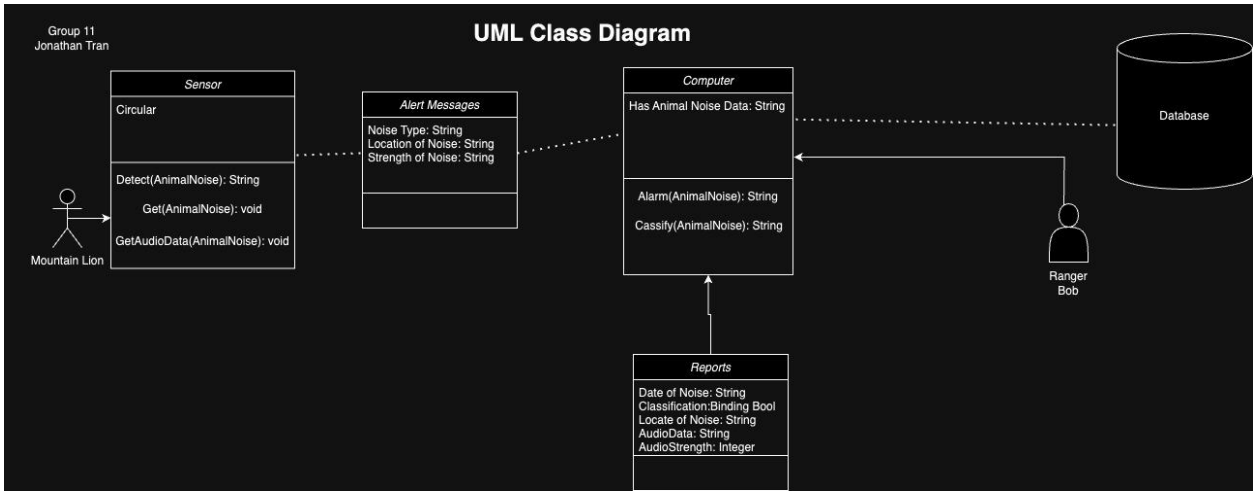
Mountain Lion Detection System Software Design Specification

Alexander Hixson, Eric Huynh, Jonathan Tran

System Description

A mountain lion detection system based on the Animals-R-Here animal detection system will use noise detection sensors to detect mountain lions within an area of 5 square miles. The sensors will be programmed to detect various types of animal noises, and alert messages will be sent to a controlling computer based on the type and strength of the detected noise. The controlling computer will be located in the park ranger station, and will sound an alarm whenever an alert message is received from the animal detection system. The ranger can then classify each alert as definite, suspected, or false, and request reports on mountain lion detections. The system will be designed to be easily reconfigured for other parks in the State of California.

