Data Management Plan:
San Diego Mountain Lion Detection System

Version 3.0

Members: James Marsh, Yan Ho, Jacob Miranda Miranda

GROUP 06

CS 250

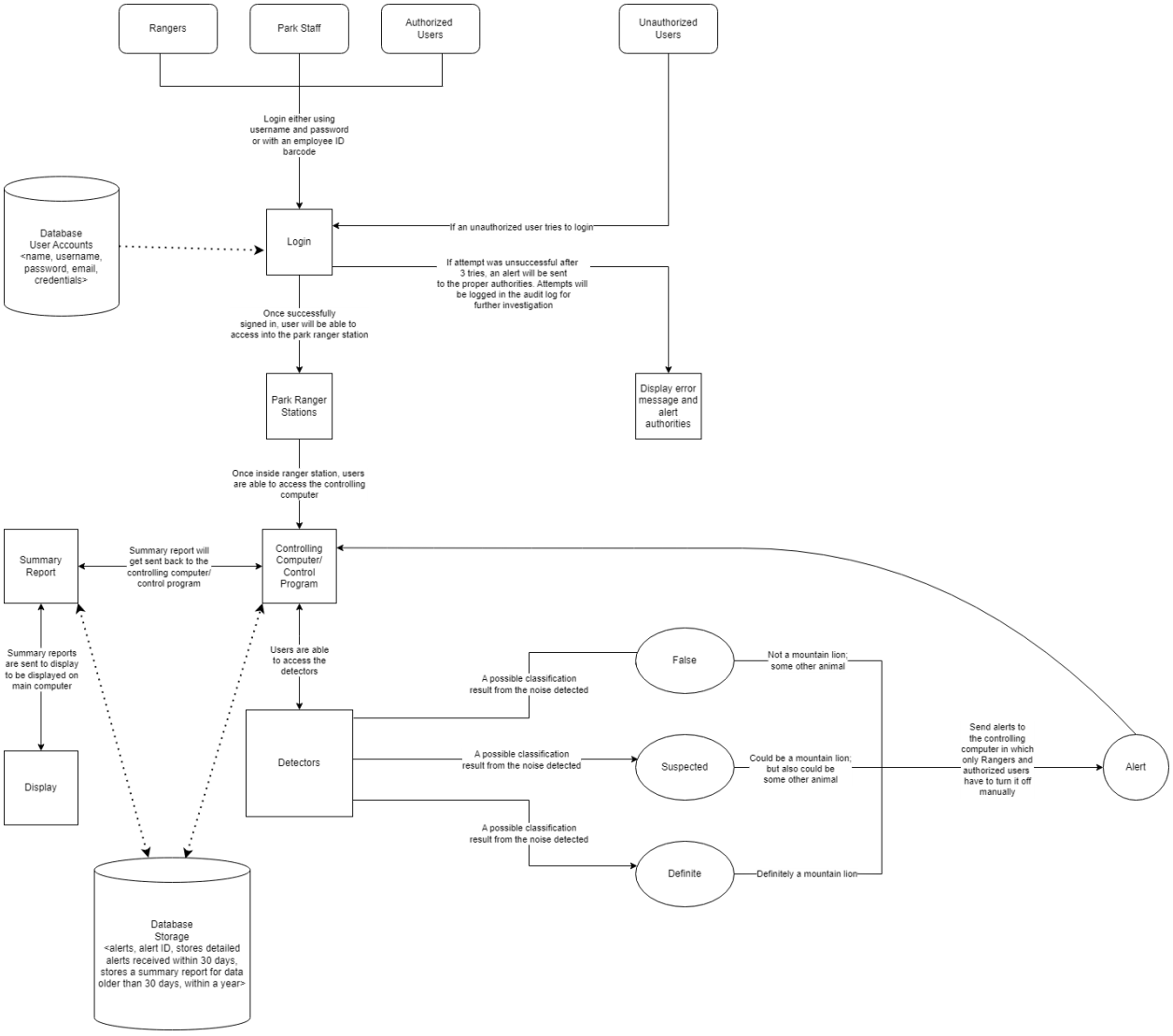
Professor Miranda Parker

November 11, 2022

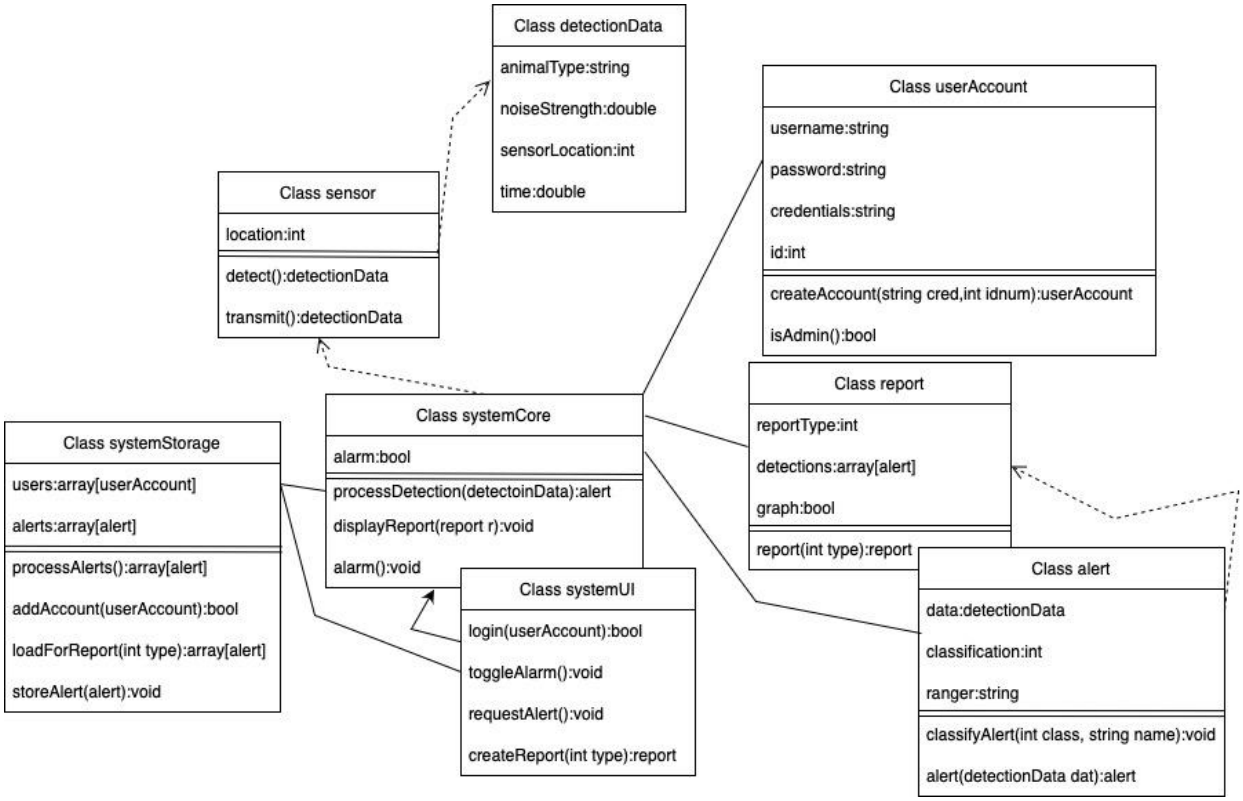
System Description

The system we are designing is the mountain lion detection system for the San Diego Parks and Recreation Department. The system will be able to detect noises of various types of animals within a five square mile of the sensor. The sensor will also be able to determine the strength of the noise and the location of the noise. There will be a controlling computer that supports system access, configurations, and storage. The controlling computer will be able to save detailed summaries within the last 30 days and a summary of the data received within the year. The controlling computer will also have the ability to continuously sound an alarm whenever it receives an alert message from the detection system until a park ranger turns it off. There will be a control program on the controlling computer that allows alerts to classify the probability of detecting a real mountain lion. The detection system control program will support a display of alert data in the form of reports. Reports consist of alert descriptions that vary depending on the type of report requested.

