Software Requirements Specification

for

Runway FOD Detection

SE490 - Group 2

William Doyke

Chi A

Jacob Belz

**Table of Contents**

**Table of Contents [ii](#_heading=h.gjdgxs)**

**Revision History [ii](#_heading=h.30j0zll)**

**1. Introduction [1](#_heading=h.1fob9te)**

1.1 Purpose [1](#_heading=h.3znysh7)

1.2 Document Conventions [1](#_heading=h.2et92p0)

1.3 Intended Audience and Reading Suggestions [1](#_heading=h.tyjcwt)

1.4 Product Scope

[1.5 Requirement Completeness](#_heading=h.3dy6vkm)

1.6 References 2

**2. Overall Description 3**

2.1 Product Perspective 3

2.2 Product Functions 4

2.3 User Classes and Characteristics 4

2.4 Operating Environment 5

2.5 Design and Implementation Constraints 5

2.6 User Documentation 5

2.7 Assumptions and Dependencies 5

**3. External Interface Requirements 7**

3.1 User Interfaces 7

3.2 Hardware Interfaces 7

3.3 Software Interfaces 7

3.4 Communications Interfaces 8

**4. System Features / Functional Requirements 8**

4.1 Detect an Object 8

4.2 Recognize an Object 9

4.3 Connect to Cameras 9

4.4 Live Detection on GUI 9

4.5 Retrieve GPS coordinates 10

4.6 Log detections 10

4.7 Generate recommend action upon detection 10

4.8 FOD Analysis 11

**5. Nonfunctional Requirements 11**

5.1 Performance Requirements 11

5.2 Safety Requirements 11

5.3 Security Requirements 11

5.4 Software Quality Attributes 11

5.5 Business Rules 12

**6. Other Requirements 16**

**Appendix**

# Introduction

## Purpose

This software requirements specification (SRS) describes the software requirements that relate to our Foreign Object Debris (FOD) detection software and is intended for the use of students, individuals interested in aviation safety, and airport staff. We are creating this system to provide a client server solution through the browser to allow various airports to use our system. We wish to extend the proof of concept of detecting FOD on an airport runway through Computer Vision (CV) in order to reduce costs related to delays and airplane repairs that come as a result of FOD on an airport runway.

## Document Conventions

This document includes an introduction which gives a summary of the project and product scope to be achieved along the way (diagrams to better explain these are presented in the ladder sections of this document), an overall description which look at the requirements of our software, interface requirements which detail how our software will look to a user, system features which look into the project as a whole and what should be included in the software’s development, and finally non-functional requirements looking at software’s goals for the future (reliability, scalability, etc.).

## Intended Audience and Reading Suggestions

The intended audience for this software include students, university faculty, airport personnel, and individuals interested in aviation safety..

It is strongly recommended to read this specification from top to bottom, however for a quick synopsis refer to the Product Scope (Section 1.4) and diagrams (Section 2).

## Product Scope

Our software intends to accomplish its purpose by implementing a combination of high definition cameras and machine learning data models. In doing so, this will help us to detect foreigh object debris on runways quickly and efficiently giving airport staff time to clean up runways, cut down on the cost of repairs and most importantly keep travel plans on safe and on schedule.

In order to complete the task mentioned above there are a series of goals and objectives our software must include to make this technology into a reality. To start, the goals we have identified as most important for this software to achieve or be capable of doing are as follows:

* The ability to detect FOD on the runway including:
  + Bolts, screws, tools, people, animals, and any other large debris (tentative list as we evaluate most relevant FOD for MSP or other regional airports)
* The implementation of a machine learning model to recognize FOD
* The software performance being able to keep up with live FOD detection
* The delivery of reliable FOD analytics
* User friendly software intended for airport personnel
* The ability to place detections on a live map

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 computer vision to detect and recognize the FOD
* Use phones/cameras/drones/datasets to train a model to detect common types of FOD
* Use of notifications notifying airport personnel when FOD is located on a runway/tarmac

## 1.5 Requirements Completeness

Our process of achieving requirements agreements included exercises where we placed ourselves in the position of an airport staff member trying to use our software and discussing what we would like to make this software as intuitive and enjoyable to use as possible. We have also taken the time to analyze the process of runway cleaners via FAA literature and discussion with airport staff.

