San Diego County Parks and Recreation Department Mountain Lion Detection Software Design Specification

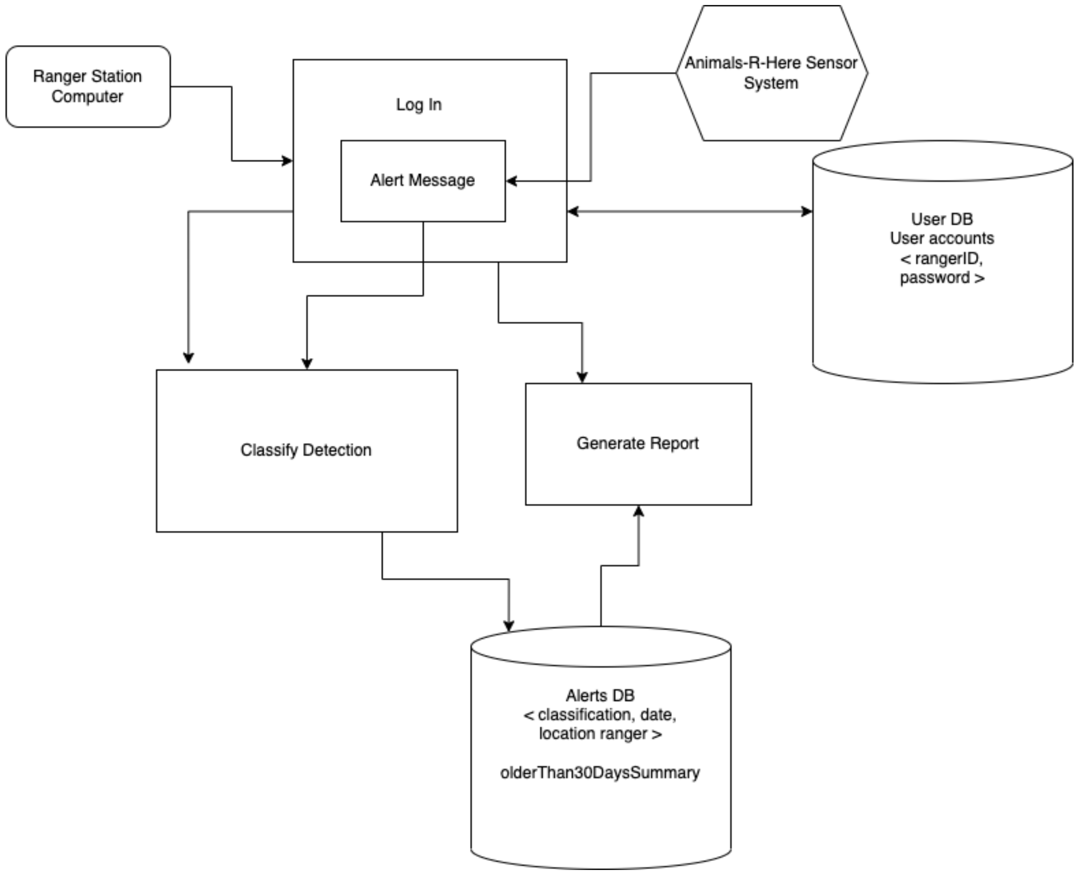
Prepared by: Gabriela Calvo, Shane Ahmad, Daniel Ha

System Description

Our system is designed for park rangers employed by the San Diego Parks and Recreation Department to use when handling data on mountain lion activity within a given park area. Our control program will facilitate receiving, classifying, and storing alerts sent from Animals-R-Here sensors that are placed throughout the parks and pre-programmed to detect mountain lions. Upon logging in, users will have the option to classify a detection or generate a report. Alerts received from the sensors will contain data on type, strength, and location of the detection but all alerts must also be reviewed by a ranger and classified as either definite suspected or false based on the probability that an actual mountain lion was detected. Within generating reports, the user has four options. The four types of reports are of detections organized by classification and date, detections organized by ranger, detections at a specific location, and detections indicated on a map that extends 2 miles beyond park boundaries.