Architectural Diagram



UML Class Diagram



Data Management Plan

We will need to implement two separate SQL databases for the system to store necessary data. One database will contain sensor and alert-related information, and the other will contain user login information for system administrators. The data split between the two databases is based on the data type, either alert-related or user-related. The data stored by the system is relatively static, but the classification of alerts and user login info is subject to change situationally. However, the data is generally the same as it was initially stored. We expect to house the databases in a secure facility with a central control computer within the park.

Additionally, we suggest that the databases can support remote access to improve system efficiency and facilitate backups, but this was not part of the system requirements. Storing the data offsite could also provide benefits for the system, including remote access and data loss prevention. However, since our system is not inherently connected to an external network, this would require system modification. SQL databases provide efficient storage for relatively static and related data, so we choose SQL for this system. NoSQL databases could increase system capability for handling large influxes of data (for example, if the system received multiple detections simultaneously). However, an SQL database would be the best fit for this system.

Databases

UserAccount DB is an SQL database that stores user login info. It stores username and password sets with corresponding employee info.

UserAccount DB			
username	password	ID	credentials
jeffEasy123	1L1k3C4t5	837429	ADMIN
BryanPeterson	$$%*_H1ka	810923	ADMIN

Alert DB is an SQL database that stores animal detections. It parses out the information from the detectionData object found within an alert and gives each set of data a unique id.

Alert DB						
alertID	classification	ranger	type	strength	location	time
1	1	Jeff	lion	37.2	1	01142022.1431
12	3	Bryan	bird	55.0	3	11112022.1643
30	3	Bryan	lion	53.1	3	12092021.1830

Data Dictionary:

UserAccount Database:

- username is a string that holds the user's login name
- password is a string that holds the corresponding password for a particular username
- ID is an int that holds the employees ID number
- credentials is a string that holds the employees clearance, which determines whether or not they have access to the system

Alert Database:

- alertID is an int that provides a unique id for each alert stored in Alert DB.
- classification is an integer field with an inclusive range from 1 to 3 that ranks the accuracy of each detection
- ranger is a string that holds the name of the ranger that classified an alert
- type is a string that holds the type of animal that is detected
- strength is a double holding the decibel level of the noise detected
- location is an int that corresponds to a specific sensor and where it is within the park
- time is a double representing the date and time of a detection with date in the whole part

and time by a 24hr clock in the decimal part formatted as: MMDDYYYY.TIME

Description of Classes

The system relies on the detectionData class which is a cluster of information received by the sensor. The sensor class implements the detection function and saves data in the corresponding fields of the detectionData class. Each detection alert will have its own unique detectionData. The sensor class also transmits detectionData to the systemCore class which processes data into an alert and stores it. Alerts are just a display of the detection data with an additional field for rangers to classify the certainty of a detection. The systemCore also sounds an alarm according to the previously received alert and implements a user interface where users can login, toggle alarms, configure alerts, and request reports. The report class supports four different types of reports that can be specified by an integer ranging [1,4]. Depending on the type of report that is specified, the corresponding data is fetched from storage via the createReport function. The storage class contains all stored alert data and user login information. Additionally, the storage class routinely processes stored data and removes outdated alerts.

Description of Attributes List

Class alert -

- data detectionData - Found in the alert class, data holds an instance of a detectionData class. data is an important attribute of the alert class because it makes up the majority of the information that alert encapsulates.
- int classification - Found in alert class, classification is an int representing the confidence that the detection was definite. One, two and three corresponds to false detection, suspected, and definite, respectively.
- string ranger - Found in alert class, ranger is a string used to hold the name of the park ranger who ranked the classification of the alert.

Class detectionData -

- string animalType - Found in detectionData class. It is a string used to hold the animal's species that created the sound detected by the system.

