**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Software Design

Document

for

Runway FOD Detection

SE490 - Group 2

William Doyke

Chi A

Jacob Belz

# 

# Table of Contents

[**Table of Contents**](#_aziss7z36kug) **2**

[**1.0. Introduction**](#_baugx43gstfo) **3**

[1.1. Purpose](#_tq1kq6vvkm27) 3

[1.2. Scope](#_cb7ovvm12xza) 3

[1.3. Glossary](#_4tl3ltsh9b0u) 4

[1.4. References](#_k57xje38rnkk) 4

[1.5. Overview of Document](#_vkisbh2vg8vo) 5

[**2.0. Deployment Diagram**](#_x3rbmk2yfpia) **6**

[**3.0. Architectural Design**](#_3galea249kp5) **7**

[**4.0. Data Structure Design**](#_x04wthby8fna) **9**

[4.1 Data Description](#_26in1rg) 9

[4.2 Data Dictionary](#_lnxbz9) 10

[**5.0 Use Case**](#_yu7q62u7d6h4) **Scenario** **12**

[**6.0 User Interface Design**](#_loyy9562ila7) **17**

[6.1 Overview of User Interface](#_wtdskxue0910) `17

[6.2 Screen Images   
 First Prototype, will change in future according to feedback.](#_2jxsxqh) 17

[6.3 Screen Objects and Actions](#_z337ya) 18

[**7.0 Help System Design**](#_ayk3ulhhf447) **19**

**8.0 CRC Cards 19**

**9.0 Test Cases 20**

# 1.0 Introduction

## 1.1 Purpose

## This software design document (SDD) describes the architecture and system design of the foreign object debris detection software we are currently developing and is intended for the use of students, university faculty, anyone who is interested in aviation safety, or related technology advancements.

## 1.2 Scope

Keeping runways clear of debris is important to the safety of airport operations. The primary goal of this software is to utilize AI for object detection to assist airport staff.

We’ve all been there or experienced that “anxiety” that comes with airline travel. Especially when a flight gets canceled, delayed, or there are issues landing so the plane has to circle around the airport for some time until everything is under control.

Our foreign object debris detection software would help with that. This is because each every day thousands of flights take off and land on airport tarmacs all around the country, but we still haven’t found an efficient way to keep those runways clear of any objects and might affect landing/take off of an aircraft. This is one of the many reasons why our travel plans unexpectedly change without us even knowing in advance.

What our software intends to do is with the help of a high definition camera and machine learning capabilities it will be able to detect foreign object debris on runways quickly and efficiently giving airport staff time to clean up runways and most importantly keep our travel plans on schedule.

In order to complete the task mentioned above there are a series of goals and objectives our software must achieve to make this technology into a reality.

The top priorities include:

* The ability to detect FOD on the runway including:
  + Bolts, screws, tools, people, animals, and any other large debris
* The implementation of a machine learning model to recognize FODs
* The software useability to be friendly enough to be used by airport personnel

Along with these goals we have also identified the objectives that would be most important to keep track of throughout the development lifecycle of this software. These include:

* Acquiring domain knowledge by talking to experts regarding FOD
* Using machine learning to detect and recognize the FOD
* Use phones/cameras/drones/datasets to collect images for training a machine learning model to detect common types of FOD
* Use of notifications notifying airport personnel when FOD is located on a runway/tarmac

## 1.3 Glossary

SDD - Software Design Document

FOD - Foreign Object Debris

## 1.4 References

## [Tensorflow Object Detection in 5 Hours with Python | Full Course with 3 Projects](https://www.youtube.com/watch?v=yqkISICHH-U) [How to Install TensorFlow Object Detection in 2020 (Webcam and Images!)](https://youtu.be/usR2LQuxhL4)

<https://neptune.ai/blog/how-to-train-your-own-object-detector-using-tensorflow-object-detection-api>

<https://neptune.ai/blog/tensorflow-object-detection-api-best-practices-to-training-evaluation-deployment>

Datasets accessible from group’s gmail account (google drive):

