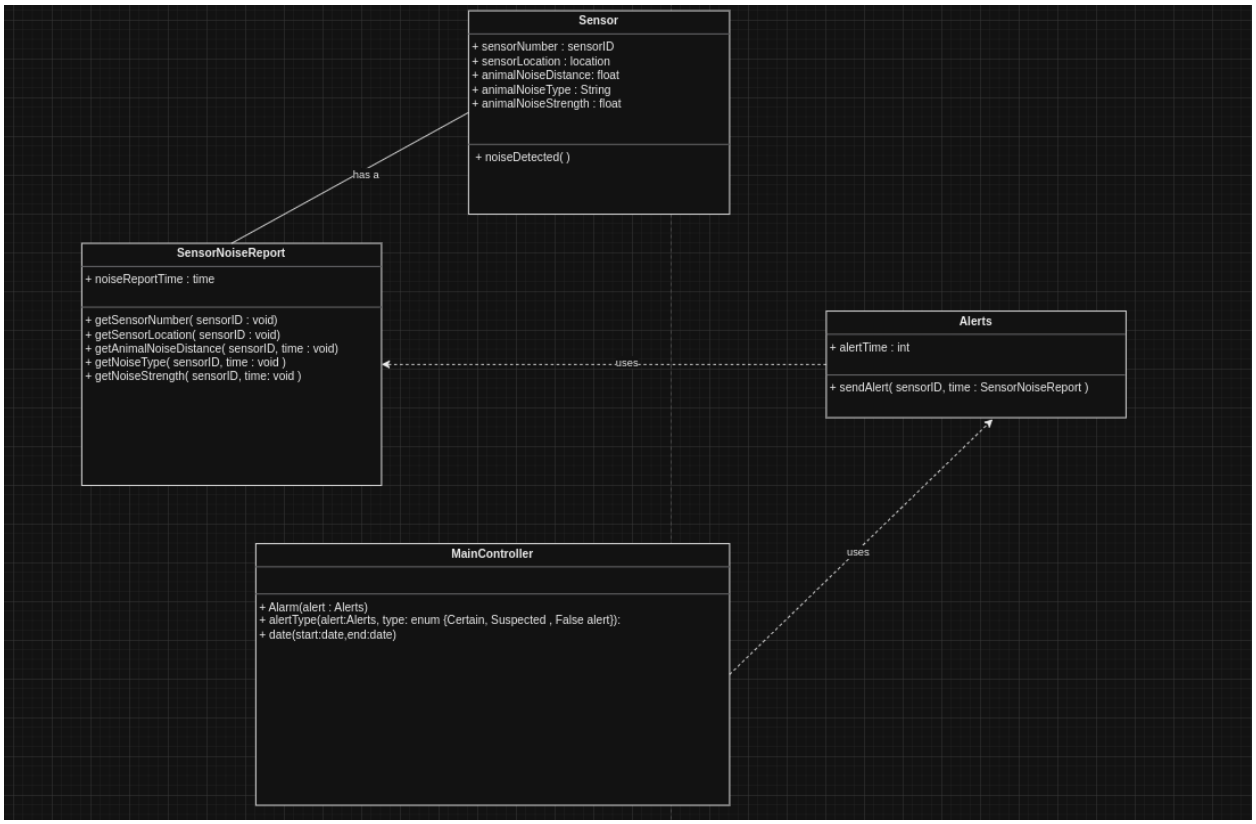
Assignment 2 Software Design Specification

Project Title: Mountain Lion Detection System

Team Members: Jai Sharma, Leo Findley and Todd Hutchison

System Description: The Mountain Lion Detection System is designed to detect Various types of animal noises and issue alert messages to the controlling computers of authorities to further analyze these alerts and also notify the rangers in the national park. This system uses the animal detection system developed by the company Animals-R-HERE and has the ability to detect noises within a zone by the strategic placement of noise detection sensors within that area. After detection, it sends the alert message to a main controlling computer on the basis of the strength of the voice and classifies these into different alert messages. This is a highly advanced system and can provide detailed information in the form of alert messages which include the type of noise, strength, and also the precise location to within 3 miles.