- ➢ double noiseStrength - Found in detectionData class. noiseStrength is a double used to hold the recorded sounds detected in decibels.
- ➢ int sensorLocation - Found in detectionData class. sensorLocation is an int used to store the number of the sensor that recorded the sound.
- ➢ double time - Found in detectionData class. time is a double use to store the current time and current date of when the sound was detected.

Class report -

- ➢ int reportType - Found in report class. reportType is an int that allows the user to select what type of report they want to view. The different types are mountain lions only, reported by the ranger who classified it, reports by a particular sensor, and a map showing all detections in the park.
- ➢ alert detections[ ] - Found in report class. detections is an array pointing to alert objects intended to keep a history of alerts. The data in detections will be used to generate reports since alert instances hold detectionData objects as well as classification data.
- ➢ bool graph - Found in the report class, graph is bool meant to be toggled by the user in case the user wants a graphical report.

Class sensor -

- ➢ int location - Found in sensor class. location is an int used to store the sensor's number. Each sensor will have a designated number to serve as its identifier and location moniker.

Class systemUI -

- ➢ **Class contains no attributes**

Class systemCore -

- ➢ bool alarm - Found in systemCore class. alarm is a bool used to determine if an alarm is on or off. True means on and false means off.

Class systemStorage -

- ➢ userAccount users[ ] - Found in systemStorage, users is an array pointing to instances of userAccount objects. users will be used to authenticate authorized personnel by making sure the credentials entered in systemUI.login() match one of its userAccount elements.
- ➢ alert alerts[ ] - Found in systemStorage, alerts is an array pointing to alert objects.

Class userAccount -

- ➢ string username - Found in userAccount class. username is a string used to store the user's login name.
- ➢ string password - Found in userAccount class. password is a string used to store the user account's password.
- ➢ string credentials - Found in userAccount class. credentials is a string used to identify the user.
- ➢ int id - Found in userAccount class. id is an int that contains a series of numbers to identify the user.

Description of Operations List

Class alert -

- ➢ void classifyAlert(int class, string name) - Found in alert class, classifyAlerts() accepts an int argument and string argument used to update the classification level of an alert and the ranger who classified it. The ints one, two, and three reclassify an alert to false, suspected, and definite, respectively.
- ➢ alert alert(detectionData dat) - found in the alert class, alert() is a constructor used to create system alerts based on the passed detection data.

Class detectionData -

- ➢ **Class contains no operations**

Class report -

- report report(int type) - Found in the report class this is a constructor used to create reports with data based on the type of report specified.

Class sensor -

- detectionData detect( ) - Found in sensor class, detect( ) runs when a noise is detected by a sensor. detect() then generates a detectionData object returns it to be transmitted.
- detectionData transmit() - Found in sensor class, detect() returns detectionData retrieved from detect( ). It is needed to transmit the sensor's data. The operation works by receiving and returning detectionData sent to it by detect( ).

Class systemCore -

- alert processDetection(detectionData) - Found in systemCore, processDetection( ) returns an alert to be stored in SystemStorage. processDetection( ) is used to create an alert. The operation will work with soundStrength( ) to see if animal noises are detected. If it is, it will sound off an alert that will set off the alarm, notifying rangers that the system has detected an animal.
- void displayReport(report r) - Found in systemCore, this method is used with the createReport function to display reports accordingly on the control computer so that they can be viewed by users. This method is also used to process graph data as well, in the case that the report requires that.
- bool alarm( ) - Found in systemCore, alarm( ) returns a boolean value so the system knows if a sensor's alarm is on or off. toggleAlarm( ) is another method that is found in systemUI allowing users to disable the alarm.

Class systemStorage

- alert[ ] processAlerts( ) - Found in systemStorage, processAlerts( ) returns an array of alerts. processAlerts is used to periodically clean out the system storage and eliminate any outdated alerts automatically.
- bool addAccount(userAccount) - this is a method used by the systemStorage class to add new accounts to the system. If this method returns true, then the account has been successfully added.

