Database Management Strategy: Mountain Lion Detection System

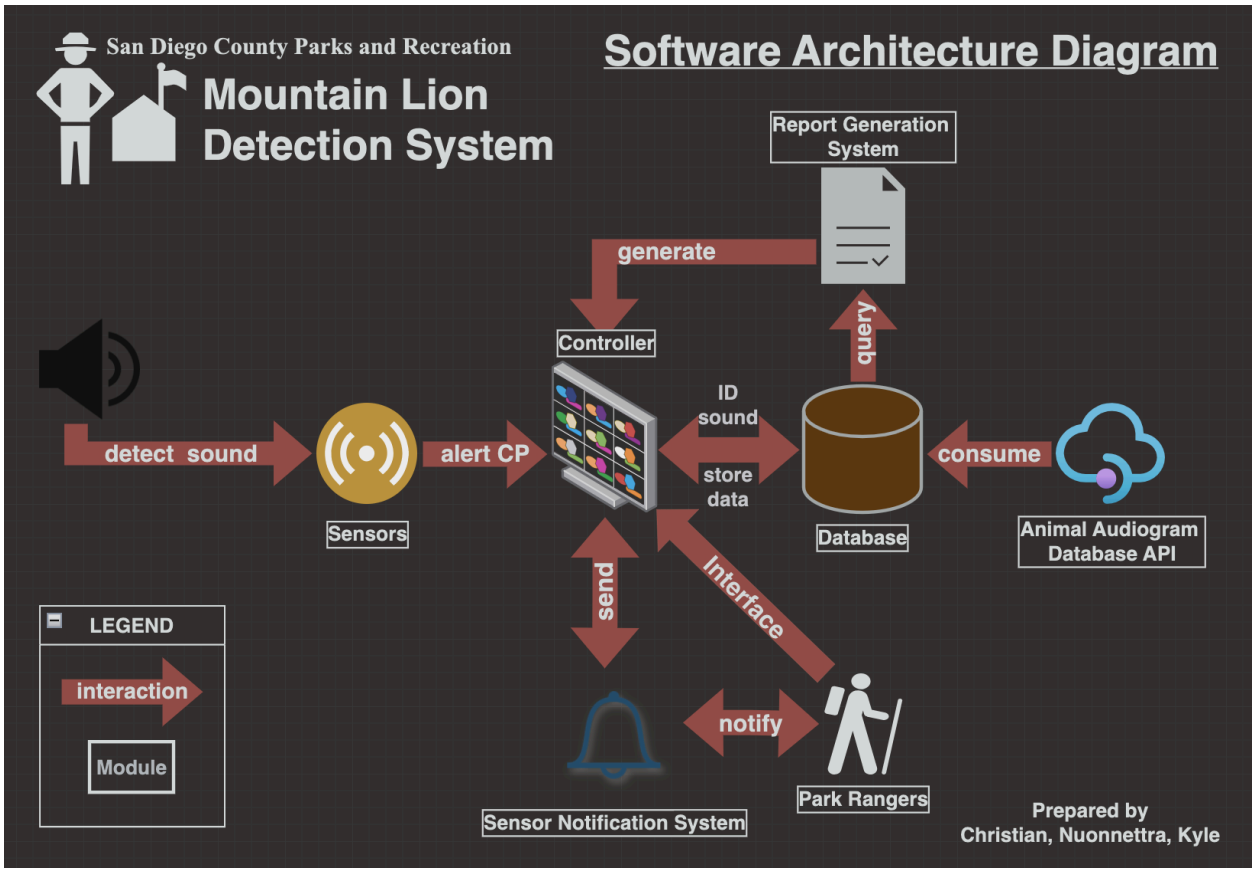
Prepared by Christian Byars, Nuonnettra Kanzaki, and Kyle Camposano