## 1.6 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://github.com/scaltintasli/Airport-Runway-FOD> – Previous Groups Repo

# Overall Description

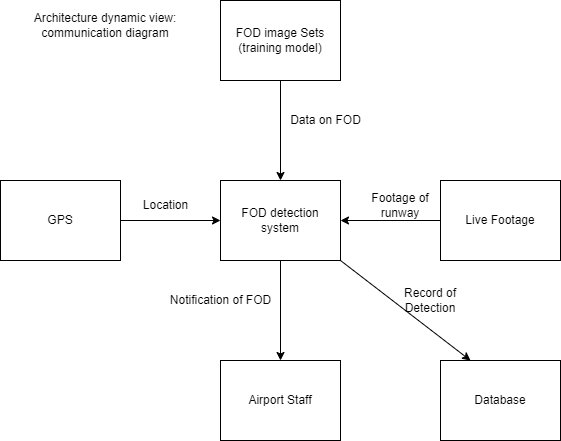
## Product Perspective

The system will be used for automating a process that is currently done manually, FOD detection. Manual detection by humans is prone to error due to a number of reasons, such as training, fatigue and limitations of human eyesight.

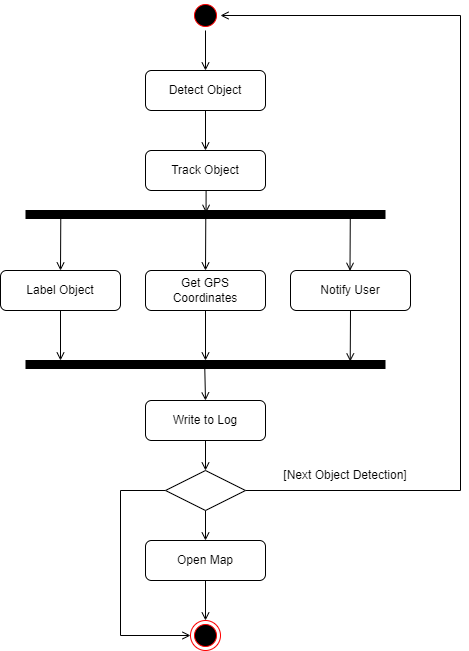
With AI, we can automate this process by training a software system composed of machine learning models to accurately detect FOD that humans could fail to notice.

By doing this, we intend to make runways safer for aircraft, pilots, passengers, and property. This can reduce the cost of airport operations and potentially save lives.

***Context Model:*** *Details an overall breakdown of the system on a general level. Which includes FOD image sets to be used for model training, live detection of runways after image training of the system completed, GPS Device to retrieve the GPS coordinates, and notifications from the system coming to airport personnel completing live detection cycles.* Along with Database connectivity to send information regarding detected FOD.



***Activity Diagram:*** *Details an overall breakdown of activities done by the system. Which include detection of a FOD object, tracking the object to remove detection duplications, notifying the user of the detection while also labeling the object with an FOD type and getting the GPS coordinates, then writing to the log. User then has the option to open the map, close the software, or move onto the next object.*



## Product Functions

The major functions planned include:

* Connecting to live video streams of runway activity
* Detecting an object on the runway as possible FOD
* Informing airport personnel of FOD when detected
* Giving user data insight into FOD
* Cloud connectivity
* Connecting multiple cameras
* Analytics on the FOD detected

## User Classes and Characteristics

Potential user classes:

* Runway personnel - people who will need to physically remove FOD once detected. They need to know the location of the FOD.
* Maintainers of the software - software engineers who will need to address any issues with the software, as well as improve the model with new training data.
* Pilots - who need to be aware of safety conditions on the runway when taking off and landing aircraft.

## Operating Environment

The software is expected to perform in a variety of weather and lighting conditions (daylight vs artificial light at night). The development and testing will be performed on a Microsoft machine with Windows 10 or Linux operating software.

