Group 9 Mountain Lion Detection System

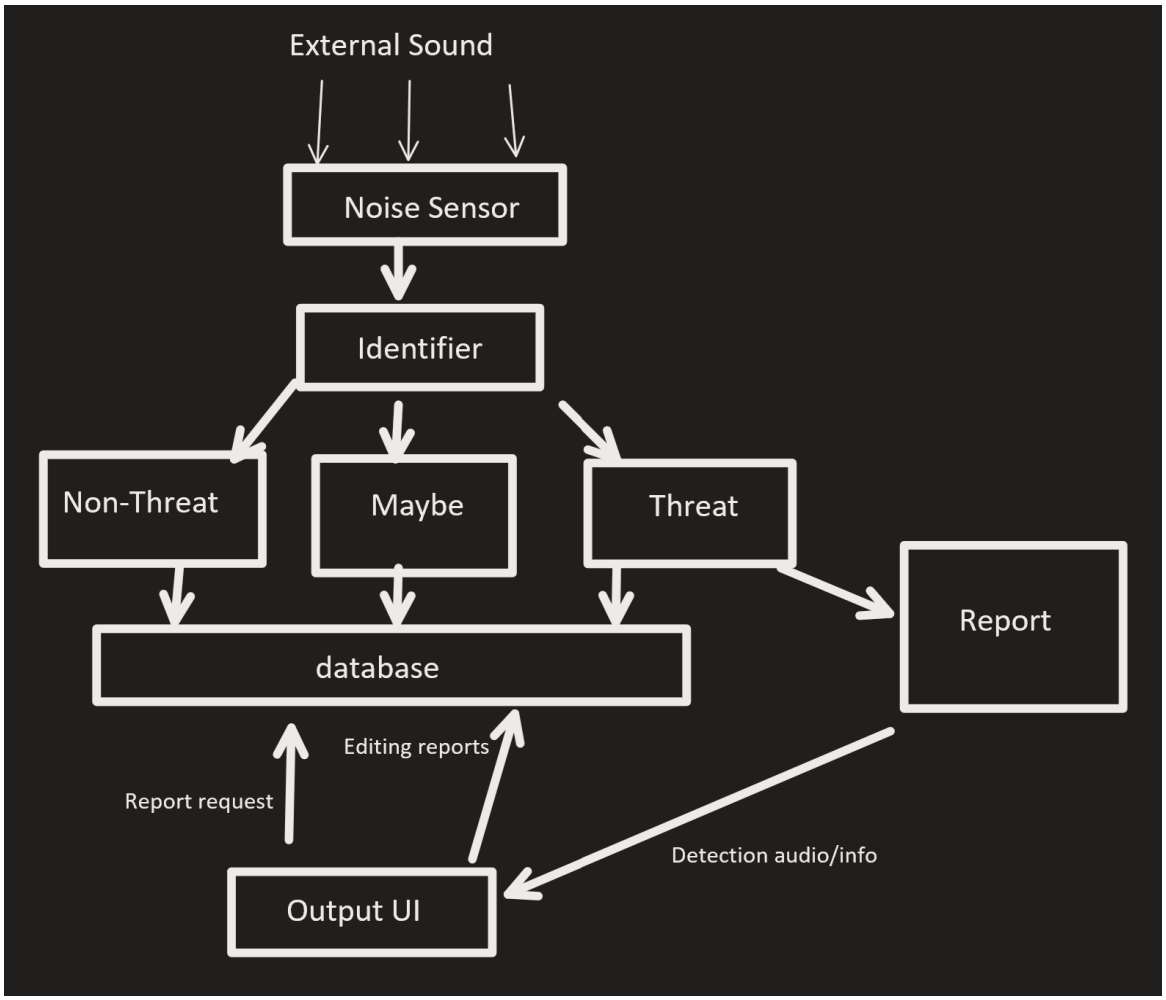
Software Design 2.0

- Revised with Official Data Management Strategy -

Team Members: Alberto Ames, Carlos Paredes, and Bobby Tomlinson

Software Architecture of Group 9 MLDS

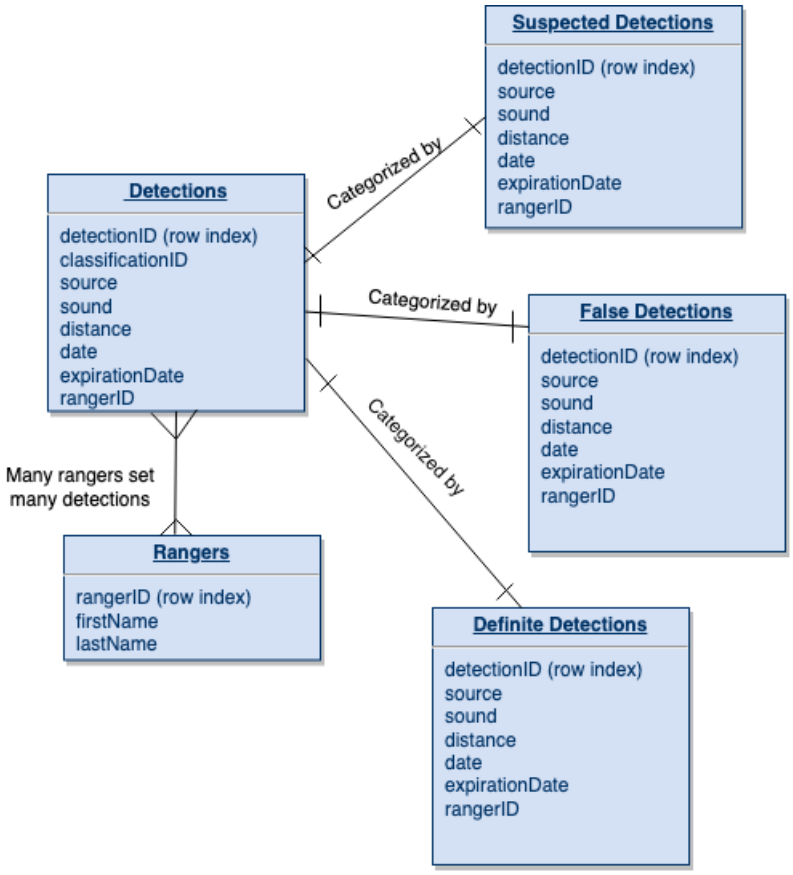
Architectural Diagram of All Major Components



The main components of the Mountain Lion Detection System consist of a noise sensor which takes in audio data, an identifier system which categorizes the sounds as threats, non-threats and maybe-threats, the database which holds reports of all audio detections, a report-generating system which sends reports to the user depending on the category of the sound, and a user interface which allows users to view, edit, and request reports. The user is able to hear audio if a report is generated, and edit the category it is classified under if they determine that the sound doesn't constitute a threat. A report is generated if the audio input is determined by the identifier to be a threat, otherwise the input is simply logged in the database. The user can request to view previous audio input data, and they will be able to edit it.

Data Management Strategy

Diagram of Group 9 MLDS Data Storage



We chose SQL to manage our data because of its power for building relational databases. We feel that this would be especially useful for the mountain lion detection system, as there is a relationship between all detections, detections categorized as threats, and those marked as non-threats. We chose to have different SQL databases for Detections, False Detections, Suspected Detections, Definite Detections, and Rangers. This is because there are different things the user can/will do depending on the category of the detection. Having distinct, separated databases for each classification of detection, and for each detection associated with a ranger, allows for reports to be built quickly. Each detection in the detections SQL table is categorized into one of 3 detection classification tables, and so this cardinality is 1:1. This is because this relationship is simply a categorization of entries in the detections database, and so

there will never be more than one entry for the same detection in each database. This 1:1 cardinality is preserved through a detectionID attribute that is used by the system to quickly access detections in the different tables. Detections are sorted into classification tables by their classification ID's, with each classification table storing one type of classification ID. The Ranger should also be able to view and edit all detections, regardless of threats, so having a Detection database composed of all detections is important. One possible drawback of using SQL as opposed to NoSQL is that it is comparatively more expensive to scale. However, we feel that the ability to create relational databases, and the fact that consistency is important for a system that categorizes threats, makes SQL the superior data management tool for this problem. Since detections are removed from all databases after a year (due to their lack of relevance after this point, as specified in the SDS document), using multiple SQL tables to store this data shouldn't be too cost-inefficient, and it should allow for quick reporting.