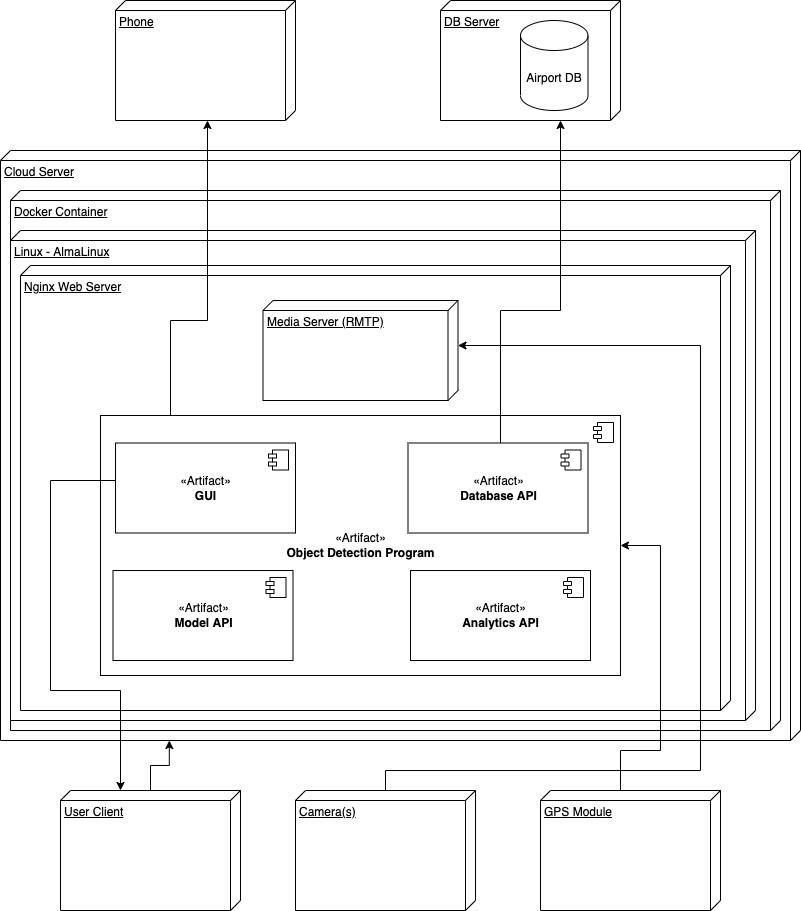
scsufodteam490@gmail.com

## 1.5 Overview of Document

This document includes an introduction which gives a summary of the project and goals/objectives to be achieved along the way (diagrams to better explain these are presented in the ladder sections of this document), a system overview and system architecture which look at the design and architecture of our software, data and component design diving deeper into what data sets will be implemented and how, and finally an interface design to give a rough idea to a user of how future iterations of the software will look like.

