Mountain Lion Detection System

Prepared by: Salvador Juarez-Ochoa, Steven Martin, Phillip Vo

DESIGN SPECIFICATION

System Description

The mountain lion detection system will be used by the San Diego County Parks and Recreation Department to detect animals and their proximity and determine how likely the noise detection came from a mountain lion. Sensors will be placed in an area with a radius of 5 miles to detect noise and send detection information and sound an alarm to a central computer. The central computer will be located at a Ranger station and will be able to be used only by Rangers with verified information. Here at the central computer, Rangers will:

- Be able to login to access the system and information stored on the system.
- Be alerted when a new alert is received by the alarm ringing.
- Be able to generate reports of alerts based on four categories (Date, Sensor Location, Map View, and Ranger Classification) as well as edit or delete reports.
- View, edit, or delete individual alerts in the system and save them to the system.

Development Plan and Timeline

Partitioning of Tasks

- Build upon Animals-R-Here company system to fit the needs of the parks.
- Frequently discuss with the client/stakeholder the progress as well as any changes needing to be made to the system at any point of the timeline.
- Acquire noise sensors for the system.
- Modify system/sensor interactivity to allow for transfer of information from sensors to system and computer.
- Set connectivity points at the location for sensors to interact with the system and computer.
- Setup/place sensors in desired locations for required range and coverage of area.
- Setup the main terminal computer at Ranger station to be able to access the system.
- Build a UI for park rangers to interact with at the main central computer.
- Set up the alarm system to sound when the system receives a new alert.
- Connect and evaluate interactivity and connectivity of all components: sensors, system, computer, and alarm.
- Test noise sensor detection, reliability, and accuracy as well as capabilities of information transfer to the system. Address any bugs or problems found.
- Test system capabilities for receiving information from sensors and accuracy of information received. Address any bugs or problems found.
- Test accessibility and interactivity of the system from the computer. Address any bugs or problems found.
- Test alarm system for receiving new alerts and disabling the alarm. Address any bugs or problems found.
- Setup first full system test with all components working in conjunction.
- View results of initial test, discuss with team and client/stakeholder. Address any bugs or problems found.
- Tuning and refinement of the system until satisfactory.

Team Members Responsibilities

- Setup/hardware team will be in charge of acquiring the sensors and placing them with an area of 5 miles. They will also be in charge of ensuring the hardware is working properly.
- The UI team will be in charge of developing the interface park rangers will interact and receive reports from.
- The backend team will be responsible for storing the reports in the allotted time frame and provide summaries for reports older than a month but less than a year.
- The software Testing team will manually and develop tests to ensure bugs are fixed and software are working properly before deployment.

