1. Unit Testing

For our unit tests we will be testing the most basic and important functions of our overall system. We will be testing the functionality of our sensors. The two main focal points for our sensor will be to test the detection of audio as a whole and pinpointing the location of the detected noise.

I. Test audio detection by sensors

The test procedure will incorporate an example environment to simulate real world use. We will place the sensors in an open area, connected to power, and the processing unit. We will play various audio clips including our main target of mountain lions alongside other animals and background noises. The mountain lion sounds will be tested at different audio levels. A variety of background noises and other animals will also be tested.

Expected outcome:
Success: The sensors accurately detect mountain lion sounds with threat levels. It ignores irrelevant sounds, and returns a high accuracy score.
Failure: The sensor fails to detect mountain lions or incorrectly alarms irrelevant audio

Pseudo Code

```
TestAudioDetection():
    sensor.initialize()
    play_audio(mountain_lion_roar)

   if sensor.detected_animal == "mountain lion":
        print("Test Passed: Mountain lion detected")
    else:
        print("Test Failed: Detection error")

    play_audio(background_noise)
    if sensor.detected_animal == None:
        print("Test Passed: No false detection")
    else:
       print("Test Failed: False positive detection")
```

II. Test location detection accuracy

During this test we will have the same setup with our previous test of an open area where we will simulate a variety of noises, specifically mountain lion noises. We will be

testing the systems ability to accurately detect the location within the required 3 meter radius of the source.

Expected outcome:
Success: The system accurately detects the location of the noise within 3 meters of the actual source.
Failure: The system reports a detection more than the 3 meter radius indicating an issue with our detection algorithm.

Pseudo Code

```
TestLocationDetectionAccuracy():
    sensor.initialize()
    known_location = (50, 30)
    simulate_noise_at_location(known_location)
    detected_location = sensor.detect_location()

    if calculate_distance(known_location, detected_location) <= 3:
        print("Test Passed: Location detected accurately within 3 meters")
    else:
        print("Test Failed: Location detection exceeds 3-meter accuracy")

     edge_location = (5000, 5000)
     simulate_noise_at_location(edge_location)
     detected_edge_location = sensor.detect_location()

    if calculate_distance(edge_location, detected_edge_location) <= 3:
        print("Test Passed: Edge location detected accurately within 3 meters")
    else:
        print("Test Failed: Edge location detection exceeds 3-meter accuracy")

     outside_location = (7000, 7000)
     detected_outside_location = sensor.detect_location()

     if detected_outside_location == None:
        print("Test Passed: No location detected for out-of-range noise")
    else:
        print("Test Failed: False detection for out-of-range noise")
```

2. Functional Testing

I. Testing the Alarm Activation and Deactivation

The goal of this test is to ensure that the alarm will activate and deactivate with the detection of a mountain lion. To test this, we will provide a real mountain lion sound that has already been captured, as well as a nature audio sample that does not contain a mountain lion. Upon providing these sounds, we will have a boolean be turned to TRUE if the alarm is playing, and FALSE if it is not. If it is false, the system will automatically fail. If it is true, the system will continue and test whether the alarm can be turned off by manually turning off the alarm, then rechecking the boolean.

Expected Outcome:

From this test, we expect for the alarm to sound when the mountain lion sound is fed to the machine, and to not play the alarm when a sound is provided that does not contain a mountain lion. The combination of these two inputs should provide an accurate understanding of whether the alarm will sound for a mountain lion.

Pseudocode

```
mountainLionSound = new Sound(“mountain_lion”)
sensor.detectSound(mountainLion)

boolean alarmState = rangerStation.checkAlarm()
if alarmState = true:
        print “alarm was triggered successfully”

        ranger.turnOffAlarm()
        alarmState = rangerStation.checkAlarm()
        if alarmState = true:
                print “alarm turned off successfully”
        else:
                print “alarm did not turn off: FAIL”
else:
        print “alarm did not sound: FAIL”

fakeSound = new Sound(“not_mountain_lion”)
sensor.detectSound(fakeSound)

boolean secondAlarmState = rangerStation.checkAlarm()
if secondAlarmState = true:
```

```
        print “alarm was triggered: FAIL”
else:
        print “alarm did not sound succesfully”
```

II. Testing the Report Generation

The goal of our report generation is to be able to retrieve reports of the mountain lions with the dates, location, and threat level classification. In order to test this, we would provide simulated mountain lion sounds to the sensors. We would provide several of different audios, allowing for several testing of whether the locations are valid, the sounds are accurate, and the threat is accurate.

Expected Outcome:

From this test, we expect the report to contain all accurate information gathered from the fake sounds that are provided.

Pseudocode

```
for i from 1 to 100:
        sound = simulateSound(randomMountainLion())
        sensor.detectSound(sound)
        Boolean nextDay = randomBoolean()
        if nextDay == true:
                currentDay += 1

report = ranger.requestReport(“mountain_lion”, lastDays = 30)

if report.animalType == sensor.getAnimals()
        print “report accurately compiled animals”
else:
        print “report did not compile animals”
if report.animalDate == sensor.getAnimalDate()
        print “report accurately logged the date the animals were sensed.”
else:
        print “report did not compile the date the animals were logged”
if report.animalType == sensor.getAnimalRange()
        print “report accurately compiled animal locations”
else:
        print “report did not compile animal locations”
if report.animalType == sensor.getAnimalThreat()
        print “report accurately compiled animal threat levels”
else:
```

print “report did not compile animal threat levels”

3. System Testing

To test the functionality of our Mountain Lion detection system we will ensure that our requirements are met in order to confirm that all aspects of the system behave accordingly and produce the expected outcomes.

I. Testing an animal walking by the Sensor:

A simulated animal would be walking through within the range of the sensor, ensuring that even the lowest amount of noise volume be detected. It would be recorded and sent back to the system where various test levels of analysis and classification would be performed.

Expected Outcome:
Data would be sent to the system in which the sound would be analyzed to determine its location and threat level which in addition would be sent to the database for later use. The alarm will go off immediately at the location of the threat level if it's too high and continue sounding until the park ranger receives a notification of it and manually turns it off.

II. Testing the Report Compilation:

A report would be generated by retrieving the recorded data and animal classification from the database in which the park ranger will be able to access and view all the information present. The ranger would be able to navigate through each report classified either as date or threat level to begin analysis and make conclusions.

Expected Outcome:
Reports should be generated accurately with as much details as possible and the park ranger can easily navigate and view them as a pdf file.