Mountain Lion Detection System Software Design Specification

Authors:

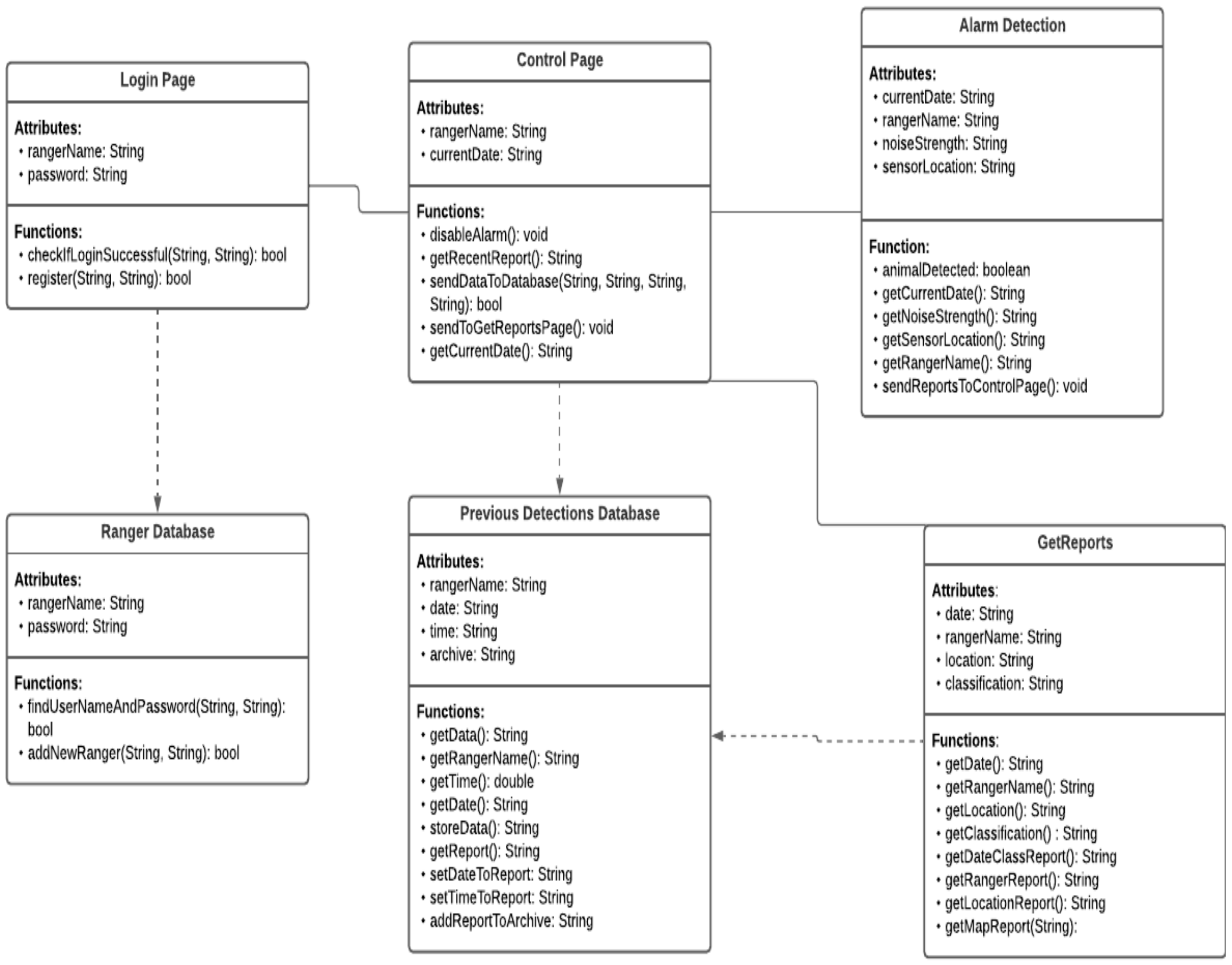
- Anthony Mina
- Kaitlyn Nguyen
- Remy Huynh
- Halah Salman

System Description:

The Mountain Lion Detection System uses noise detection sensors throughout the San Diego County Parks and Recreation Department placed within a 5 square miles area. The device detects various animal noises and sends alert messages to the control computer located in the park ranger station. Alert messages contain the strength of the noise, the type of animal detected and location to within 3 miles. The alert message triggers an alarm that will sound in the park ranger station until a park ranger responds to the alert and the alarm will not sound again until another, separate noise is detected in another location. Rangers will be able to classify an alert as definite, suspected, or false, indicating the probability that a real mountain lion was detected. The control computer will save all mountain lion alerts received within 30 days and a summary of alert messages for data older than 30 days but received within the year. Rangers will be able to request reports showing all mountain lion detections by date detected and classification or by specific location of sensor; geographical report showing detection on a map of the park and 2 miles outside of the park; classification report by ranger.

Software Architecture Overview:

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:
• rangerName: String
• password: String
Functions:
• checkIfLoginSuccessful(String, String): bool
• register(String, String): bool
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:
• rangerName: String
• password: String
Functions:
• findUserNameAndPassword(String, String): bool
• addNewRanger(String, String): 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(): String
• getLocationReport(): String
• getMapReport(String):

Login Page:

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

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

- **Functions:**
  - **checkIfLoginSuccessful()** = sends information to database and returns bool whether it was successful or not.
  - **register(string, string)** = registers new ranger and takes in name and password as arguments and returns bool whether it was successful or not.

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()** = This function distinguishes between the sensors to report where the animal was detected.
- **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

- **getRangerReport()** = gets classification report by ranger and print to screen
- **getLocationReport()** = gets report by sensor location and print to screen
- **getMapReport(String)** = gets geographical report of detection locations within park and display

Development plan and timeline:

The system we have been given is a mountain lion detection system that will detect and send alarms to a control computer in the ranger's office when a mountain lion is near a sensor. It will also be able to store the information into a database for ease of access. It was Anthony's job to get the control page, login page, and ranger database to get up and running. He was tasked with making sure the classes and databases communicate well with each other. Anthony's responsibilities should take about 2-3 weeks to complete. Halah was responsible for making the Alarm detection page and she worked with the detection monitors out in the park. She placed them in areas where mountain lions are often seen and made sure they communicated with her code. Halah's alarm detection class will take an estimated 2 weeks. Kaitlyn's task was to make sure the database was pulling up the correct information when you wanted to pull up reports. She made a page specifically for accessing previous reports and made connections to the database with that page. Kaitlyn's task will be finished within about 2 weeks. Lastly, Remy constructed the database that would hold previous reports information. He had to make sure that things were being stored correctly and was being pulled up and displayed correctly. Remy's database has an estimated completion time of 3 weeks. Overall, our client wants the system to be up and running within three months. For every week we have made milestones and deliverables to make sure we meet that three month deadline.