# Grazioso Salvare Dashboard README

# 

## About the Project/Project Title

In this project, Grazioso Salvare is an international rescue-animal training company that partners with a regional non-profit operating five Austin-area shelters. They receive real-time outcome data on shelter animal, including species, breed, age in weeks, outcome type, and geographic coordinates, and require a tool to pinpoint dogs suitable for different types of rescue training.

This dashboard builds on the AnimalShelter CRUD module. It leverages that Python package’s create, read, update, and delete methods to fetch and manipulate MongoDB data seamlessly within Dash callbacks.

The dashboard ingests this data from a MongoDB collection populated with the Austin Animal Center Outcomes dataset. End users can:

* Apply rescue-type filters to identify candidate dogs:  
  • Water Rescue  
  • Mountain/Wilderness Rescue   
  • Disaster/Individual Tracking   
  • Reset filters to view all records
* Browse results in an interactive data table with sorting
* Explore spatial distribution on a geolocation map, plotting each dog’s last known coordinates
* Analyze demographics via a secondary pie chart

All components update dynamically in response to user input, enabling the Grazioso Salvare team to quickly narrow down candidates for specialized training programs without writing custom database queries.

## Motivation

Grazioso Salvare needed a user‑friendly dashboard to reduce training time and minimize errors when identifying dogs that qualify for various rescue types. Building the dashboard in Dash allowed rapid development of interactive widgets, while MongoDB’s flexible schema accommodated semi‑structured outcome records.

## Getting Started

If you want to start working with both the AnimalShelter CRUD module and the interactive dashboard, you’ll need access to a MongoDB instance with the AAC database populated using the Austin Animal Center Outcomes CSV.

*Database and User Authentication*

This project assumes the MongoDB database is named AAC, and the relevant collection is named animals. Authentication uses a standard username and password approach, with the credentials and port configured directly in the AnimalShelter class.

The data source used is a CSV file containing animal outcomes. This file should be imported into MongoDB using the mongoimport tool:

mongoimport –username=”${MONGO\_USER}” –password=”${MONGO\_PASS}” –host=${MONGO\_HOST} –db AAC –collection animals –authenticationDatabase admin –type csv –headerline –drop /usr/local/datasets/aac\_shelter\_outcomes.csv

Once imported, your MongoDB instance will hold all outcome records needed for both modules.

*The Python Module (CRUD)*

The AnimalShelter class was written in a separate Python module to support reuse in future projects. It includes:

* Create(data) – Inserts a document and returns True if successful.
* Read(query) – Retrieves documents based on the given query and returns a list.
* Update(query, updated\_data) – Updates one or more documents that match the query with the values provided in updated\_data. Returns the number of documents updated.
* Delete(query) - Deletes documents from the collection that match the query. Returns the number of documents removed.

The database connection is created inside the class, so users don’t have to configure anything manually when importing and using the module.

*Dashboard*

The ProjectTwoDashboard.ipynb notebook builds on the CRUD module to deliver a live web interface. Key points:

1. **Imports & Setup**: Loads AnimalShelter, Dash components, Pandas, and Plotly.
2. **Data Table & Filters**: Radio buttons filter data via shelter.read(query); results populate a data table.
3. **Charts**: A geolocation map and a pie chart update dynamically.
4. **Branding**: Includes the Grazioso Salvare logo.

Open the notebook in Jupyter, run all cells, and the dashboard will start on your localhost.

## Installation

This project uses the following tools and libraries:

1. Python 3.9
   1. This project was built using Python 3.9. It was chosen for its wide support and compatibility with libraries like PyMongo. All development and testing were done in Python, and it’s used to create the backend class that handles MongoDB interactions.
2. MongoDB
   1. MongoDB is a NoSQL database used to store animal outcome data. It’s a good fit for this kind of semi-structured data where each document (record) might not follow the exact same format. For this project, MongoDB is hosted in the Apporto virtual lab, but you can use any local or cloud MongoDB instance.
3. PyMongo
   1. PyMongo is the official Python driver for MongoDB. It allows your Python scripts to connect to Mongo, run queries, and handle documents. It was chosen because it’s lightweight, reliable, and works well with object-oriented designs.
   2. To Install PyMongo run:
      1. pip install pymongo
4. Jupyter Notebook
   1. Jupyter was used to test the CRUD module interactively. It makes it easier to verify if each function is working correctly and to view results in real-time.
5. Dash
   1. Python framework for building interactive web dashboards for rapid development.
   2. To Install Dash:
      1. pip install dash dash-core-components dash-html-components dash-table
6. Plotly
   1. Charting library powering Dash graphs that provides rich interactivity and easy figure creation.
   2. To install Plotly:
      1. pip install plotly

## Usage

Default Reset:

*A screenshot of a computer

Description automatically generated*

*A screenshot of a map

Description automatically generated*

Disaster/Tracking:

*A screenshot of a computer

Description automatically generated*

Mountain/Wilderness

A screenshot of a map

Description automatically generated

Water Rescue:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Contact

Your name: Landon Phillips