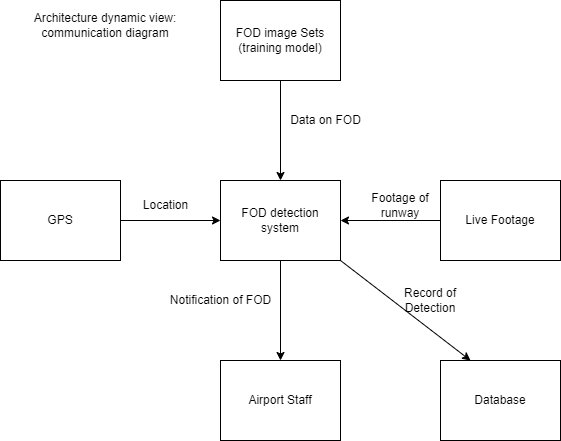
# 

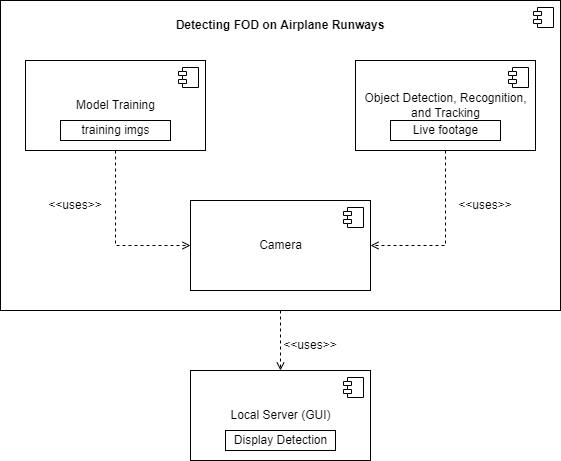
# 2.0 Deployment Diagram

**

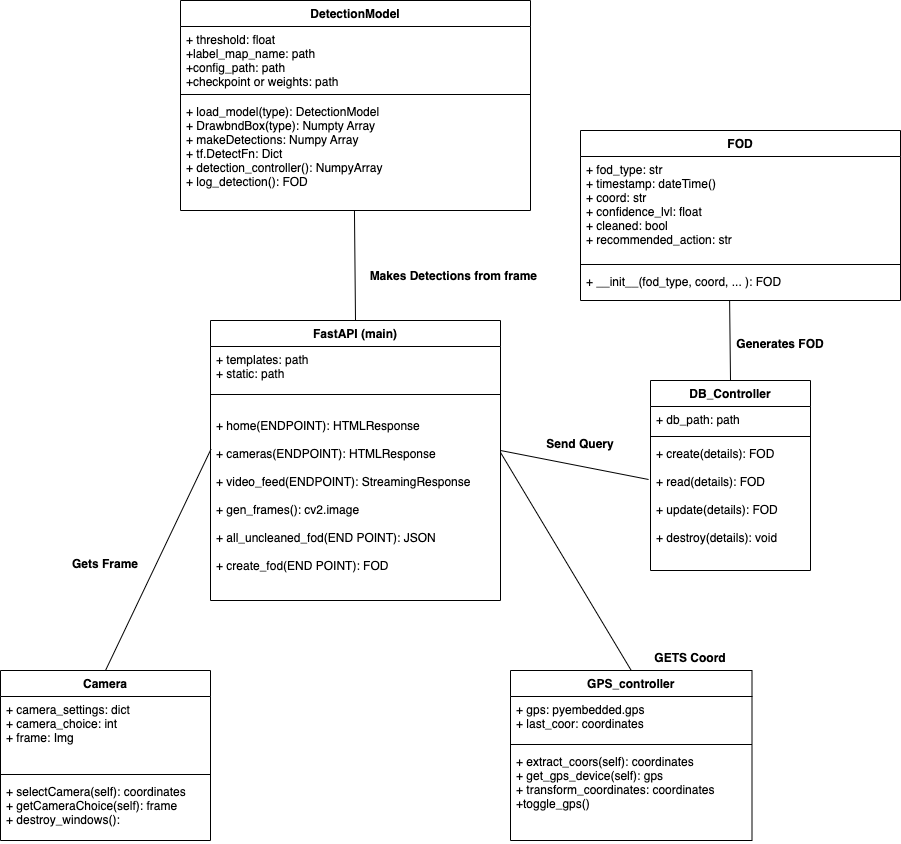
# 3.0 Architectural Design

Architecture: Context Model

**

Architecture: Structural View

Class Diagram



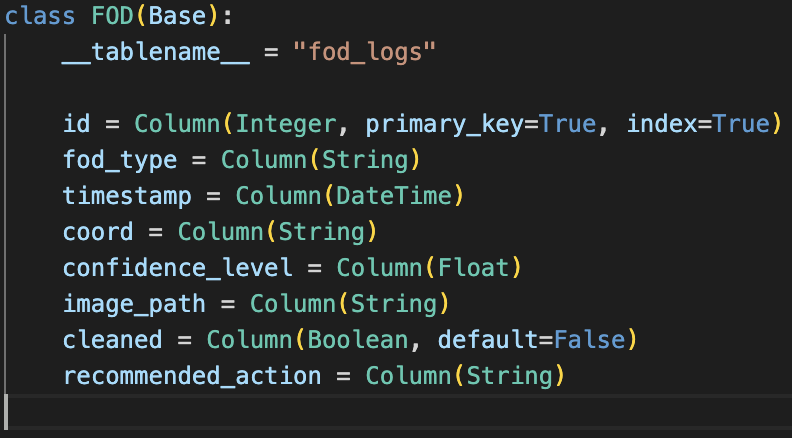
# 4.0 Data Structure Design

## 4.1 Data Description

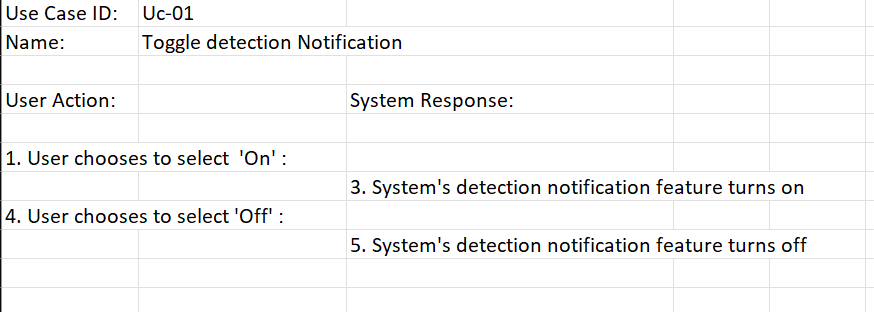
The data that we collect starts as an image file (jpeg or png) and then must be labeled. Labeling an image is when you are putting a box around the item of interest that is in the image. When this process is performed the image is converted to an xml file. With this xml file you are able to create the TFRecord that can be used to train the tensorflow model. To create our dataset we have to collect images, either using publicly available datasets or by taking our own photos and videos. Currently we have 7 FOD types: pliers, screwdrivers, a piece of metal, a piece of wood, bolts, nuts and wrenches. This catalog of objects can be expanded later.