- alert[ ] loadForReport(int type) - Found in systemStorage this is a helper method for report requests that compiles and returns an array of alerts from the system storage corresponding to the type of report request made. Accepts one int argument.
- void storeAlert(alert) - Found in systemStorage. This operation stores the created alert from systemCore (systemCore.processDetection( )) and updates systemStorage.alert[ ] with the new alert

Class systemUI

- bool login(userAccount) - Found in systemUI, login( ) will be called every time a user attempts to access San Diego Mountain Lion Detection System. login( ) is critical for the system's security as it allows only authorized personnel access. An user account object is passed to login and if systemStorage contains the same user account then true is returned and the user is granted access to the system, else false is returned and access is denied.
- void toggleAlarm( ) - Found in systemUI, toggleAlarm will be used to turn off any active alarms set off by processDetection( ). This operation is needed for sensors because they require manual input to stop throwing alarms. When toggleAlarm( ) is called a sensor will receive a signal to stop transmission of its alarm. If no alarm is being transmitted toggleAlarm( ) informs the user that the sensor is not sounding off.
- void requestAlert( ) - Found in systemUI, requestAlert( ) will be used to retrieve alert data from systemStorage. This operation allows users to retrieve and view various alerts held in the system. requestAlert( ) works by asking the user to identify a specific alert to retrieve. If no alert is specified the newest alert is retrieved. If requestAlert( ) fails to retrieve a specified alert then the user is informed that no such alert exists.
- report createReport(int type) - Found in systemUI, createReport( ) is used to generate and return a report object while also displaying its contents. createReport( ) is needed so rangers can make a summary of alerts and view them. This operation works by taking an int which corresponds to a classification of an alert and retrieving all queried items for the report object.

Class userAccount

- ➢ userAccount createAccount(string cred, int idNum) - createAccount( ) is a constructor found in userAccount used to create accounts with a system generated login for specific employees. This operation accepts a string and int argument and also returns an userAccount object.
- ➢ bool isAdmin() - Found in the userAccount class, this method returns true if the account is valid for system configuration and operation according to the employee's credentials.

Development Plan and Timeline

With all the coding and testing, we plan to have a finished product within 6 months. Perhaps in the future, we can expand to other locations for parks and recreations. Maybe we can even make detectors for other animals. We will continue to update the system to make sure the system is up to date and there are no bugs. We will ask our clients for their feedback and alter the system to their responses.

Yan will be responsible for:

- Class detectionData
- Class sensor

James will be responsible for:

- Class systemCore
- Class systemUI

Jacob will be responsible for:

- Class report
- Class alert

Verification Test Plan

isAdmin Login Testing

Unit Testing

This test objective for isAdminLogin is to check if the user is an authorized user or not. Authorized users are park ranger personnels. If they are authorized, they will be able to log into their accounts and use the control program to access information related to the system. isAdminLogin module controls are:

- Login
    - Username
    - ID
    - Password
- Logout

Functional/Integration Testing

The isAdminLogin will use the username and password inputted to determine if the login is valid or invalid. This function will communicate with the database to make sure the inputted data is valid in the database. Authorized users will be able to login to the website and access all the features an authorized personnel is able to view. If an unauthorized user attempts to login and fails, authorities will be alerted. isAdminLogin will test the modules:

- String: username
- String: password
- Int: id

System Testing

In order to have a successful login, the information inputted into username and password has to be correct. If login is valid, user will be able to access the control program and view:

- Settings
- Audit Log
- Alerts
- Logout

If login is invalid, authorities will be notified. An investigation will commence.