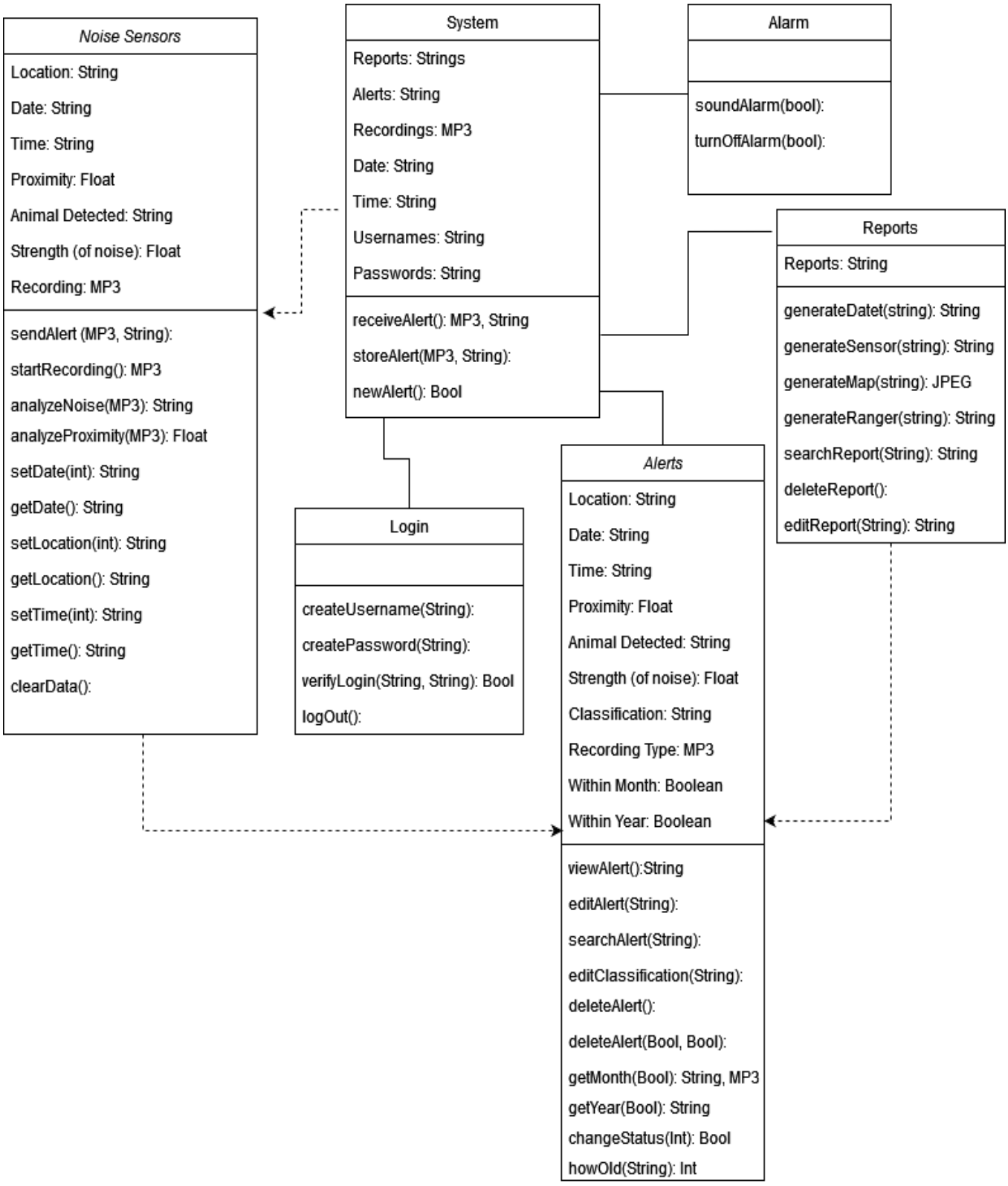
Timeline

We expect to have the program done within a month and the setup of equipment and adjustments made for the system to be another month. One month will be used to connect and test the system and all the components together. We plan to have a working prototype after three months.

After running some full system tests with all components, we plan to dedicate two (or more) additional months for testing, bug fixing, refinement, and changes needing to be made to the system as requested by the client/stakeholder.

If all goes according to the schedule, we plan to have the system up and running within half a year's time. This includes various testing, proper equipment setup and connection, program built as specified by the client/stakeholder, and bug fixing as well as any additional problems whether it be software or hardware that may need to be addressed.

Software Architecture Overview



Description of Classes

- Alerts class: Stores information regarding what was detected by the noise sensor. This includes the location, date, time, proximity from the noise sensor, strength of the noise, and the detected animal.
- Reports class: Creates a summary of all alerts over the past year that are older than one month.
- Noise Sensors class: The noise sensor triggers when the sensor detects a noise causing it to start recording audio as well as get the current information (date, time, location, etc) and send it to the system.
- System class: Receives alerts, recordings, reports, and login information from the rest of the system. Tells the alarm system when to activate.
- Alarm System class: Sounds an alarm whenever an alert is received by the main system. The alarm must be turned off manually by the park ranger.
- Login class: Handles the creation of accounts using a username and password, verifies any login attempts, and logs the user out of the system per request. All accounts are stored in the main system.

Description of Attributes and Operations

- Noise Sensors class
    - Attributes
        - Location: This contains the gps location of a sensor. Datatype: String
        - Date: This displays the date of the alert. Datatype: String
        - Time: This displays the time of the alert. Datatype: String
        - Proximity: This displays a numerical value of how close the noise was detected. Datatype: Float
        - Animal Detected: Displays what animal sound was analyzed. Datatype: String
        - Strength: Displays a numerical value of how loud the sound was (on a scale from 0-10, 0 being weak, 10 being strong). Datatype: Float
        - Recording: The recording of the noise in MP3 format.
    - Operations
        - sendAlert: Takes an MP3 and string file from the internal storage of the noise sensor and sends it to the main system.
        - startRecording: When the noise sensor picks up something, it will instantly start recording for 30 seconds.
        - analyzeNoise: It will take an MP3 and analyze the audio to be able to output a string with the name of the animal detected.
        - analyzeProximity: It analyzes how loud the noise was and outputs a float that represents the distance away from the sensor.
        - setDate: Takes an int from the internal computer to set the date as a string.
        - getDate: When a recording is triggered, it grabs the date and attaches it to the recording information.
        - setLocation: Takes an int from the GPS and uses it to set the location of the sensor as a string.