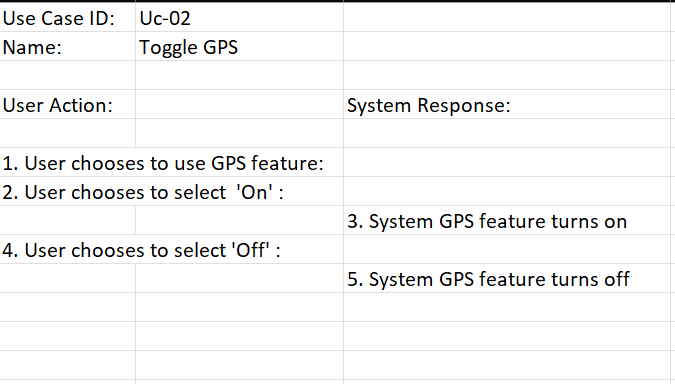
## 4.2 Data Dictionary

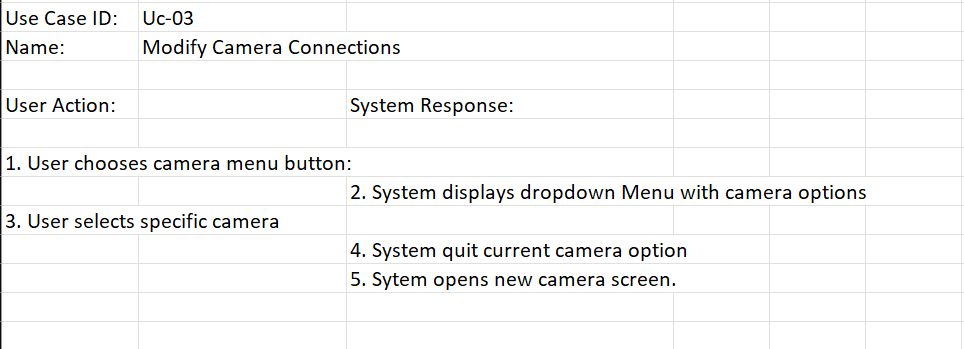
* Dataset - A collection of the images that will be used
* TensorFlow - a machine learning framework used for object detection
* TFRecord - A file format that tensorflow can use for the training of the model
* XML - A file format that stores the object labels corresponding to each image
* Detection List - List of Tensors with data about Detected objects
* Input Tensor - Data Array filled with data about the images
* GPS\_Controller – Object that encapsulates functions and data related to the GPS device
* Detection – Object that encapsulates functions and data related a Detected Object
* Frame – Variable containing image related data, frame of each video.
* Model Interface- Rest API built in Python using Fast API
* FAST API - An Instance of the Web framework allowing for web functionality
* FOD - Pydantic model to validate and create instances of FOD
* DB Controller – Connects to DB