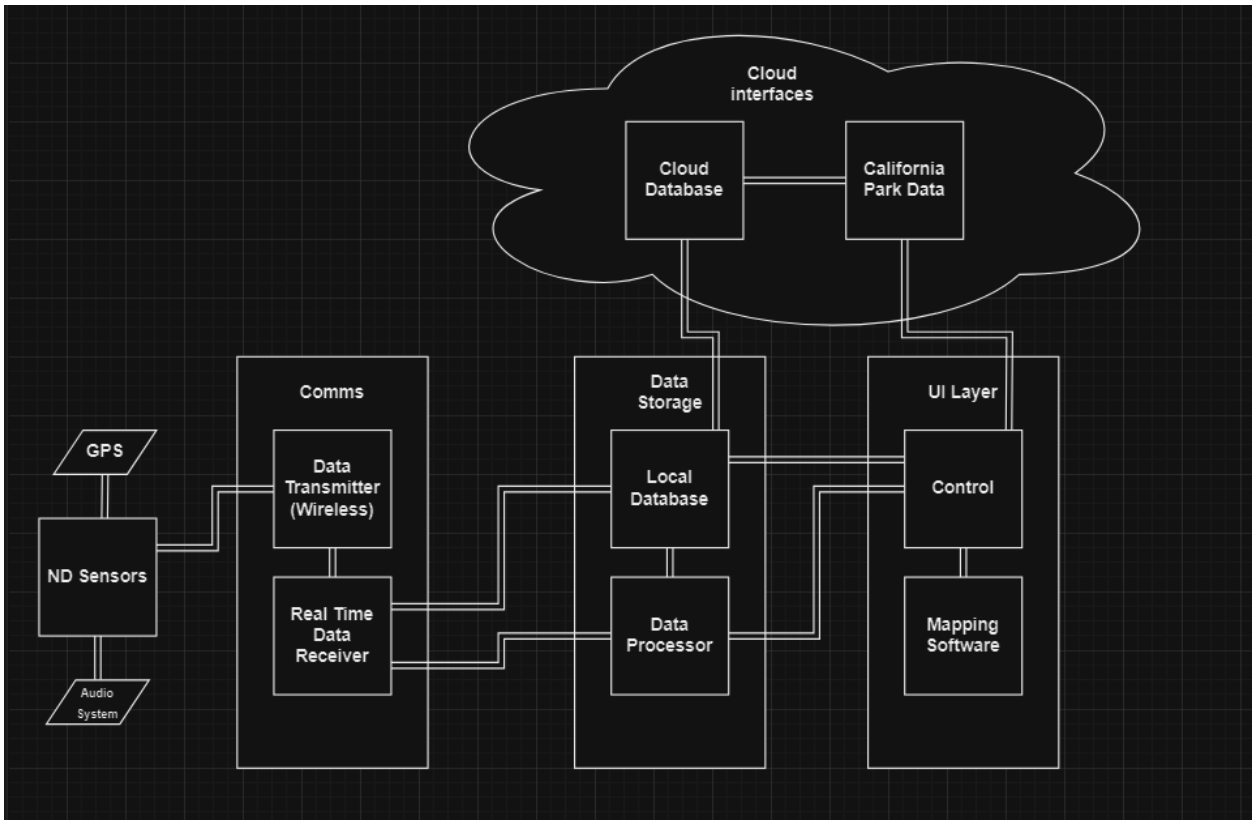
Software Architecture Overview



UML Class Diagram Overview

This UML Class Diagram depicts the various attributes and functions of this animal detection software system. When a mountain lion or any other animal noise is detected by the sensor, it sends the notification in the form of alert messages. The alert messages will be sent in the form of Strings and contain the noise type, the location of the noise and the strength of the noise. This notification is then sent to the computer where the data is stored. The Computer can also back up this data by sending it to the database. The ranger can access the computer to access the animal noise data. The computer can also send notifications to the ranger to notify the ranger of the animal noise as well as classify the animal noise by their given types. The Ranger also has access to reports generated from the computer that contains the date of the noise detection, the classification and location of the noise.

Software Architecture Diagram



Overview of Software Architecture Diagram:

- ND (Noise Detection) Sensors use GPS for location data and it interacts with Wireless Data Transmitter by transmitting detected noises and location data.
- Wireless Data Transmitter forwards data that's received from the ND Sensors to the Data Receiver at the ranger station.
- Data Receiver stores summarized data in the Local Database, and then sends real time data for analysis in the data processor.
- Data Processor takes in data from the Data Receiver and analyzes it to see if it needs to work with Control to trigger an alarm. Also allows for manual classification by rangers.
- Local Database gives and receives data from the Data Receiver, the Data Processor, Control, and the Cloud Database. It stores detailed alerts for 30 days, and summarizes older data up to a year.

- Control retrieves data about old alerts from the Local Database, stores classifications, and then takes other data for reports.
- Mapping Software takes in commands from the Control to visualize alerts on a map of the park.
- Cloud Database takes data in from the Local Database to include in statewide detection systems if needed. Then takes said data and gives it to the California Park Data to use.
- California Park Data receives data given from the Cloud Database, which was taken from Local Database, and interacts with Control to show the information visually or configure.

Description of Attributes:

- **Animal Detection System**

  This is derived from the Animals-R-Here company. It can cover up to 5 square miles using noise detection sensors, be customized to detect different animal noises, generate alert messages based on the type and strength of the detected noise and its location accuracy within 3 meters.

- **Controlling Computer**

  This computer will be located in the park ranger station, and it stores detailed data about mountain lion alerts for up to 30 days. For alerts that are older than 30 days and up to a year, it will save a summarized version of the alert.

- **Alarm System**

  This system will sound an alarm for every new alert that is received from the detectors. The alarm will continue until it is turned off by a ranger. Also, a new detection at a different location will reactivate the alarm if turned off from before.

