# CS 340 README

## About the Project - Grazioso Salvare Dashboard

This project is a dashboard application built to help the rescue organization Grazioso Salvare identify ideal dogs for specialized search and rescue training. The application connects to a MongoDB database containing Austin Animal Center outcomes data and presents this data interactively through tables, charts, and geolocation tools.

## Motivation

Grazioso Salvare needed a tool to locate dogs in the Austin area that are suitable for search-and-rescue training. This dashboard was developed to streamline that selection process and reduce the time and effort required to identify qualified candidates.

## Getting Started

To get a local copy up and running:

1. Install all required libraries
2. Make sure MongoDB is accessible with your credentials
3. Run the IPYNB file in Jupyter

## Installation

You will need:

Python 3.8+

Jupyter Notebook

MongoDB instance + AAC dataset

The required libraries:

* 1. pymongo — to connect to the MongoDB database
  2. dash — for building the interactive web dashboard
  3. jupyter-dash — to run Dash apps in a Jupyter notebook
  4. dash\_leaflet — to render maps
  5. plotly — to create interactive charts

## Usage

Start by opening up the project in Jupyter Notebook, making sure that the IPYNB file is in the same directory as animal\_shelter.py. Run the program and follow the link to launch the dashboard. It should look something like this:

A screenshot of a computer

AI-generated content may be incorrect.

You can also use the filter buttons at the top of the data table to filter the results to only show dogs that are suitable for a specific type of training. Here’s an example showing the chart and map with the filter set to “Water Rescue”:

A screenshot of a computer

AI-generated content may be incorrect.

When you change the filter, the chart showing outcome types will automatically update. This allows you to see which proportion of the dogs are available for adoption, which have already been adopted, euthanized, etc. Example:

A screenshot of a map

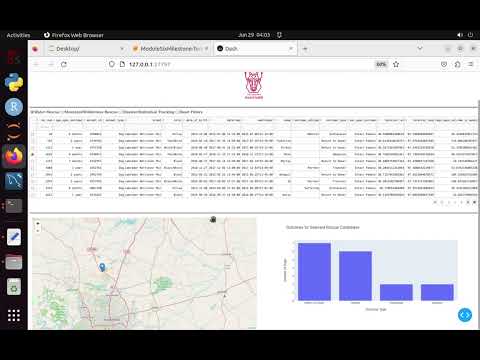
AI-generated content may be incorrect.

When selecting a new row, the map will update to show the location of the selected dog. Here’s another example demonstrating this. Also note that the bar chart shows that all animals that fit the filter have the outcome type “Return to Owner”. This can be useful for quickly identifying that there are no dogs available for training.

A screenshot of a map

AI-generated content may be incorrect.

A video showing the dashboard updating as filters or new rows are applied is also available if you’d like to see the dashboard in action before continuing:

[](https://www.youtube.com/embed/7io6SXR8h0M?feature=oembed)

**Tools Used:**

* MongoDB: Used as the database (model layer) to store animal shelter data. MongoDB is well-suited for this project because it handles semi-structured data and pairs seamlessly with Python through the pymongo driver. It allows for quick, which is helpful given the varied nature of animal records.
* Dash: Serves as the framework for both the view and controller components of the dashboard. Dash was useful for building an interactive apps like this one. Its use of callback functions makes it easy to link user interactions (like filters and selections) with dynamic updates to tables and charts.
* Dash Leaflet: A Dash extension that provides map-based visualizations. Used here to plot the geolocation of selected animals.

**Contact:**

Gregory Anderson