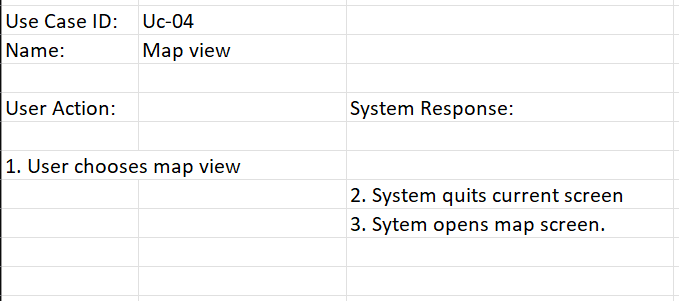


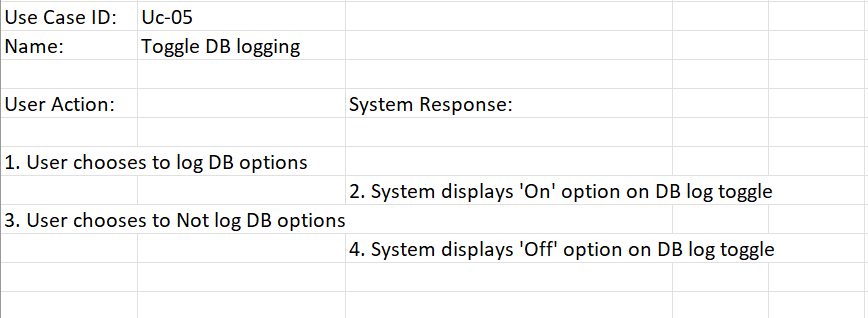
# 5.0 Use Case Scenarios

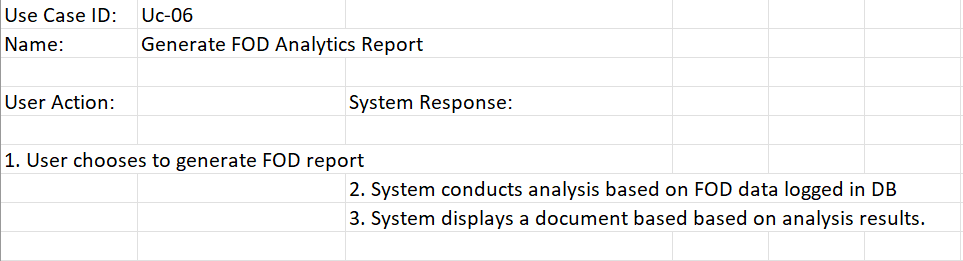




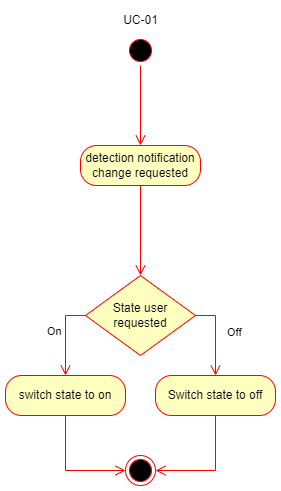
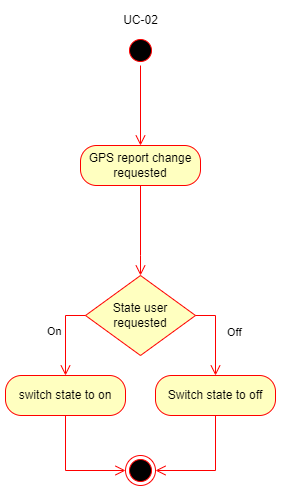


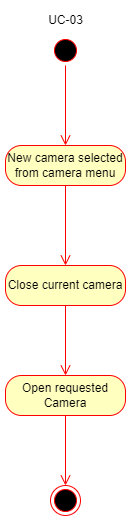
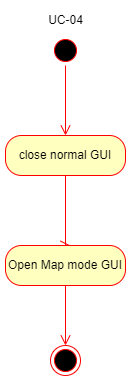


