Mountain Lion Detection System Software Design
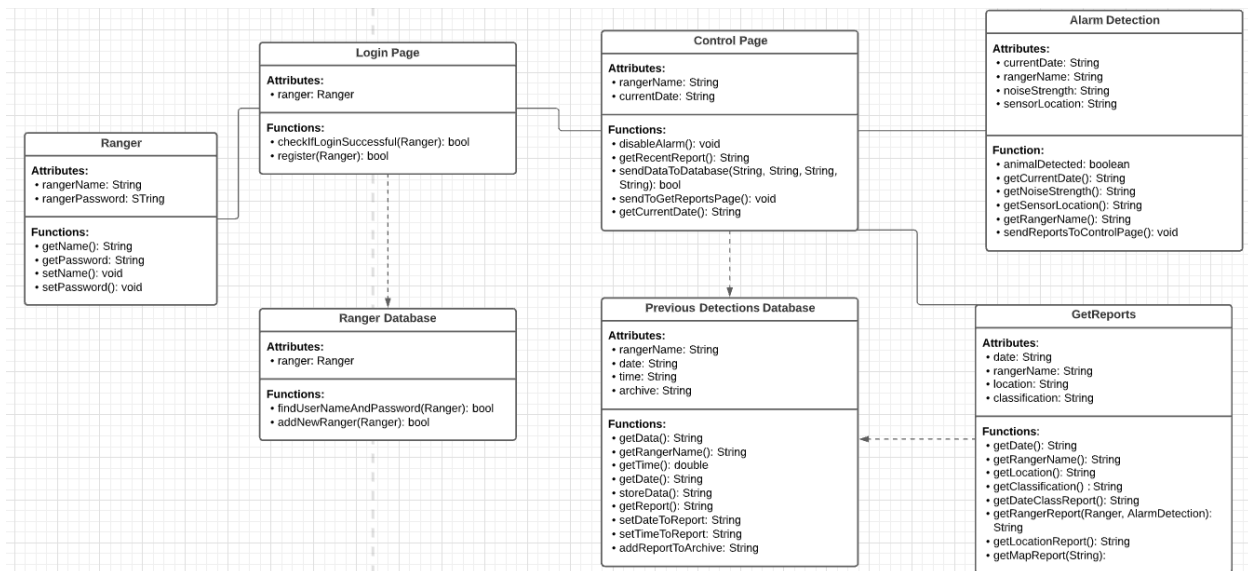
2.0

Authors:

- Anthony Mina

- Kaitlyn Nguyen

- Remy Huynh

- Halah Salman

1. Software Design Specification:

The System has Four pages and two databases. The login page is to see if the ranger using the system is existing or new and checks the ranger database to confirm that. The Alarm Detection page collects information from the Sensors if an Animal/Mountain Lion is detected. Information such as the sensor's location that detected the animal and the strength of the noise detected. The Alarm Detection class also keeps track of the date and time the animal was detected and the name of the Ranger responsible for the area of the sensor. All this information then gets sent as a report to the Control Page. The operator of the Control page can disable the alarm, send the data to the Database and look up previous reports from the report's page. The database retrieves and saves the reports for documentation and future reference. The Get Reports page is where all the reports would be stored and archived to help predict and manage future incidents.

Login Page

Attributes:
• ranger: Ranger

Functions:
• checkIfLoginSuccessful(Ranger): bool
• register(Ranger): bool

Ranger

Attributes:
• rangerName: String
• rangerPassword: STring

Functions:
• getName(): String
• getPassword: String
• setName(): void
• setPassword(): void

Control Page

Attributes:
• rangerName: String
• currentDate: String

Functions:
• disableAlarm(): void
• getRecentReport(): String
• sendDataToDatabase(String, String, String, String): bool
• sendToGetReportsPage(): void
• getCurrentDate(): String

Alarm Detection

Attributes:
• currentDate: String
• rangerName: String
• noiseStrength: String
• sensorLocation: String

Function:
• animalDetected: boolean
• getCurrentDate(): String
• getNoiseStrength(): String
• getSensorLocation(): String
• getRangerName(): String
• sendReportsToControlPage(): void

Ranger Database

Attributes:
• ranger: Ranger

Functions:
• findUserNameAndPassword(Ranger): bool
• addNewRanger(Ranger): bool

Previous Detections Database

Attributes:
• rangerName: String
• date: String
• time: String
• archive: String

Functions:
• getData(): String
• getRangerName(): String
• getTime(): double
• getDate(): String
• storeData(): String
• getReport(): String
• setDateToReport: String
• setTimeToReport: String
• addReportToArchive: String

GetReports

Attributes:
• date: String
• rangerName: String
• location: String
• classification: String

Functions:
• getDate(): String
• getRangerName(): String
• getLocation(): String
• getClassification() : String
• getDateClassReport(): String
• getRangerReport(Ranger, AlarmDetection): String
• getLocationReport(): String
• getMapReport(String):

Login Page:

- **Attributes:**

  - **currRanger** = ranger object containing information of current user

- **Functions:**

  - **checkIfLoginSuccessful(ranger)** = sends information to database and returns bool whether it was successful or not.

  - **register(ranger)** = registers new ranger and takes in name and password as arguments and returns bool whether it was successful or not.