- **Noise classifier**

  The ranger will be able to label each detection as definite, as in definite mountain lion, suspected, as in suspected mountain lion, or false, as in not a mountain lion.

- **Reports**

  There will be a date and classification report that displays all mountain lion detections sorted by date and classification. A location specific report is available too that shows the detections in a specific location with a specific sensor. There will also be a graphical report that highlights detections in a park map with areas within a 2 mile radius. Lastly there will be a ranger classification report that shows the detections that are classified per ranger.

- **Scalability**

The design will be modular and adaptable for deployment across all of California State Parks.

Description of Classes:

The Sensor class is responsible for identifying animal noises and sends data to the alert messages class when the Detect function returns “Mountain Lion.” The Sensor class is what communicates with the Animals-R-Here system and can request the audio bytes from that system.

Alert Message objects are sent by the sensor to the computer when a mountain lion is detected. Each alert message contains information about the noise type, location of the noise, and the strength/volume of the noise.

The Computer class has two functions. The first being the alarm that it sounds when an alert is present. The alarm will show the location associated with the alarm and will continue to sound until a ranger responds to the alarm. The second function of the computer class is classifying the alerts which the ranger will do by selecting one of the three options available: definite, suspected, or false.

The “Reports" class holds and organizes data that is used to generate reports. Each report will have data regarding the date of the alert, the classification the alert was given, the location associated with the alert, the strength or volume of the noise heard, and the actual audio byte that triggered the alert.

Development plan and timeline:

1. **Project Planning - About 2 weeks**
    a. Defining the scope of the project (Collective)
    b. Identifying the main partners and their roles (Collective)
    c. Outlining milestones (Collective)
    d. Brainstorming risks and avoidance tactics (Collective)
2. **Functional Analysis - About 3 weeks**
    a. Dive into requirements (Collective)
    b. Discuss with Animals-R-Here to get an idea of how the system works and the capabilities, with the idea of adaptation (Hixson)
3. **System Design - About 4 weeks**
    a. High level system architecture design (Huynh)
    b. Select hardware and software platforms to use (Hixson)
    c. Integration approach with existing software and detection systems (Hixson)

d. Design of user interface with ranger computer and control program (Tran)

e. Design of database for storing data, logs, and summaries (Huynh)

4. **Development - About 8 weeks**

    a. Develop software interfaces with animal detection system (Hixson)

    b. Develop alert processing mechanism and alarming system (Tran)

    c. Develop database storage and get functions (Huynh)

    d. Develop report modules with visualization on map and data (Hixson)

    e. Develop classification formula for sounds and alerts (Tran)

5. **Testing - About 6 weeks**

    a. Unit testing of individual components in the software (Hixson)

    b. Integrate actual animal detection system (Tran)

    c. Testing in the field with fake noises (Tran)

    d. Park ranger testing (Huynh)

    e. Bug fixes and feedback (Huynh)

6. **Deployment - About 2 weeks**

    a. On site usage at San Diego County Parks and ranger stations (Huynh)

    b. Configuration and calibration of system to park (Tran)

    c. Area designation for sensors and test their ability to collaborate (Tran)

7. **Future work - TBD**

    a. Mountain lion robots that look like lions and are indiscernible to an animal eye, that also works as a sensor and camera, that will infiltrate the mountain lion base of operations (Michelle Obama)

Projected Completion Date: About 6 months

Test Set 1

1. **Unit Test of Sensor Class**

   Objective: To ensure that the sensor class can detect and categorize animal noises accurately.

   Input: Animal Noise

   Method: The DetectAnimalNoise method of the sensor class is invoked with the given input.

   Expected Output: "Detected Animal Noise" (e.g., "Detected Lion's Roar")

   Failures Covered: Sensor inaccurately classifying the animal noise or not detecting it at all.

2. **Integration Test Between Sensor and Alarm**

   Objective: To ensure the sensor and alarm systems work cohesively when an animal noise is detected.

   Scenario: An animal (e.g., lion) walks within the Sensor Radius.

   Input: Detect(AnimalNoise) followed by Alarm(AnimalNoise)

   Method: The sensor detects the noise and the alarm system is triggered accordingly.