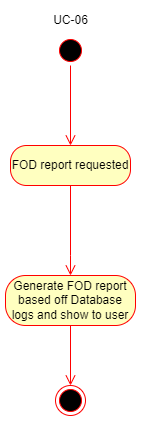




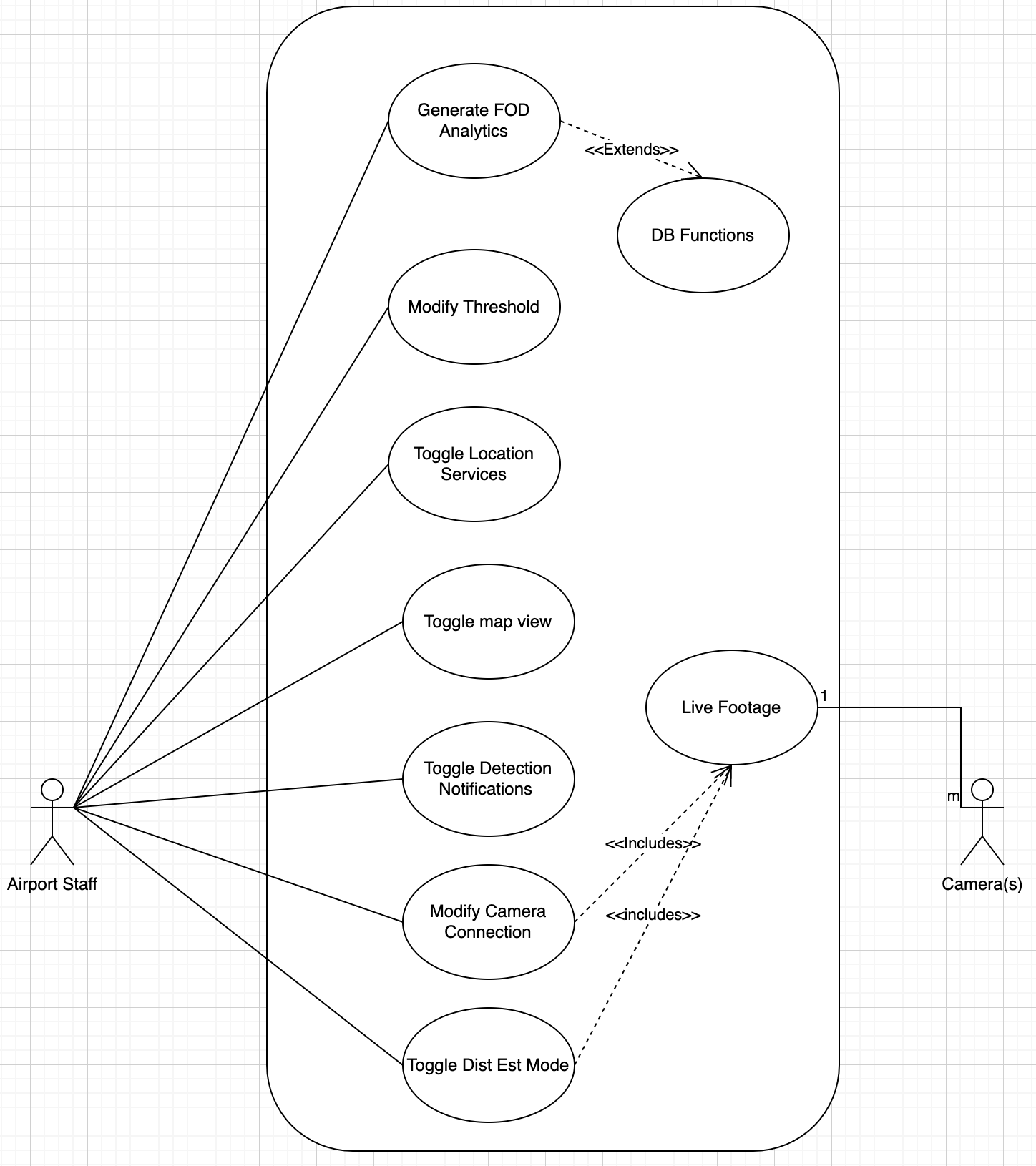
Use Case Scenarios Activity Diagram:

 `



Use Case Model:

