. The animal detection system consists of five noise detection sensors. The sensors detect the strength, type, and location of the noise within a 3-meter radius. This information is sent to the controlling computer to be processed into a structured alter message. The Controlling Computer receives the alert message from the sensors based on both the type of animal noise and strength. The control program which is on the controlling computer has multiple functions for the ranges. Once an alert has been received from the animal detection system the control program will initialize the alarm to sound off. The alarm must be turned off manually by a ranger, otherwise it will continue to sound. The control program allows for the ranger to review and classify each alert message as definite, suspected, or false. The controlling computer saves all mountain lion alerts along with a summary of alert information to the database. Because the control program has access to the database it allows the rangers to request several types of reports, and display information with a graphical interface. Note that the control program will display any information such as the GUI and reports in the controlling computer.

UML Diagram



The handler will be the entry point of the program that runs the main watcher loop. This watcher loop will use an interrupt system to wait for notifications from the sensors (instead of something like polling) so that the rangers can also interact with the terminal while the server is waiting for a notification. These interactions may include things like generating reports. When an interrupt occurs, the program flow will be given to the appropriate function. The handler will have instances of the other classes as attributes so that it can call the necessary functions. It will have a state attribute that will change as the system goes through different stages such as all clear, warning, and emergency. It will also be in charge of giving the official notification to sound the alarm.

When the detection_sensor senses strange activity, it will send a flag to the main server along with its sensor id (this will be an integer that identifies which sensor is which). This causes the main server to query the sensor and record a brief audio. The sensor can give back information from its attributes such as the approximate position of the animal, the intensity of the sound recorded, and the time during recording.

The handler will pass the audio information from the sensor to an audio_processor class which will then perform a variety of functions to prepare the audio for classification of the animal. It will have a preprocess() function for cleaning up noise and then a classify() method which will return the believed animal type using a deep learning algorithm. For testing, a known animal noise can be passed to show that it is classifying animals correctly.

The Ranger_Interface class will have the functions that are tied to the buttons of the graphical interface. It will keep track of information on the ranger's login session through its attributes like the login_time, username of the ranger, sensor state (all of which to be shown on screen), and any notification to be displayed. It will provide a graphical link to the database functions for rangers to be able to request reports using a few different parameters, for example one could request reports based on the date it happened or by an event id (each event will have an id attached). There will also be a dataStream(int deviceID) method that allows the ranger to watch a certain sensor on screen at will by clicking the desired sensor.

The Database_Handler class will simply define CRUD operations for interacting with the database and functions that wrap specific SQL commands to easily get specific information without having to design requests. Each report will be an object defined by the File class, this

way the reports will have a consistent format. These files in the database can be created, read, updated, and deleted. These may be expanded to perform more specific and useful functions like SQL requests that get all the events for a certain day or to get the most recent event, etc.

3. Development Plan and Timeline

PROJECT TIMELINE
Approximate Budget: 3 million USD
APRIL 2023 - JULY 2023

Month of April (Starting April 3)
The engineers will first have to install all the appropriate hardware to support the system. We'll have one person for field setup such as the sensor and alarm devices, with a group of three for the mainframe server and network setup. The equipment will take about two weeks to configure to a basic level but will also need to be fortified to withstand various weather conditions such as rain, snow, and heavy winds, so we can expect hardware setup to take about three weeks. (ETA April 24 - May 1)

Month of May
Before starting with the higher-level development, we will give developers three weeks to get the most basic functionality running and do robust testing to ensure that our interfaces are working. The developers will set up unit tests to make sure devices have basic capabilities such as having the server successfully recognize a sensor message (without location), having dummy entries successfully saved to the database, and having the main server accept login information from rangers. With these foundations in place, higher-level development should be started by the beginning of June (June 1 - June 8).

Months of June and July
Developers will be able to use existing infrastructure and low-level capabilities to start plugging in higher-level functionality. This will include creating the classes and functions that will perform the needed functionality instead of using dummy data. The major aspects to work on will be the ranger interface, the main loop/sensor interrupt functionality, the CRUD operations to perform on the database using different querying parameters (event id, date, etc.), and the audio processing functionality which will involve extensive training to classify animals correctly. Expect

8 - 10 weeks to have the system working to an acceptable level for deployment (Delivery: August 5 - August 12)

Maintenance

We'll have to put measures into place to keep the system updated and add features as needed without complications. The system will be designed to accept new sensors or alarms with no changes to the existing infrastructure. Hardware checks on the devices will have to be performed every week to check for cracks or damage to devices as well as electrical issues. The audio processing will have to be fine-tuned to accommodate unforeseen circumstances and improve accuracy.