   Expected Output: The alarm sounds with an alert indicating the type of animal detected.

   Failures Covered: The alarm not sounding, incorrect animal detection, or a delay in the alarm system.

3. **System Test of Sensor**

   Objective: To evaluate the overall functionality of the sensor within the complete system.

   Scenario: An animal (e.g., lion) walks within the Sensor Radius.

   Input: Detect(AnimalNoise) followed by Alarm(AnimalNoise)

   Method: The system, as a whole, responds to the detected noise and triggers the appropriate alert.

   Expected Output: The alarm sounds, and the system logs the detection event.

Test Set 2

4. **Unit Test of Report Class**

   Objective: To validate that the report class can assemble a complete report from given data.

   Input:

   - Report Data
   - Date of Noise
   - Classification of Alert (e.g., "Lion's Roar")
   - Location of Noise (e.g., "Zone B")

- Audio Data
- Audio Strength (e.g., "80dB")

Method: The report generation method will be invoked using the provided data.

Expected Output: A full report containing all of the data provided, organized and complete.

Failures Covered: Missing data in the report, misclassification, or incorrect arrangement of data.

5. **Integration Test Between Computer and Reports Database**

   Objective: To validate the communication between the computer and the database when retrieving reports.

   Input: A request for a specific report (e.g., a date or type of animal).

   Method: The computer queries the database for the requested report.

   Expected Output: The report is fetched and displayed, containing all information associated with the alert.

   Failures Covered: Missing data, inability to retrieve a report, or the report not matching the criteria provided.

6. **System Test of Reports**

   Objective: To validate the system's ability to retrieve and display detailed reports based on user queries.

   Scenario: A Ranger wants to see a report from the previous month about a mountain lion detection.

   Method: The ranger initiates a search for the data of the alert.

   Expected Output: The ranger finds a report matching the date of the alert. The report contains all information about the alert, including the location, classification, volume, and a clip of the noise.

   Failures Covered: The system failing to retrieve the desired report, incomplete or inaccurate report data, or a delay in report generation.

Database Management Strategy

We are going to use SQL for this system, as SQL offers better consistency and relationship integrity which is important for our system. It is less flexible when it comes to being scalable, but in this system there is no need for scalability as all data is structured and the need for complex querying and data integrity. A single database is sufficient, given that the size and scope of our system isn't too complex or big. Multiple databases may be needed if different departments need isolated data access or for scalability. The data is logically split between Rangers, Reports, and Animals, as these align with the distinct data categories and relationships. We could use NoSQL as an alternative for future scalability and flexibility in data modeling where we need to apply this to a different area and different animals for different parks. As stated before the trade-off between SQL and NoSQL is consistency with SQL and relationship integrity with structured data, while NoSQL offers more scalability and is better for unstructured data.

The Rangers table includes personal and operation data about rangers, so SQL is fit for this as they have strong relationship and integrity constraints that handle structured data.

Reports Table contains detailed report data and SQL works well as their query power is very useful for generating complex reports.

Animals Table lists animals and their sightings count, this is simple and straightforward structured data which is good for SQL.

Architecture Diagram

Rangers Table

Ranger 1	Ranger ID; Name; Username; Password; Contact number; Role; Date of Account Creation; Login log; Alert History; Management history; Report history; Edit history
Ranger 2	Ranger ID; Name; Username; Password; Contact number; Role; Date of Account Creation; Login log; Alert History; Management history; Report history; Edit history
Ranger 3	Ranger ID; Name; Username; Password; Contact number; Role; Date of Account Creation; Login log; Alert History; Management history; Report history; Edit history

Ranger n	Ranger ID; Name; Username; Password; Contact number; Role; Date of Account Creation; Login log; Alert History; Management history; Report history; Edit history

Reports Table

Report 1	Date of Noise; classification; Location of Noise; Audio Data; Audio Volume
Report 2	Date of Noise; classification; Location of Noise; Audio Data; Audio Volume
Report 3	Date of Noise; classification; Location of Noise; Audio Data; Audio Volume
Report n	Date of Noise; classification; Location of Noise; Audio Data; Audio Volume

Animals Table

Animal-Type	**Number of Citings**
Mountain Lion	1
Tiger	3
Deer	5
Hawk	2