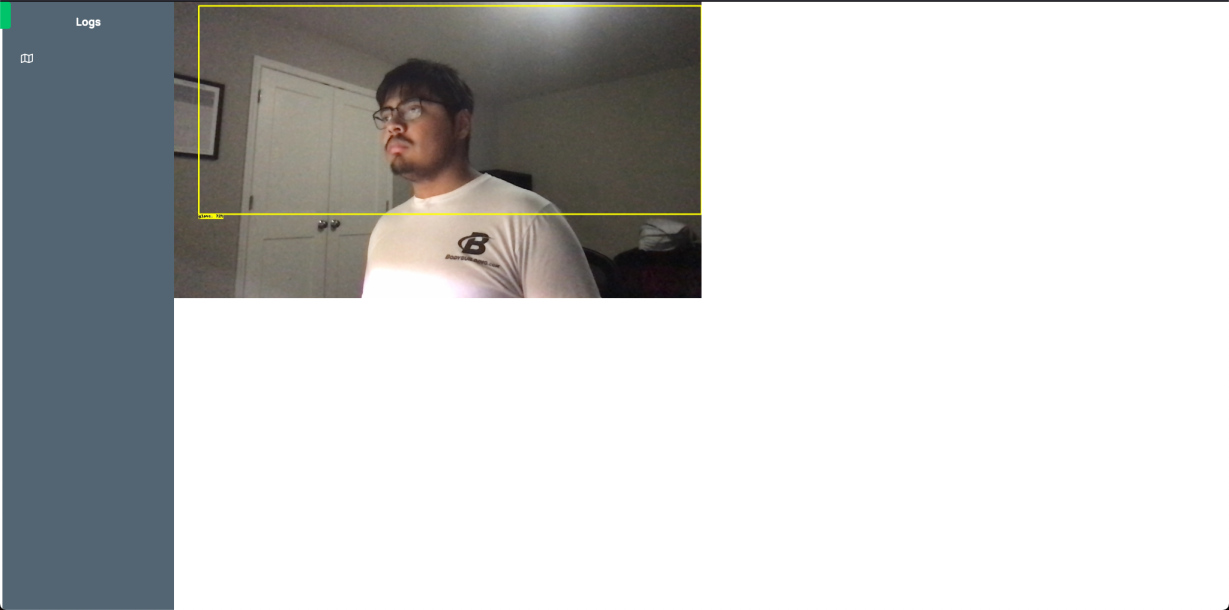


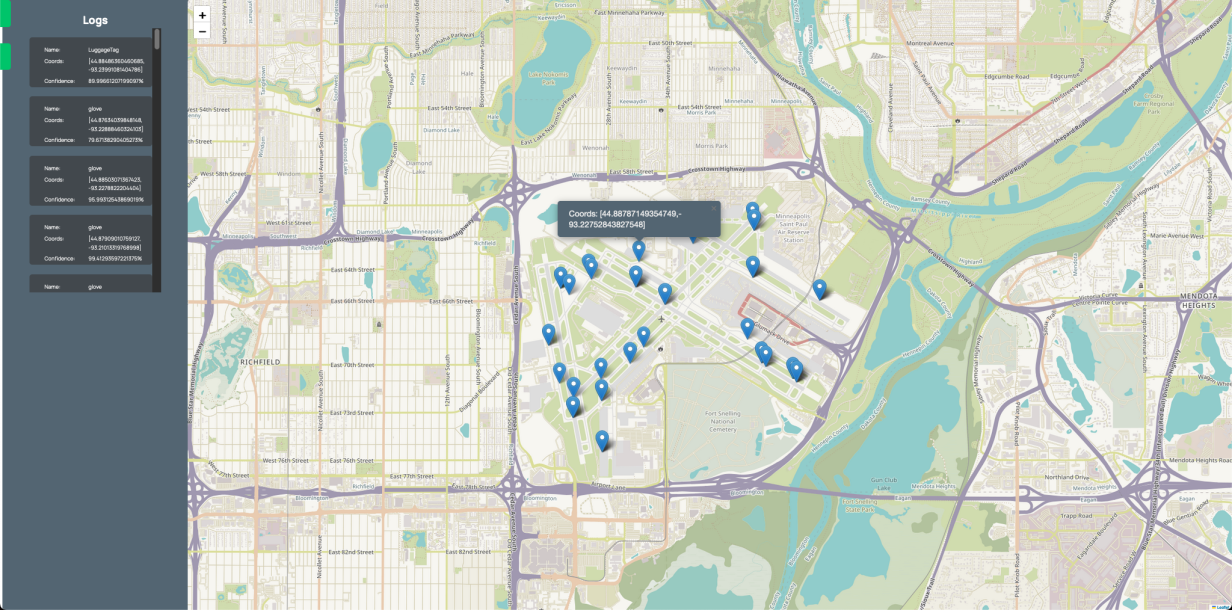
# 6.0 User Interface Design

## 6.1 Overview of User Interface

Users will be able to see images that are coming from the cameras, and if there is a detection the program will alert the user and will show the image that has a detection.