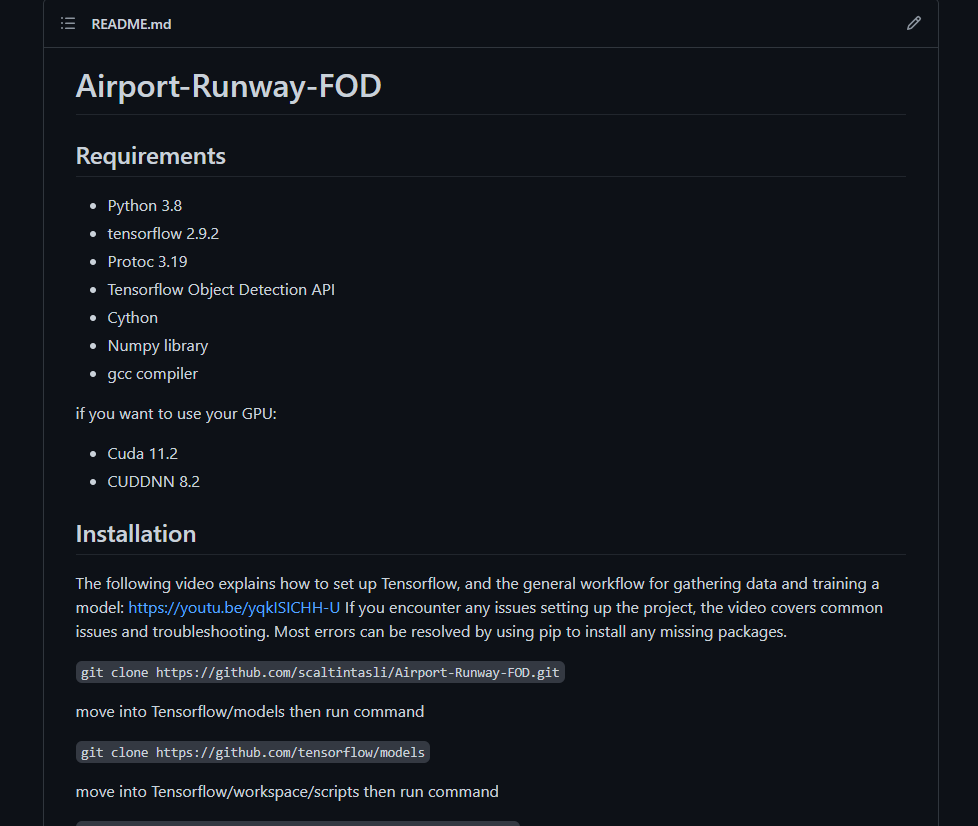
## Design and Implementation Constraints

Constraints include:

* Quality of images available for training data
* Hardware limitations for capturing images/video
* Hardware limitations for training an accurate model in a timely manner
* Hardware limitations for storing a model of an appropriate size (less than 500GB)
* Must operate within airline regulations and safety standards

## User Documentation

A user manual will be provided with instructions for use cases and general operation of the software, as well as troubleshooting.



Snippet of the *README* detailing installation and usage instructions for future user.

## Assumptions and Dependencies

The machine learning model is being trained with objects and camera angles that we received from our conversations with MSP airport personnel. We have established contact with these airport professionals in order to validate our assumptions so that we can train the model effectively by simulating real-world operation of the software.

We also assume that runways are sufficiently lit so that no night vision equipment is required.

Depending on how effective our model is, we may need to implement a way to continue training the model after deployment by supplying more labeled training images.