        - getLocation: When a recording is triggered, it grabs the location and attaches it to the recording information.
        - setTime: Takes an int from the internal computer to set the time as a string, updates constantly.
        - getTime: When a recording is triggered, it grabs the time (at that instant) and attaches it to the recording information.
        - clearData: Once the sendAlert function is done sending the information, it then clears the internal storage to make space for more recordings.

- System class
    - Attributes
        - Reports: Multiple alert information combined into a single document, can generate reports by certain attributes. Datatype: String
        - Alerts: Document containing information regarding a single instance of a noise detection. Depending on age of alert, it can have an associated MP3 recording.
        - Recording: The recording of the noise in MP3 format.
        - Date: This displays the date of the alert. Datatype: String
        - Time: This displays the time of the alert. Datatype: String
        - Usernames: Verified users that are capable of accessing the system. Datatype: String
        - Passwords: Verified users create their own passwords to be able to access the system. Datatype: String
    - Operations
        - receiveAlert: Receives the information from the noise sensor (MP3 file and string).
        - storeAlert: Stores the newly received alert.
        - newAlert: When an alert is received, it will set a boolean to true and then send it to the alarm.

- Login class
    - Operations
        - createUsername: Rangers can create a string username of their choice.
        - createPassword: Rangers can create their own string password of their choice.
        - verifyLogin: Checks the credentials stored in the system that have been verified as Rangers, any other usernames or passwords will not grant access to the system. This also helps identify rangers who are using the system to keep track of who makes changes, etc.
        - logOut: This signs the ranger out so other rangers can login or to keep unwanted users out from accessing the system.

- Alarm class
    - Operations
        - soundAlarm: It will take a boolean from the newAlert function and if true, it will sound the alarm until it is turned off.
        - turnOffAlarm: Once the ranger manually turns it off, it will not sound the alarm until a new alert is received.

- Reports class

    - Attributes
        - Reports: Multiple alert information combined into a single document, can generate reports by certain attributes. Datatype: String
    - Operations
        - generateDate: This generates and names a report based on the dates imputed and allows for a name to be assigned. This will take all the alerts for that date and create a report.
        - generateSensor: This generates and names a report based on the sensor imputed. This will take all the alerts for that sensor and create a report.
        - generateMap: This will take all the reports in a specific week and generate and name a map using a template map of the park and plot points on the map that represent approximate locations of noise heard.
        - generateRanger: This will generate and name a report based on all the classification assignments a specific ranger has made.
        - searchReport: Rangers can search a report name or keyword contained in the name and see if any results pop up for that search.
        - deleteReport: Rangers have the option to delete reports if needed, the system can delete multiple reports at once.
        - editReport: Rangers have the option to edit the report and add or remove information and save the changes made.

- Alerts class
    - Attributes
        - Location: This contains the gps location of a sensor. Datatype: String
        - Date: This displays the date of the alert. Datatype: String
        - Time: This displays the time of the alert. Datatype: String
        - Proximity: This displays a numerical value of how close the noise was detected. Datatype: Float
        - Animal Detected: Displays what animal sound was analyzed. Datatype: String
        - Strength: Displays a numerical value of how loud the sound was (on a scale from 0-10, 0 being weak, 10 being strong). Datatype: Float
        - Recording: The recording of the noise in MP3 format.
        - Classification: This contains the type of classification that the ranger assigns to it: definite, suspected, false. Datatype: String
        - Within Month: This lets the system know that the alert is within a month of being received. Datatype: Boolean
        - Within Year: This lets the system know that the alert is older than a month but still within a year of age. Datatype: Boolean
    - Operations
        - viewAlert: Produces a list of alerts from most recent to oldest. Then the rangers can click on a specific alert to view the information.
        - editAlert: Lets rangers edit individual alerts.
        - searchAlert: Lets rangers search for specific alerts.
        - editClassification: Lets a ranger change the classification of a certain alert from one type to another (definite, suspected, false).
        - deleteAlert: Lets rangers manually delete an alert if needed.