UML CLASS Diagram



Description of Classes

1. Class: Sensor

a.)Attributes:

sensorNumber : sensorID

sensorLocation : location

animalNoiseDistance: float

animalNoiseType : String

animalNoiseStrength : float

b.)Methods:

noiseDetected(void)

How it works: This class represents the sensors installed at various locations in the park and has attributes representing the IDs of sensors and their locations along with the type of noise they detect. The noiseDetected parameter detects the intensity of the noise and issues an alert message according to the intensity of the noise.

2. Class: Alerts

a.) Attributes:

alertTime : int

b.) Methods:

sendAlert(sensorID, time : SensorNoiseReport)

How it works: the Alerts class represents the alerts generated when the sensors detect sound. The attributes it has help us transmit each sensor's specific ID (sensorID), location (sensorLocation), time the alert was sent (alertTime), type of sound detected (animalNoiseType), intensity of the noise (animalNoiseStrength), and distance of noise with respect to the location of the sensor (animalNoiseDistance).

3. Class: SensorNoiseReport

a.) Attributes:

noiseReportTime : String

b.) Methods:

getSensorNumber(sensorID: void)

getSensorLocation(sensorID : void)

getAnimalNoiseDistance(sensorID, time : void)

getNoiseType(sensorID, time : void)

getNoiseStrength(sensorID, time : void)

How it works: the SensorNoiseReport class generates the data used in the Alerts sent to the Main Controller class. When the sensor detects a noise it generates a SensorNoiseReport with a time stamp. The Alerts class uses this sensor data to create an alert to the Main Controller class which uses an audible alarm to alert the ranger. The Main Controller class will also use the data from the alert to store varying amounts of information based on the time the alert was created.

4. Class: MainController

Methods:

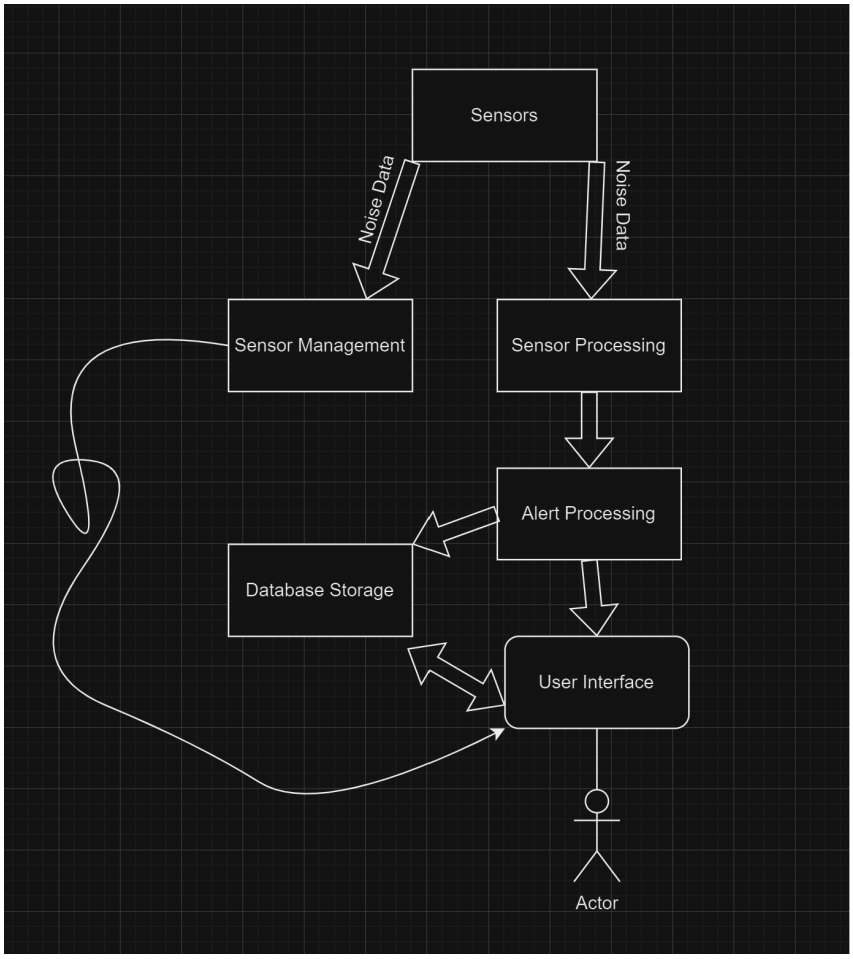
a.) Alarm(alert : Alerts)