# External Interface Requirements

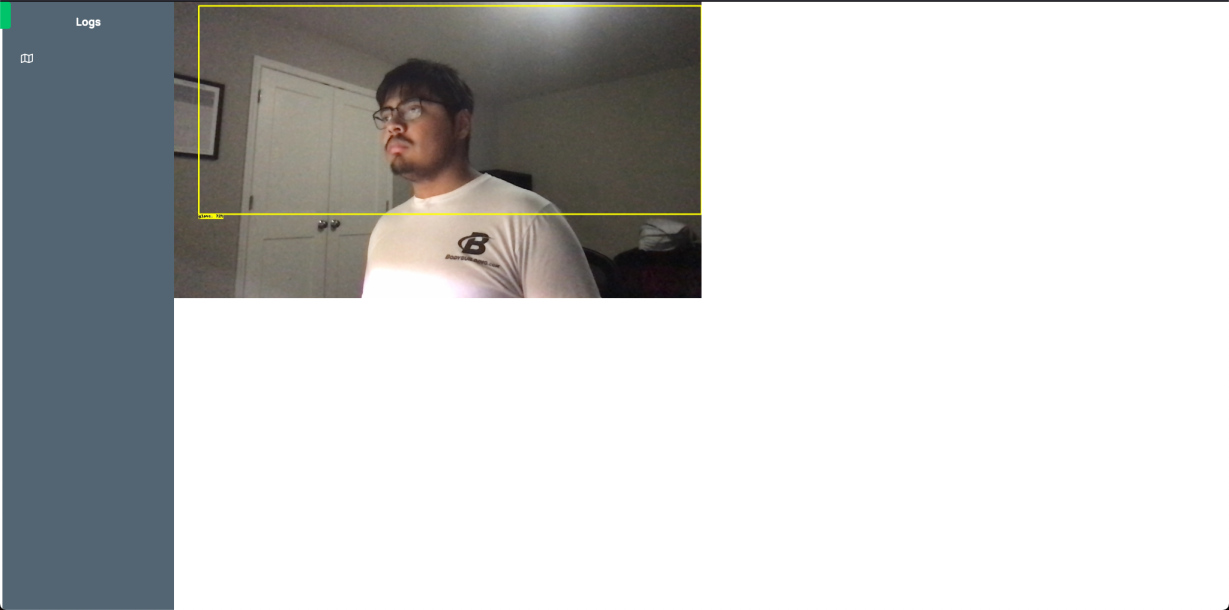
## User Interfaces

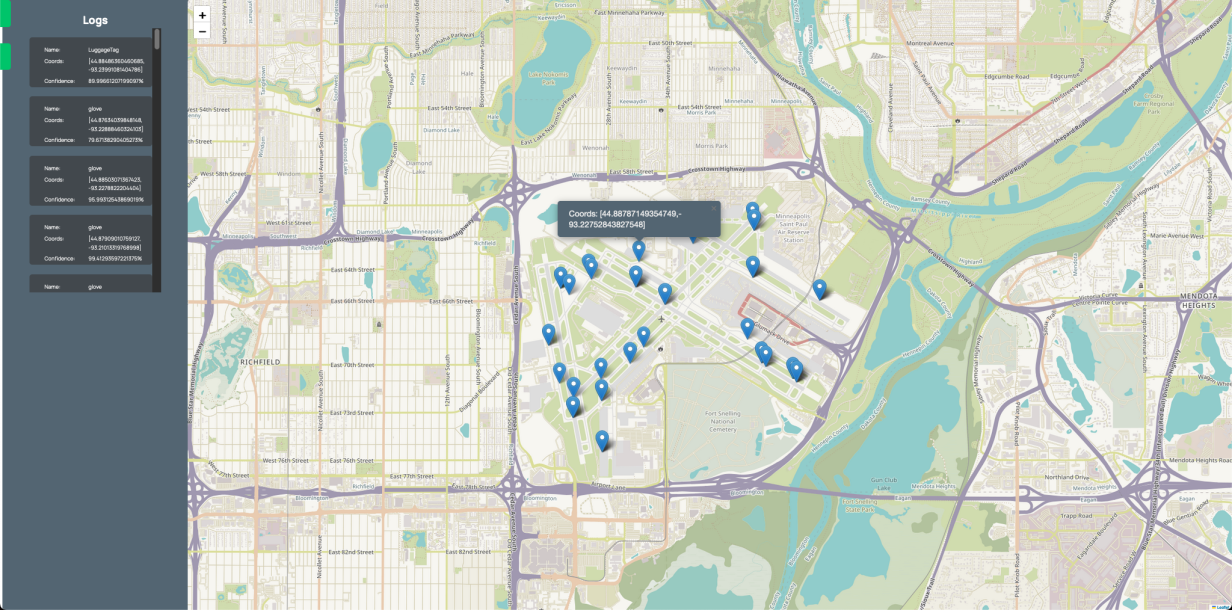
The user interface for the application shall be an application that you are able to download to your machine and connect to the cloud to handle the computation.

## Hardware Interfaces

Hardware includes a camera(s),a GPS, and a computer. We’ll need to connect the software to the runway cameras. We may also extend it to include a radar device as a supplement.

## Software Interfaces

 ***Camera Mode GUI:*** *The previous is a screenshot of the GUI of our software displaying the camera(s) available for detection.*



***Camera Mode GUI:*** *The previous is a screenshot of the GUI of our software displaying a map of all the detections and their coordinates along some information about the coordinate.*

***DETECTION Screenshot:*** *The following is the first mockup GUI currently being employed with our software and shown are various camera angles including a detection of an object from Camera 2’s angle.*

## Communications Interfaces

Our software doesn’t involve any physical communications beyond the connection of cameras via USB-C cable to the personnel laptop running the software for detecting FOD as well as an optional GPS device. We may extend this in the future to include a connection to a radar device. Our hope is to also extend functionality such that we may use IP cameras tied to a network for detection purposes as well.