07 November 2024

1. System Description

The Mountain Lion Detection System (MLDS) is dedicated to its hikers and park employees to detect possible dangerous animals on the mountain trails and to notify nearby recreational hikers. The system consists of detection sensors covering up to 5 square miles to pinpoint the approximate location of the animals within 3 meters. Upon detection of mountain lions, the system notifies park rangers, stores all mountain alerts in local storage for up to thirty days, and stores a summary of alert information for up to one year. The park rangers utilize the best course of action to notify nearby hikers of the mountain lion threat within the proximity. The system's control program allows park rangers to access the stored data at any time, which can then be used to generate reports on the detections of mountain lions. The development of the Mountain Lion Detection System is crucial for the safety of the general public and park employees at San Diego County Parks and Recreation and possibly other California State Parks. This document is separated into six categories: Software Architecture Diagram and its descriptions, description of updates on the original Software Design Specification, Data Management diagram and tables, alternative Data Management and Organization, Trade Offs, and Team Members' Responsibilities.

2. Software Architecture Diagram



2.1 Descriptions of Software Architecture Diagram

1. **Module:** Sensors
    a. **Description:** The sensors are responsible for detecting animal sounds immediately and accurately at their designated location. Each sensor operates in a 5 square mile radius and pinpoints the sound's approximate location within 3 meters. The sensors are placed throughout the park near the hiking trails to ensure the safety of the hikers.
    b. **Interaction:** The sensor is event-driven and is only triggered when animal sounds are detected. The sensor then relays this information and alerts the Control Program to classify the sound, notify the hikers, and store the sound data.

2. **Module:** Sensor Notification System
    a. **Description:** The sensor notification system is responsible for making the possible threat of mountain lions apparent. This is done by sounding off an alarm once an alert message is received.
    b. **Interaction:** The Sensor Notification System will sound off the alarm through the controller once it has the appropriate data. This will then notify park rangers about the threat of one or multiple mountain lions.

3. **Module:** Animal Audiogram Database API
    a. **Description:** The Animal Audiogram Database contains data regarding the sounds of various animals. This allows the system to accurately detect mountain lions and not the sounds of other animals.
    b. **Interaction:** The audiogram database feeds its data to the primary database.

4. **Module:** Report Generation System
    a. **Description:** The Report Generation System provides written documentation of various data the detection system picks up. This includes reports showing all mountain lion detections, the sensor triggered, a graphical report of detections within two miles of the park, and detection classifications by ranger.
    b. **Interaction:** The report generation system needs the data from the database in order to provide reports. Once the reports are generated, they will be accessible via the control program.

5. **Module:** Controller
    a. **Description:** The controller is the most crucial component of the software architecture. It is where most of the information and interaction go. This is where the data about the possible mountain lion will be sent, which in turn, the control program will send to the database. The controller is also where the park ranger can interact to use other components of the program. This includes the module, report generation system, database, and sensor notification system.
    b. **Interaction:** The park ranger can generate specific types of reports, view notifications, turn on or off the alarm system, and put in a classification (Definite, Suspected, or false) for a report.

6. **Module:** Database
    a. **Description:** This component is where all of the data is stored or will be stored. This is where all the history reports are, to which all the controller and the report generation system have access for their own functioning.
    b. **Interaction:** The database depends on the control program to receive the data and send the data back to the control program. This component will store according to the standard rule of which to have a full report for 30 days and a summary of the report for up to a year and then discard the reports once it passed the set time period.

7. **Module:** Park Ranger

a. **Description:** This is the verified user which is the authorized park rangers that will have control and access over the system.
b. **Interaction:** The user can go to the controller to access the database, the reports, and the alert notification system.

3. Updates on the Software Design Specification

This section describes the changes made to the original version of the Software Design Specification as the project timeline progresses. The changes were minimal, yet necessary to ensure clarity and flow for the test plan phase.

Software Architecture Diagram: There is no major change or update on the diagram for the software architecture. The main components which interact with the database remain the same, and those connections are identified throughout the draft of data management and organization strategy.

UML Class Diagram: Reports Class now shares a dependency with Database Class. This change is necessary as the Reports Class needs to directly access the stored data (classifications, location, and animal sounds) in the Database Class. Without this change, there's no bridge to the gap between the Database and Reports Classes in the test cases.