## 6.2 Screen Images Current Prototype, will change in future according to feedback:





## 6.3 Screen Objects and Actions

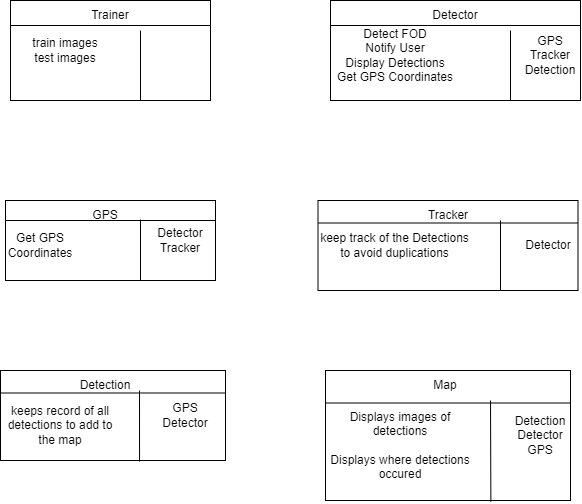
System will show multiple cameras, each mounted on different points of the vehicle. When an object is detected by the AI, the following things will happen:

* Bounding box appears around that object, with a confidence rating and the prediction class
* The entire frame for that camera will get a colored border as well, to draw attention to that specific camera
* Sound effect will play, alerting the user in case they aren’t looking at the screen
* The detection info will be written to a log (not implemented)

# 7.0 Help System Design

We will include a user manual that explains the user interface and how to interpret detection results. The user manual will be created later in our timeline, as the software is subject to change in the meantime. We will also explain the process of setting up the software on a new machine.

# 8.0 CRC Model



# 9.0 Test Cases

| Test ID | Description | Steps | Data | Expected Result | Actual Result | Pass/fail |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Check ability to function on windows | 1. Install program on a Windows environment  2. Run program with a device connected to a camera |  | Camera connects, GUI opens,  The command line reports detected objects / coordinates |  |  |
| 2 | Check ability to function on mac | 1. Install program on a Mac environment  2. Run program with a device connected to a camera |  | Camera connects, GUI opens,  The command line reports detected objects / coordinates |  |  |
| 3 | Check whether GPS connects | 1. Run the program | GPS connected to the device report the port used but it should be able to use any port | Command line should report a connected GPS |  |  |
| 4 | Check how program functions with no gps connection | 1. Run the program | No gps connected to device | Command line should report there being no GPS to connect to but still detect object with the camera |  |  |
| 5 | Turn off the GPS functionality with a connected GPS | 1. Run the program  2. After initial start up, click the GPS button | GPS connected to the device report the port used but it should be able to use any port | The command line will state the GPS is turned off/disconnected and no longer report coordinates |  |  |
| 6 | Turn on GPS functionality with connected GPS after it has been previously turned off | 1. Run the program  2. After initial start up, click the GPS button  3. Wait until it no longer reports coordinates and click the GP button again | GPS connected to the device report the port used but it should be able to use any port | The command line report a GPS connection and begin to report coordinates for each detection |  |  |
| 7 | Turn on GPS functionality without a connected GPS | 1. Run the program  2. After initial start up, click the GPS button | No gps connected to device | The command line may display extra text but should not start reporting coordinates |  |  |
| 8 | Check Program  Start up time – Performance Testing | 1. Run the program, start timing  2. Stop timing once the first object is detected in command line |  | Less than 10 seconds, 3 seconds is ideal |  |  |
| 9 | Check GPS turn off for lag | 1. Turn on Program  2. Hit GPS button, start timing  3. Stop timing once the program starts report the lack of connected gps to the command line | GPS connected on a port | Less than 3 seconds but less than 1 seconds is ideal |  |  |
| 10 | Check GPS turn on for lag | 1. Turn on Program  2. Hit GPS button  3. Wait 5 seconds hithe GPS button and start timing  3. Stop timing once the program starts report the coordinates to the command line | GPS connected on a port | Less than 3 seconds but less than 1 seconds is ideal |  |  |
| 11 | Check Map Feature | 1.Run program  2. Click Map button | GPS connected on a port – Gps Coor, Map, Webpage | Upon clicking the button, the user sees a map with tacks marking the location of detected FOD. If they click the tack, information about the FOD appears. |  |  |