- deleteAlert(bool, bool): This lets the system automatically delete alerts that are older than a year.
- getMonth: This will list the reports that fall within a month and are full reports that include the MP3 recording and information.
- getYear: This will list the reports that are older than a month but fall within a year, these will only have a summary of the alert, not a full report nor MP3 recording.
- changeStatus: This will update the alert boolean by taking an int. If the int is 30 or less, it will assign within month = true, within year = false. If it is past 30 days but less than 365 days it will assign within month = false, and within year = true. If it is older than 365 days, it will assign both within month and within a year as false.
- howOld: This will compare the date attached to the alert and today's date and calculate how old the alert is. This will then spit out an int which will be sent to the change status to set the booleans for the alert.

Test Planning

Unit Testing

1. Noise Sensor

    a. Testing setDate()
        i. Input: 10282021 / Output: “October 28, 2021”
        ii. Input: 12252021 / Output: “December 25, 2021”
        iii. Input: 05269999 / Output: “Error: future date”
        iv. Input: 01011900 / Output: “Error: date out of range”
        v. Input: abx?173& / Output: “Invalid date”
        vi. Input: 102821 / Output: “Invalid date”
        vii. Input: 13372021 / Output: “Invalid date”
        viii. Input: / Output: “No date provided”

Here we are testing the setDate function which takes an int and outputs it as a string. We test many possible combinations of input that can result in errors or acceptable outputs. Some examples are dates that are too far in the future or past, inputs that are not ints, inputs that are not complete, no input, and input that has no month or date attached (for example there is no 13th month or 37th day possible).

    b. Testing analyzeNoise()
        i. Input: Sensor1-10282021-1705.mp3 / Output: “Detected animal: Coyote”
        ii. Input: Sensor4-09052021-2330.mp3 / Output: “No animal recognized”
        iii. Input: / Output: “Error: no mp3 input”
        iv. Input: Sensor1-10282021-0503.mp3 / Output: “Detected animal: Mountain Lion”

For the analyzeNoise test, it takes an mp3 file and analyzes the sound and recording, then outputs whether or not an animal was detected. If there is no recognizable animal sound, it will output that no animal is recognized. If it is able to recognize a sound, it will output the detected animal. And finally if there is no mp3 input when the function is called, it will output an error message.

2. Alerts

    a. Testing deleteAlert(bool, bool)
        i. Input: false, false / Outcome: Deletes alert because it is not within month or year
        ii. Input: true, false / Outcome: Does not delete alert, within month but not year
        iii. Input: false, true / Outcome: Does not delete alert, within year but not month
        iv. Input: true, true / Outcome: “Error: Invalid Operation”

The deleteAlert is called whenever an alert gets its bool withinMonth and withinYear changed. When only withinMonth is true, it will keep all the information. When only withinYear is true, it will keep an archived version of the report and delete the mp3 recording. When both withinMonth and withinYear is false, that means it is older than a year so the entire report will be deleted off the system. The system will run into an issue when withinMonth and withinYear are true because it is not possible to be less than a month old and older than a month, so then that will produce the error message. The withinYear is meant to signify older than a month but less than a year so when both are true, it’s a conflicting statement.

b. Testing viewAlert()
    i. Input:
    Output: “ Location: Sensor 4
    Date: 10/29/2021
    Time: 10:32 PM
    Proximity: 2.4
    Animal Detected: Mountain Lion
    Strength of noise: 9.5
    Classification: Definite
    Click to listen to MP3: Sensor4-10292021-2032.MP3
    Within Month: True
    Within Year: False“
    ii. Input:
    Output: “ Location: Sensor 1
    Date: 12/16/2020
    Time: 8:09 AM
    Proximity: 9.7
    Animal Detected: Coyote
    Strength of noise: 1.2
    Classification: Suspected
    Within Month: False
    Within Year: True
    *** *You are viewing an archived report* *** “
    iii. Input:
    Output: “ “

The viewAlert function is gonna work by clicking on an alert in the alerts menu and then it will produce a report. There are three types of test, one test is to view an alert that is less than a month old. This produces the full alert with the option of listening to the mp3 that was captured with the alert. The next test is viewing an alert that is older than a month, which only shows an archived report with less information than a full report. And the final test is viewing an empty report that could have been generated by mistake or edited by a ranger and deleted all information from the alert.