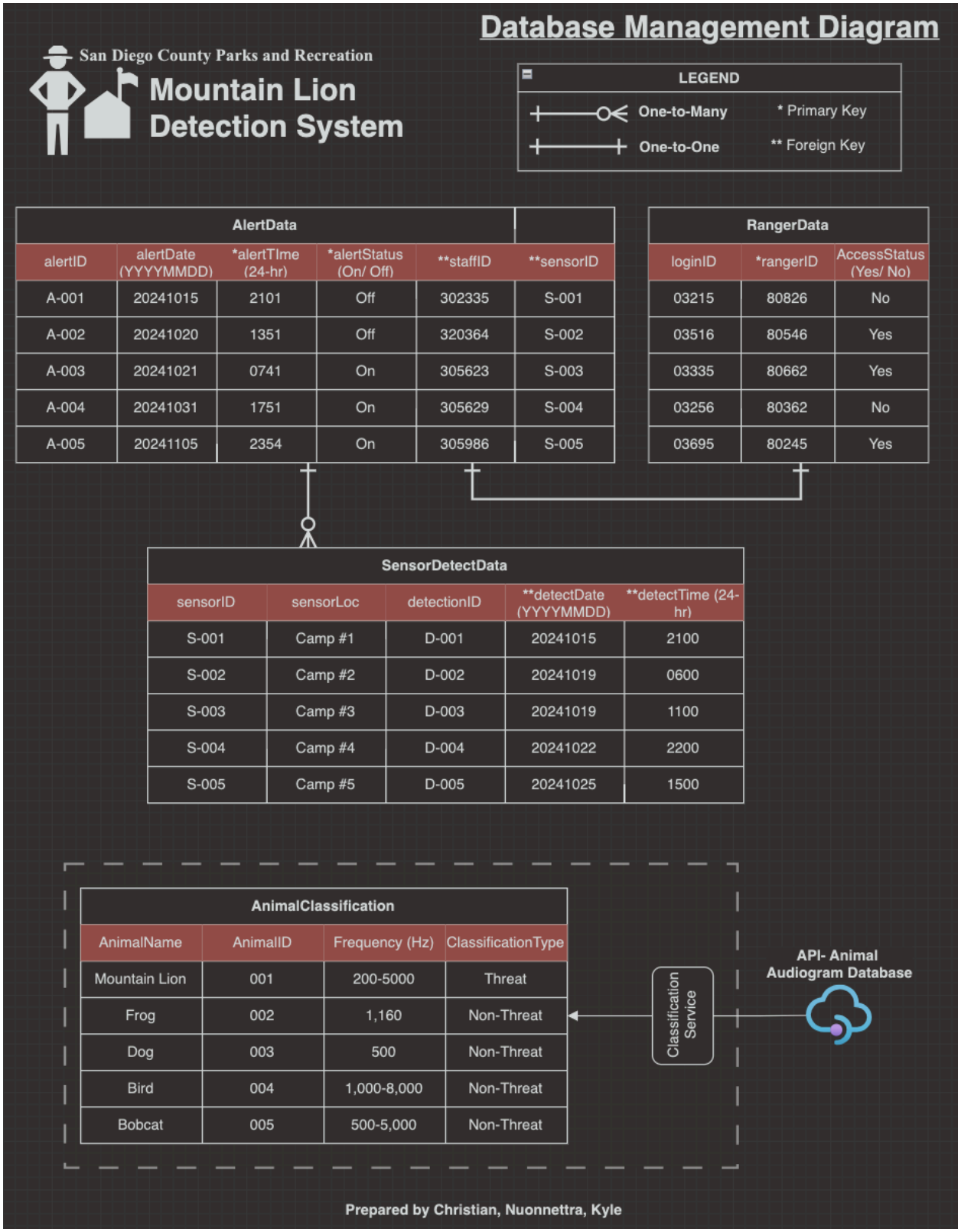
Database Class: The description of Database Class now matches the change on the UML Class Diagram. The Database Class specifically mentions the interaction (dependency) of the Reports

Class to the Database Class. For the SWA, the attributes and operations for this class remain the same. In the initial design, the data storage and retrieval from and to the database were considered, so there's no changes necessary during the data management phase.