Ranger Class:

- **Attributes:**

  - **rangerName** = holds the currents user's name

  - **rangerPassword** = holds the current user's password

- **Functions:**

  - **getName()** = returns name

  - **getPassword()** = returns the password

  - **setName()** = set the name to a new one if needed

  - **setPassword()** = set the password to a new one if needed

Ranger Database:

- **Attributes:**

  - **rangerName** = holds the current user's name

  - **rangerPassword** = holds current user's password

- **Functions:**

- **findUsernameAndPassword(string, string)** = looks through the database for matching pair to get logged in. Returns bool whether successful or not.

- **addNewRanger(string, string)** = stores new rangers name and password into database and returns whether it was successfully done.

Control Page:

- **Attributes:**

  - **rangerName** = this is the name of the ranger who is currently using system

  - **currentDate** = holds the current date

- **Functions:**

  - **disableAlarm** = this function allows the ranger to turn off the alarm after a detection was made

  - **getRecentReport()** = gets the most recent report that was sent from AlarmDetection and prints it to screen

  - **sendDataToDatabase()**: after information has been sent from alarm detection the control page allows you to send the information to the database. It takes in 4 String arguments that are for: date, location, strength, and name of the ranger reporting. It returns a bool depending on if information was stored successfully

  - **getCurrentDate()** = gets the current date and sets our currentDate attribute to it

  - **sendToGetReportsPage()** = this function sends you to the getReportsPage where you will be able to access previous reports and pull up information about them.

Alarm Detection:

- **Attributes**:

  - **CurrentDate()** = Keeps track of the current date the animal was detected and we

can format it to include month, year, day & time of the incident.

- **NoiseStrength()** = This gets from the sensor the strength of the noise of the detected animal.

- **SensorLocation()** = Gets the number of the sensor to report on the location of the incident.

- **RangerName()** = Gets the assigned Ranger's name which would be based on the location of the sensor or the shifts.

● **Functions:**

- **animalDetected** = This function monitors the sensors to ensure that the alarm gets activated when an animal is detected.

- **getCurrentDate()** = This function keeps track of the date and time.

- **getSensorLocation(String)** = This function distinguishes between the sensors to report where the animal was detected and what ranger reported it.

- **getNoiseStrength()** = This function gets the strength of the Noise detected to keep track of how close the animal is to the sensor.

- **getRangerName()** = This function links the sensor's location to the assigned ranger for that location.

- **sendReportsToControlPage()** = This function sends a report of all the findings from above to the Control Page for the Operator to make the next move.

Detections Database

● **Attributes:**

- **rangerName** = this supplies the name of the ranger using the detection system

- **Date** = the date an alert was sent to the database

- **Time** = the time an alert was sent to the database

- **Archive** = stores all past reports in one area for easy access

● **Database Functions:**

- **getRangerName()** = retrieves the name of the latest ranger using the detection

  system

- **getData()** = retrieves the data (detection details) from the alert system and control

  center

- **getDate()** = retrieves the date of occurrence for report

- **getTime()** = retrieves the time of occurrence for report

- **storeData()** = stores data from all reports into the archive for future access

- **getReport()** = retrieves all reports from the control page to store for future

  reference

- **setDateToReport()** = saves the date from each report so that it can be accessed

  chronologically

- **setTimeToReport()** = saves the time from each report so that it can be organized

  by details

- **addReportToArchive()** = saves the recent report into the archive

Get Reports:

● **Attributes**:

- **Date** = date of detection

- **rangerName** = name of responding ranger

- **Location** = location of sensor that detection activity

- **Classification** = classification (definite, suspected, or false) of alert determined by rangers

● **Functions**:

- **getDate()** = gets detection date of alert in order of most recent to oldest

- **getRangerName()** = gets the name of the ranger that classified the alert by alphabetical order

- **getLocation()** = gets the sensor location of detection

- **getClassification()** = gets the classification of alert in order of definite, suspected, and false

- **getDateClassReport()** = gets report based on date and classification of alert and print to screen

- **getRangerReportBySensor(ranger, alarmDetection)** = gets report of location based on which ranger reported it.

- **getLocationReport()** = gets report by sensor location and print to screen

- **getMapReport(String)** = gets geographical report of detection locations within park and display

2. Data Management Strategy:

We will be using SQL style databases to represent data in our system. We will have two databases each with one entity. One of the databases will represent Ranger login information and the other will store reports of detections that have occurred.

1. Ranger DB

Username(unique key used for relations to other tables)	Password

2. Detections DB

Username	Time Detected	Date Detected	Location	Classification (strength)

1. **Data Dictionary Ranger DB:**
   - **Name: Ranger**
   - **Description: Holds ranger user's information to log in and later be used to find previous detections**
   - **Type: Table**
   - **Primary Index: Username**

   - **Name: Username**
   - **Description: Holds a ranger's username**
   - **Type: String**
   - **Length: 8-20**

   - **Name: Password**
   - **Description: Holds a ranger's password**
   - **Type: String**
   - **Length: 8-20**
   - **Constraints: Minimum (1 capital, 1 digit, 1 special character)**

2. **Data Dictionary Previous Detections DB**
   - **Name: Detections**
   - **Description: Stores all past reports in one area for easy access.**
   - **Type: Table**
   - **Primary Index: username**

   - **Name: Username**

- **Description: Holds the ranger's username who reported on the detection**
- **Type: String**
- **Length: 1-20**

- **Name: Time Detected**
- **Description: Stores the time of the detection**
- **Type: String**

- **Name: Date Detected**
- **Description: The date the alert was sent to the database**
- **Type: String**
- **Length: 8**

- **Name: Location**
- **Description: Holds location of sensor that detected the noise**
- **Type: String**
- **Length: 1-20**

- **Name: Classification**
- **Description: Strength of the noise detected**
- **Type: String**