Our software uses cameras (up to 4) and a GPS to run our software for detecting FOD.

Our system will connect to the cloud, where user data is stored.

# System Features

## Detect an object

4.1.1 Description and Priority

The software must be able to detect an object so we know that something is on a runway. This will help us be able to remove it and protect future damages. The priority of this is high.

4.1.2 Stimulus/Response Sequences

A user would just have to run our software and see if the software has detected any FOD.

4.1.3 Functional Requirements

REQ-1: Software must be trained

REQ-2: Software should give location on where the object is

REQ-3: Software must have a box around the object

## Recognize an object

4.2.1 Description and Priority

The software should be able to name an object that is on the runway. These names must be in the list of what we have trained the software to look for. This is to know what is out on the runway. The priority of this is medium/high.

4.2.2 Stimulus/Response Sequences

A user does not need to do anything for this.

4.2.3 Functional Requirements

REQ-1: The object must first be detected

REQ-2: The object must be trained into the software

REQ-3: The object must be in the list of what we have trained

## Connect to cameras

4.3.1 Description and Priority

The software should be able to connect to your camera(s). We cannot detect any objects without a camera. This makes the priority high.

4.3.2 Stimulus/Response Sequences

A user must make sure at least one camera is connected.

4.3.3 Functional Requirements

REQ-1: Have a camera(s)

REQ-2: Connect to the camera(s)

## Live detection on GUI

4.4.1 Description and Priority

The software should be able to detect and recognize FOD on our GUI in live time with the cameras that are connected

4.4.2 Stimulus/Response Sequences

A user must make sure at least one camera is connected.

4.4.3 Functional Requirements

REQ-1: Have a camera(s)

REQ-2: Connect to the camera(s)

REQ-3: Have our software running

## Retrieve GPS coordinates

4.5.1 Description and Priority

The software should be able to retrieve GPS coordinates if a GPS is connected. If one is not connected, the software will use randomly generated coordinates.

4.5.2 Stimulus/Response Sequences

A user must make sure to connect a GPS device.

4.5.3 Functional Requirements

REQ-1: Have a GPS device connected

REQ-2: Have our software running

## Log of Detections

4.6.1 Description and Priority

The software should be able to provide a list of outputs that say what was detected and on what camera and store them in RDBMS.

4.6.2 Stimulus/Response Sequences

A user must make sure at least one camera is connected.

4.6.3 Functional Requirements

REQ-1: Have a camera(s) connected

REQ-2: Have our software running

## Generate a recommended action upon Detection

4.7.1 Description and Priority

The software should be able to generate a recommended action based upon the FOD detected (i.e sweep, use a magnet, move animal)

4.7.2 Stimulus/Response Sequences

A user must make sure at least one camera is connected and the software is on.

4.7.3 Functional Requirements

REQ-1: Have a camera(s) connected

REQ-2: Have our software running

## FOD Analysis

4.7.1 Description and Priority

The software should be able to generate an analysis every week or everyday to help airport staff gain insight into their FOD issues.

4.7.2 Stimulus/Response Sequences

User opens the app and clicks a button to be taken to the analysis navigation window. From there they can manipulate settings to generate a list of FOD incidents and graphs corresponding to these incidents.

4.7.3 Functional Requirements

REQ-1: Have a camera(s) connected

REQ-2: Have our software running

# Other Nonfunctional Requirements

## Performance Requirements

The system should be able to detect the FOD objects quickly, ideally as soon as it appears in the camera's frame.

## Safety Requirements

The possibility of damage happening to the airplanes is possible if the application does not successfully identify a foreign object that is on the runway. This would come at the cost of damage to the plane or even worse people's lives. This is why it is important that the model will correctly identify objects.This means it is absolutely necessary to minimize the number of false negatives or type two errors.

## Security Requirements

Once the application is developed, the model should not be accessed by any outside sources other than the admins that would have access to the back end of the model.

## Software Quality Attributes

* Availability: Should be available to any airports that need foreign object detection
* Correctness: The application must be correct when it is detecting objects so that no damage will be done to the airplanes
* Maintainability: The application should be easy to maintain, the only thing that will be modified is the model.
* Portability: The application should be able to port to any systems and cameras. If the user is using a GoPro as a webcam, the GoPro software must be downloaded. This software, currently, only supports Windows 10.
* Reliability: The application must be reliable, there should be no errors or malfunctions. App has as much uptime as we can achieve.
* Usability: The application should be easy to use and set up so that users with little technology experience can use it.