4. Database Management

This section details the strategy of the database management of the Mountain Lion Detection System. The purpose of the database is to facilitate the stored animal sounds, alerts, and detections. The strategy consists of utilizing the structured approach of the SQL database as this highly supports the proposed structure of the MLDS database. With SQL, data storage, fetching, retrieval, and classification are more efficient and scalable which directly improves the overall performance of the software system. The database management comprises one database, organized into four data tables. Data tables are split into their respective groups to keep relevant information together for convenience, scalability, and efficiency. This section breaks down each data table to its purpose, components, interactions, and description.

4.1 Database Management Diagram



4.2 Data Tables

Data Table 1: AnimalClassification

Components: AnimalName, AnimalID, Frequency, ClassificationType

Interactions(s): Classification Service (API- Animal Audiogram Database)

Description: The AnimalClassification data table stores and classifies the animal sounds received by the database. The identification of the animal sounds relies on the Classification Service of the API external to the database. The soundData object the database receives is compared with the data fetched from the API service. The animal sound and its respective classification are only stored if the soundData and fetched data match. The AnimalClassification data table consists AnimalName, AnimalID, Frequency, ClassificationType as these are the relevant information to identify the classification of an animal sound. It's necessary to have this table to reliably and efficiently access the classification of all animal sounds in the park. In the absence of this data table, the animal sounds may not be processed or inaccurately identified.

Data Table 2: SensorDetectData

Components: sensorID, sensorLoc, detectId, detectDate, detectTime

Interaction(s): Data Routing and Alert Service

Description: The SensorDetectData data table stores data regarding the detection of animals. Once an animal is detected by a sensor, that information will be sent to the controller in order to display an alert. There will be an alert for any and every animal that is detected, but mountain lions have a separate alert to indicate such. The data table consists of sensorID, sensorLoc,

detectId, detectDate, and detectTime, which are all necessary pieces of information for the controller to display in the case of an emergency.

Data Table 3: Alert Data

Components: Alert ID, AlertDate, alertTime, alertStatus, staffID, SensorID

Interactions(s): Alert System

Description: The AlertData table concludes the information of a specific alert. This table stores the ID, date, time, status (manually input from the authorized user), staff ID (authorized user), and lastly the sensor ID. This table is crucial to the whole system because this is where the authorized ranger will finally have access and see the data. This portion of the table ensures that the data that is going to be displayed to the authorized user is useful and understandable. This table is connected to the SensorDetectData which is where it gets the time and date and the sensor ID. It is also connected with the Ranger Data to see if the user is the authorized personnel and to allow access to change the alertStatus.

Data Table 4: Ranger Data

Components: loginID, rangerID, AccessStatus

Interaction(s): Verification for authorized rangers

Description: This table is the smallest table of this data management but is a vital table despite its size. This is where the authorized ranger can have access directly to the database to manually adjust the AlertStatus in the Alert data. This table checks for verification to determine whether the login user is the authorized personnel or not. This is more of a security layer for the database.

This table consisted of loginID and rangerID and this will be used to determine their verification. This table is connected to the Alertdata as it will then allow the authorized user access.

5. Alternative Data Management and Organization

This section details the alternative designs/ approaches to the data management and organization of the Mountain Lion Detection System. The alternatives are outlined with their respective advantages and disadvantages compared to the proposed data management and organization: SQL structure, consisting of one database and organized into four data tables.

noSQL (Redis)

Redis is an open-source database that stores data inside of RAM as opposed to a disk. Due to data being stored in RAM as opposed to disk, Redis is much faster than SQL in many cases. To prevent data loss from happening, Redis uses a built-in function that dumps files on the disk on certain occasions. Redis also stores a wide variety of data types, which would be crucial for the Mountain Lion Detection System, which has a wide variety of data that is stored. Despite some of the benefits of Redis, it still does come with its downsides. Despite the many data types Redis supports, it would still likely be slower than SQL. The Mountain Lion Detection System has a wide variety of data that all interconnects, making SQL the more appropriate way to manage data.