Integration Testing

1. Login + System

    a. Testing verifyLogin(String, String)
        i. Input: “SMartin”, “Xyz123__” / Output: true
        ii. Input: “SMartin123_”, “yz1X23_!” / Output: true
        iii. Input: “randomuser123”, “password” / Output: false
        iv. Input: “sudnhfkdehy”. “120983474” / Output: false
        v. Input: “129804738”, “3485794” / Output: false

The verifyLogin function relies on both the Login class and the System class. It accepts two Strings: The username and password. The System class has all of the valid username/password combinations, and the verifyLogin function compares the input with the stored usernames/passwords. Valid usernames depend on the ranger trying to login. Usernames contain the first letter of the ranger’s first name, followed by the

ranger's last name, and optionally, some numbers or special characters at the end. Valid passwords are at least 8 characters long, with at least one lowercase letter, one uppercase letter, one number, and one special character.

2. Reports + System

    a. Testing searchReport(String)
        i. Input: "October 28, 2021" / Output: Report from October 28, 2021
        ii. Input: "September 26, 2021" / Output: Report from September 26, 2021
        iii. Input: "February 30, 2021" / Output: "No Results"
        iv. Input: "October 26, 2022" / Output: "No Results"
        v. Input: "October 33, 2021" / Output: "No Results"

The searchReport function takes a String as input, and uses that String to filter through the names of the reports that are stored in the System class. This function outputs a String of reports that contain the given input string.

System Testing

1. All Classes

    a. verifyLogin(String, String):
        i. Input: "SMartin", "Xyz123_!" / Output: true
    b. analyzeNoise(MP3):
        i. Input: "Sensor2-10292021-1603.mp3" / Output: "Mountain Lion"
    c. analyzeProximity(MP3):
        i. Input: "Sensor2-10292021-1603.mp3" / Output: 5.03
    d. sendAlert(MP3, String)
        i. Input: "Sensor2-10292021-1603.mp3",

            "Location: Sensor 2
            Date: 10/29/2021
            Time: 4:03 PM
            Proximity: 5.03
            Animal Detected: Mountain Lion
            Strength of noise: 9.2
            Classification: Definite
            Click to listen to MP3: Sensor2-10292021-1603.MP3
            Within Month: True
            Within Year: False"

        ii. Output: sends the Alert to the System class
    e. receiveAlert()
        i. Input:
        ii. Output: "Sensor4-10292021-2032.MP3",

            "Location: Sensor 2
            Date: 10/29/2021

Time: 4:03 PM
Proximity: 5.03
Animal Detected: Mountain Lion
Strength of noise: 9.2
Classification: Definite
Click to listen to MP3: Sensor2-10292021-1603.MP3
Within Month: True
Within Year: False”

iii.

f. storeAlert(MP3, String)
    i. Input: “Sensor2-10292021-1603.mp3”,

“Location: Sensor 2
Date: 10/29/2021
Time: 4:03 PM
Proximity: 5.03
Animal Detected: Mountain Lion
Strength of noise: 9.2
Classification: Definite
Click to listen to MP3: Sensor2-10292021-1603.MP3
Within Month: True
Within Year: False”

    ii. Output: Stores the Alert in the System class
g. soundAlarm(bool):
    i. Input: true / Output: Sounds the alarm
h. turnOffAlarm(bool):
    i. Input: true / Output: Turns off the alarm
i. viewAlert()
    i. Input:
    ii. Output: “Location: Sensor 2
Date: 10/29/2021
Time: 4:03 PM
Proximity: 5.03
Animal Detected: Mountain Lion
Strength of noise: 9.2
Classification: Definite
Click to listen to MP3: Sensor2-10292021-1603.MP3
Within Month: True
Within Year: False”
j. searchReport(String):
    i. Input: “10292021” / Output: New Report
k. logOut()
    i. Outcome: Successful logout

This system test goes through a typical flow, starting with the ranger logging in to the system. Then, sensor 2 picks up a sound and analyzes it using the analyzeNoise and analyzeProximity functions. The Noise Sensor class sends the newly generated alert to the System class, and once it’s been received, the alarm goes off. The alarm stays on until the ranger turns it off manually. After turning off the alarm, the ranger can then view the alert that has just been stored, and look through any past reports. Finally, the ranger logs out of the system, and the flow ends.