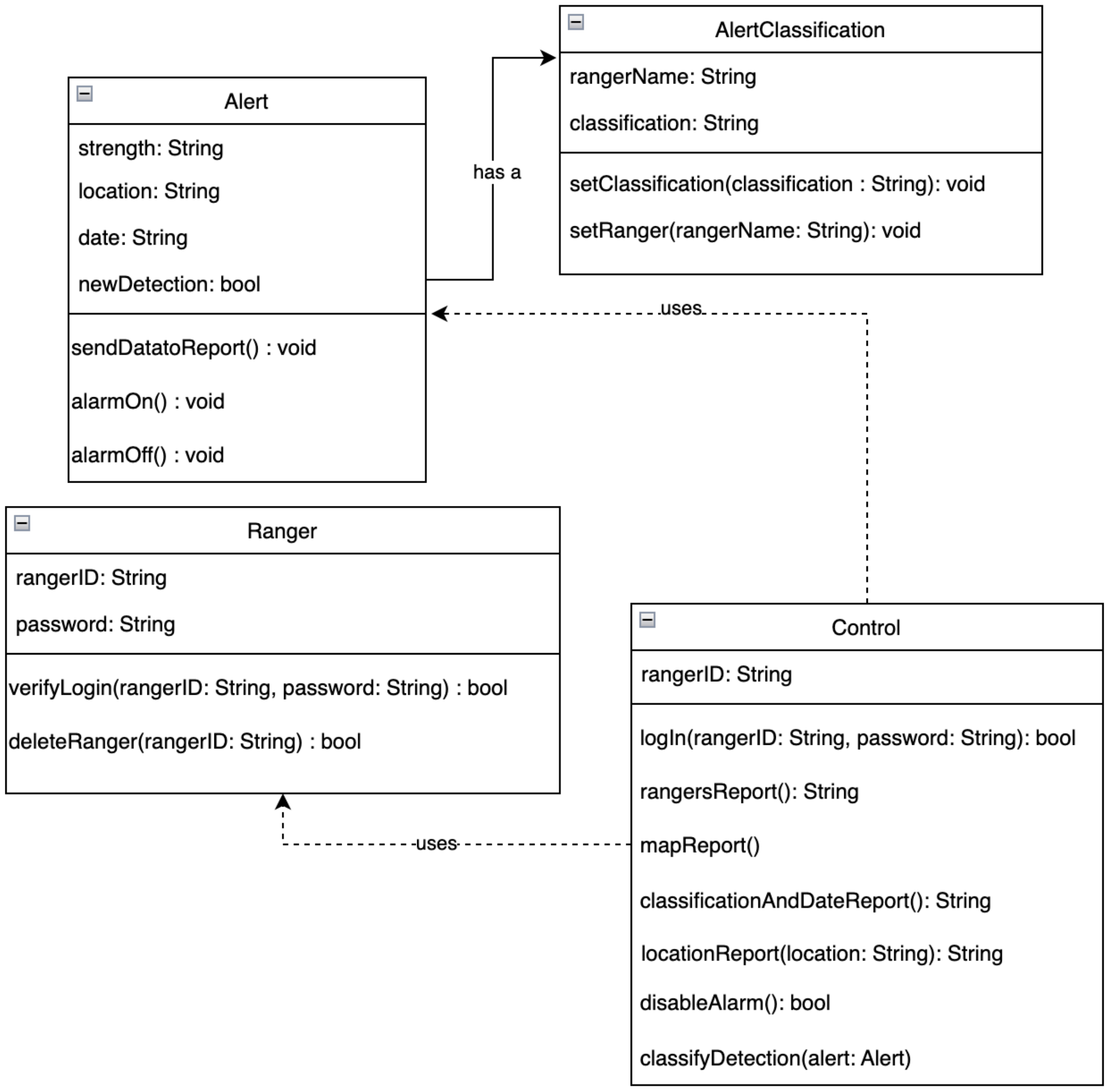
Software Architecture Overview

Software Architecture Diagram



The design of our system is centered around the two main types of actions that the user performs; classifying alerts and requesting reports. For our purposes, the user will always be a ranger on a ranger station computer which is represented by the box in the top left. That box connects to our login page which has a sub-category representing the case where the user is responding to an alert message. The alert message depends on data that is sent from the Animals-R-Here Sensor system which is represented in the top right corner. Whether or not they are responding to an alert, the user will have to log in with their rangerID and password and that is where the login page will communicate with the User Database to verify their credentials, indicated by a double ended arrow. From there, the user will either classify a detection or generate a report both of which will interact with the Alerts Database that stores the data for previous detections. Classifying a detection sends data to the Alerts Database, which is depicted by a one directional arrow towards the database. Generating a report retrieves data from the Alert Database, depicted by a one directional arrow from the database.

UML Class Diagram



AlertClassification
rangerName: String
classification: String
setClassification(classification : String): void
setRanger(rangerName: String): void
Alert
strength: String
location: String
date: String
newDetection: bool
sendDatatoReport() : void
alarmOn() : void
alarmOff() : void
has a
uses
Ranger
rangerID: String
password: String
verifyLogin(rangerID: String, password: String) : bool
deleteRanger(rangerID: String) : bool
uses
Control
rangerID: String
logIn(rangerID: String, password: String): bool
rangersReport(): String
mapReport()
classificationAndDateReport(): String
locationReport(location: String): String
disableAlarm(): bool
classifyDetection(alert: Alert)

Description of Classes

Alert

The Alert class is designed to retrieve information from the third-party sensor that detects animal noises. This class includes the functionality for sending data to generate reports as well as controlling the alarm that goes off when a detection is received from the sensors.