## Business Rules

The user should be able to access the object detection but should not be able to access the model as this would be a security risk.

Business Use Case #1

| 1. Description |
| --- |
| This document will provide details of the inner business workflow of our system. |

| 2. Goals |
| --- |
| It will focus on how the business we are build the software for, will view the workflow or process of what is expected from our software |

| 3. Performance Goals |
| --- |
| Recognizing foreign objects and providing sufficient details to quickly and efficiently handle the foreign object and resume business dealings and workflow. |

| 4. r-conditions |
| --- |
| A condition that must be true before the use case can start is a foreign object existing on the runway |

| 5. Workflow | | |
| --- | --- | --- |
| Flow Identifier: Object Detected – upon detecting an object | | |
| Step | User Action | System Response (optional) |
| 1 | Acknowledge existing foreign object on runway | Log the object in database |
| 2 | Record time discovered and location of foreign object |  |
| 3 | Recommend manner to handle foreign object |  |

Business Use Case #2

| 1. Description |
| --- |
| This document will provide details of the inner business workflow of our analytics system |

| 2. Goals |
| --- |
| Generate insight into the FOD activity of the day/week. |

| 3. Performance Goals |
| --- |
| Display information regarding FOD activity and trends within the last day/week in an intuitive manner |

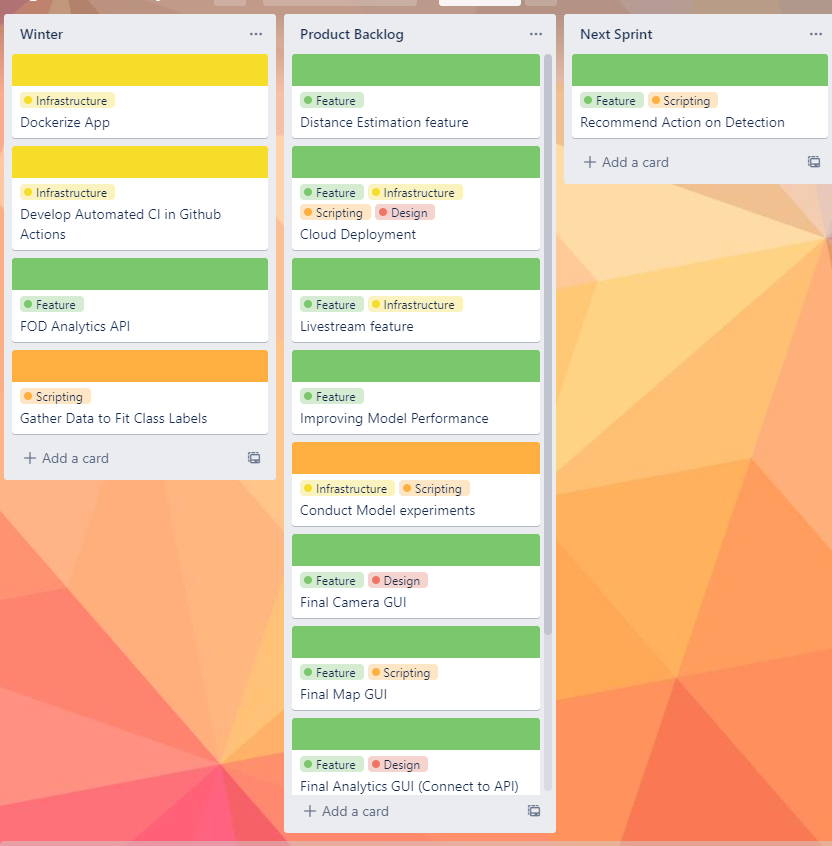
| 4. r-conditions |
| --- |
| A condition that must be true before the use case can start is FOD must be stored in a database |

| 5. Workflow | | |
| --- | --- | --- |
| Flow Identifier: Object Detected – upon clicking analytics button | | |
| Step | User Action | System Response (optional) |
| 1 | Adjust time-period, fod-type buttons | See data regarding trends, coordinates on a map, ability to select multiple class of FOD |

**Appendix**

*Diagrams, datasets, screenshots, etc*

*Backlog:*

**