Login	isAdmin Login	My Account

String: username
String: password
Int:id
Bool: isAdmin
void setIsAdmin()
	Login Test
	bool (true, false): loginTest
	if (loginTest = False)
	return "Invalid Input";
	else if (loginTest = True)
	return "Login Success!";
	else
	return null;
	void invalidLogin()
	void validLogin()
		Settings
		Audit Log
		Alerts
		Logout

Unit testing: Login.setIsAdmin()	Functional/Integration testing: loginTest.invalidLogin(Login) loginTest.validLogin(Login)	System testing: Access to control program Read all alerts Logout

Test Case #	Test Case Description	Test	Expected Results	Actual Results	Pass/Fail
1	Park ranger goes and logs into the control program. Ranger types in their account information that consists of username/ID and password.	Username: SDMLRanger1 ID: 979831 password: iDontknowPlzhe1p	Login is successful. Ranger was able to login and access the control program.	Login successful.	Pass
2	Some unauthorized user got into the control room with the control program. Tries to login to the control program using what they think is the login/ID and password of one of the park rangers.	Username: SDParkRanger12 ID: 165168 password: Broireallydontknow	Login is unsuccessful. The login information was incorrect. After three unsuccessful tries, authorities will be alerted of this attempt.	Login unsuccessful.	Fail

classifyAlerts Testing

Unit Testing

This test objective is to ensure that the classification given is valid. The classification consists of numbers from 1 to 3. The detection system will send a number back to the control program to determine the classification of the animal noise whether the noise is:

- 1, a false alarm.
- 2, there might be a chance of a mountain lion.
- 3, there is definitely a mountain lion.

If anything else is given, there is an error in the detection system and needs to be checked.

Functional/Integration Testing

The classifyAlerts program will check the input given from the detection system. This function will communicate with the detectors to make sure the classification has the correct outputted ratings. The alert will then get sent back to the control program for rangers to store in the report. The ranger who's monitoring the alerts will include their name in the report since alerts have to be checked by a ranger. classifyAlerts will test the modules:

- Data from the detectionData
- Int: classification
- String: ranger

System Testing

In order to get a correct classification, park rangers will have to make sure the classification received from the detectors are only numbers between 1 to 3. If the classification received is valid, then the monitoring ranger's name and the classification number will be stored in the summary/report. The information will appear as:

- Classification 1, monitoring ranger's name
- Classification 2, monitoring ranger's name
- Classification 3, monitoring ranger's name

If the classification received is not a number between 1 to 3, it was appear as:

- Error, not a classification. Please check the detection system.

	classifyAlerts	Classification

Classification
Data: detectionData
Int: classification
String: ranger
void setIsClassification
	Classification Test
	if (classification == 1 || classification == 2 || classification == 3);
	return Alert<Int, String>(classification, rangerName);
	else
	return "Error";
	void readAlert ()
		1
		2
		3
		Error

Unit testing: Classification.setIsClassification	Functional/Integration testing: classifyAlerts.readAlert(Classification)	System testing: Classification 1, Ranger's Name Classification 2, Ranger's Name Classification 3, Ranger's Name Error - Not a Classification

Test Case #	Test Case Description	Test	Expected Results	Actual Results	Pass/Fail
1	Alert comes in with a classification of 1. Classification of 1 means false alarm.	The ranger will check the classification to make sure it is correct.	If classification is 1, the system will log the result along with the name of the ranger who checked the alert.	The classification is successful, the classification and ranger name will be saved in the update.	Pass
2	Alert comes in with a classification of 2. Classification of 2 means there may be a chance that a mountain lion is present.	The ranger will check the classification to make sure it is correct.	If classification is 2, the system will log the result along with the name of the ranger who checked the alert.	The classification is successful, the classification and ranger name will be saved in the update.	Pass
3	Alert comes in with a classification of 3. Classification of 3 means there is a mountain lion present.	The ranger will check the classification to make sure it is correct.	If classification is 3, the system will log the result along with the name of the ranger who checked the alert.	The classification is successful, the classification and ranger name will be saved in the update.	Pass
4	Alert that comes in with anything other than 1,2 or 3. That means there's an error with the detection.	The ranger will recheck the classification and check out the error.	If classification is anything that isn't 1, 2 or 3, the system will send out an error.	The classification is unsuccessful, the classification and ranger name will be no update. The data will need to be double checked.	Fail