b.) alertType(alert:Alerts, type: enum {Certain, Suspected , False alert}):

c.) date(start:date,end:date)

How it Works: This class is kind of like the brain behind our system and is responsible for issuing alerts and managing the alert system in general. It has methods to sound alarms and also helps classify them into certain, suspected and false alarms categories.

Architectural Diagram:



In the architectural diagram, our aim is to show how the high-level components of this system interact with each other.

Major components of the Mountain Lion Detection System:

1. Sensors: The sensors are key to detecting the lions as they are responsible for recording noises and sending them to authorities for classification.

2. Alerting System: The sensors relay the message to our alerting System which processes the noise and classifies them in order to know whether the mountain lions are present or not.

3. Storage: Stores alerts and alert notifications. The duration for alerts to be saved within the storage is 30 days and the alert summary older than 30 days but within one year.

4. User interface: The system also provides an easy-to-use user interface for the rangers to interact with the system in an easier and more efficient manner.

Interactions of the major components of the Mountain Lion Detection System

1. Sensors: They are responsible for sending noise to the sensor management.
2. Sensor Management System: Sensor management is responsible for sending to to alert processing system.
3. Alert Processing: This receives alerts from sensors and process them to classify them as serious alert or false alerts which it then sends to the database storage along with the user interface.
4. User interface and Data Storage: They are responsible for communicating with each other as the user interface requires reports and classification from database storage.

How the Architectural diagram works:

1. Firstly we would install sensors in different parts of the national park with unique IDs these sensors would detect animal noises and their strength which would then be sent to the sensor management.
2. The sensor management processes the noise and sends it back to the alert Processing system.
3. Alert processing further sends these alerts to database storage and these alerts are then shown on the user interface as alarms.
4. The Rangers can then use the simple-to-use user interface to their advantage to classify the alert as certain, suspected, or false alert. The Control computer sounds an alarm when an alert is received and rangers can manually turn them off using the user interface. The user interface also helps with generating reports which can be saved up to one year on the database.

Development Plan and Timeline:

Using the Waterfall Model

Requirement Analysis Phase (2 weeks)

- Gather detailed requirements from stakeholders, including park rangers and system users.
- Define the scope and constraints for the system.
- Create a comprehensive requirement specification document.

System Design Phase (4 weeks)

- Design the system architecture and define the technical specifications.
- Identify the technology stack and tools required for development.
- Create a detailed system design document and review it with stakeholders.

Implementation Phase (16 weeks)

- Develop the Sensor Management module for handling sensor data.
- Implement the Alert Processing module for analyzing and storing alerts.
- Create the Database Storage module for efficient data storage and retrieval.
- Develop the User Interface module for ranger interaction and reporting.

Testing Phase (6 weeks)

- Conduct unit testing for individual modules to ensure their functionality.
- Perform testing to verify that the system meets the system design specifications

- Perform integration testing to validate the interactions between different system components.

Deployment Phase (2 weeks)

- Prepare the system for deployment in the park environment.
- Configure the necessary hardware and software components for the system.
- Conduct a final round of testing to ensure a smooth deployment process.

Training and Documentation (2 weeks)

- Provide training sessions for park rangers and system administrators on how to use and maintain the system.
- Create comprehensive documentation including user manuals, technical guides, and troubleshooting documents.

Maintenance and Support (Ongoing)

- Establish a dedicated support team to address any issues or concerns raised by users.
- Regularly update the system based on user feedback and emerging requirements.
- Perform routine maintenance tasks to ensure the system's smooth operation.