Single main table

Pros: The alternative of utilizing one single main table for all relevant fields and records is beneficial because of its simple properties. Data storage, fetching, retrieval, and classification are

simple, resulting in minimal queries. Accesses with this design require one single query per access (in a smaller data table). Minimal query means reduced processing time, which is beneficial for the software requiring immediate classification of an animal sound. Another benefit of one single main table is that it doesn't require dependencies (or at least minimally) as all data can be accessed through a simple query.

Cons: While there are benefits to utilizing one single main table for all data, it also has disadvantages that are particularly crucial to consider regarding the integrity and structure of the database for the MLDS software. One downside is the possible extensive damage(s) of data error or data mismanagement. This means that a set of data that is handled incorrectly or corrupted can be detrimental to the entire database. Certainly, the larger data table means points of failures/ errors increase as well. Some of the possibilities include, but are not limited to incorrect data type, data inconsistency, corrupted soundData, or misclassification of soundData. The underlying issue with this design is that it's rigid, which is not beneficial for the database of the MLDS software. Another downside to utilizing one single main table is the lack of scalability and maintainability for future larger data (and future implementations of the same database with a new software similar to the MLDS). As the single data table gets larger, the processing time of the data queries takes longer as it needs to sift or traverse through all data, most of it may be irrelevant to the query. Both downsides (effects of data error and lack of scalability/ maintainability) are too risky, and while there are safety measures that can be implemented, a simpler solution is to consider another design that's more suitable for the MLDS software's database.

Multiple tables, duplicate data

The benefits of having multiple tables with duplicate data show how each of those duplicate data moves between tables. This should be implemented because some data will be used concurrently between different tables and allow for faster access time. The drawback would be the memory allocation would take up more space. As the data gets larger, so does the memory allocation. This is especially detrimental to the implementation of the database for the MLDS software as it requires efficient and reliable queries. It's also crucial to consider the scalability and maintainability of the data organization as this directly affects the database performance, which ultimately results in the total performance of the software.

6. Tradeoffs

For the implementation of the software for the Mountain Lion Detection System, the most efficient and structured data management and organization is SQL with multiple tables, utilizing minimal data duplicates and primary/ foreign keys. There are multiple alternatives to data management and organization, some of which are detailed in the previous section (3. Alternative Data Management and Organization). Each of the alternatives is listed with its advantages and disadvantages. Utilizing SQL for the database benefits the software as it is structured which ensures reliability and consistency. These traits are highly necessary to provide data integrity. In addition, this is where the use of primary and foreign keys comes in certainly handy. The keys directly relate to the certain fields that are needed in their respective tables. This is where the usage of multiple data tables is beneficial. It compartmentalizes relevant data, with minimal memory overhead. Unlike the alternative of using multiple tables and duplicate data, minimizing data redundancy takes less space which promotes more efficient queries. While there

are alternative implementations for MLDS's data management and organization, the most efficient is SQL, splitting data into their respective data tables.

7. Team Members' Responsibilities

Christian Byars

- Responsible for completing the sensor detection data table along with its description
- Responsible for writing the "noSQL (Redis)" section
- Contributed with correcting spelling and grammar

Nuonnettra Kanzaki

- Wrote half of the Multiple tables, duplicated data
- Wrote alertData overview
- Wrote rangerData overview
-

Kyle Camposano

- Responsible for drafting the format of the Database Management Strategy and its diagram.
- Responsible for writing the overview of the following sections:
    - Data Management
    - Alternative Data Management and Organization
- Responsible for completing the following:
    - one of the data management/ organization alternatives (single main table).
    - Tradeoffs section (section 4)

- Responsible for changing the necessary updates on the Software Design Specification concerning the Verification Test Plan and Database Management Strategy.