AlertClassification

The AlertClassification class is an extension of the Alert class which adds variables for classification and the name of the ranger that reviewed the detection. This completes the information necessary for a detection to be included in a report, so each Alert should have an associated AlertClassification.

Ranger

The Ranger class represents all of the individual ranger profiles stored in the User Database depicted in the Software Architecture Diagram. This class will mostly be used to verify the rangers when they log in.

Control

The Control class includes the functions the rangers can perform on their computer. This class facilitates the user storing information related to the detections and retrieving reports.

Description of attributes

The Alert class has four attributes. The first attribute, animalNoise, is a String that defines the kind of animal that made the sound detected. The strength String attribute measures the intensity of the sound. The location String attribute is defined by the location of the sensor when it detects a sound. The date String attribute is determined by the date when the sound is detected. Finally, the newDetection attribute is a boolean variable that is set to true whenever a new detection is received from the Animals-R-Here system. The AlertClassification class builds onto the Alert class with two additional attributes, one for the classification of the detection set by the ranger and one for the ranger that reviewed the detection, both of which are String type. The Ranger and Control classes both use a String attribute for rangerID that is unique to each ranger. The Ranger class contains an additional String attribute for password, which will be used alongside rangerID when the user logs in.

Description of operations

The Alert class has three functions. The first is sendDataToReport() which returns the information of a given detection to be included in a report. The class also has two operations associated with the alarm system, alarmOn() and alarmOff(). The alarmOn() function is triggered whenever the newDetection bool variable is true. The alarm will turn on and stay on until a ranger manually turns it off using the alarmOff() operation. The AlertClassification class has two methods. The first is setClassification() which allows the ranger currently at the computer to review the detection and classify it as “definite”, “suspected”, or “false” according to how likely it was to be an actual mountain lion detection. The setRanger() function will associate the current ranger with the detection so it can be shown in the reports. logIn(String rangerID, String password) is used in the Control class to allow a ranger access to the database so they can request reports. classifyDetection(Alert alert) is used to select an alert to classify. There are four functions within the Control class that generate reports. The classificationAndDateReport() function retrieves a report that includes all mountain lion detections organized by date and classification. The locationReport(String location) takes one String argument indicating location and returns to the user a report including all mountain lion detections at a specific location. The requestMap() function provides a map of the park and the surrounding area indicating all stored detections. Lastly, rangerReport(string ranger) takes one String argument indicating a specific ranger and returns to the user all detections reviewed by that ranger. Elsewhere in the class, disableAlarm() is used to stop the alarm from making sound. The Ranger class includes two functions managing a ranger's ability to use the system. deleteRanger(String rangerId) is used to stop allowing a particular ranger to access the database. Finally, verifyLogIn(String rangerID, String password) checks to ensure that a ranger's ID and password match.

Development plan and timeline

The primary function of our system is to detect certain animals with sensors and send alarms to the ranger's computer. That said, databases are required to be able to store the information regarding the alerts and reports but also for the ranger's login information. We went about our development by splitting up tasks by notable elements. To begin, we decided that Daniel will be tasked with everything ranger related. To be more specific, Daniel will be responsible for the log in page, ranger's information, and the database that goes along with user accounts. This should take approximately 3 to 4 weeks to complete so his deadline will be in 4 weeks. Gabriela will be responsible for the reports component of our system. While it may seem like one class, this section is quite hefty as there are many methods and different types of reports that can be requested. Keep in mind that the reports sections should work alongside the Alerts Database especially considering reports will only be saved for 30 days. That said, Gabriel will be mainly responsible for the Control class and work with the Ranger class from Daniel. Considering that Gabriela is also responsible for the report information database, this will most likely mean she will need more time. Likewise, Gabriela's task will take approximately 3 to 4 weeks to complete. To bring everything together, Shane is responsible for the Alert and Alarm systems as they work hand in hand. The component might seem simple but is crucial to making our system work as intended. Shane needs to make sure his code works with the alarm system and sensors. It is also worth mentioning that not only does the alarm system work directly with the alert system, but interoperability should be prioritized to make sure it works seamlessly with the reports database. Since Shane does not need to implement a database, he does not require as much time and should be able to complete his task in 3 weeks. We will make sure to have weekly meetings to make sure we reach our milestones each week. Our client has given us two months to get our system up and running so we should have plenty of time to test and make sure our system works as intended.