# CS 340 Project Two ReadMe

Andrew Emilio DiStefano

SNHU

Dr. Reuben Wilson

October 20, 2024

# K9Recruit

K9Recruit facilitates the process of identifying the best canine candidates for search-and-rescue training. This powerful dashboard allows the user to filter a database of dogs by breed, age, and sex to show the optimal canine specimens for a variety of different types of rescue applications. The dashboard also allows the user to select a specific dog from the database and shows the geolocation of the animal on a map. An interactive pie chart at the bottom right of the dashboard shows the percentage of each breed of dog which is available for each rescue application.

MongoDB was used as the database component for the development of this project. The scalability, versatility, and ability to easily and efficiently manage large amounts of data made MongoDB the best option for the K9Recruit system. Non-relational database models such as MongoDB also creates far less rigidity than would a relational database in this scenario, making MongoDB an ideal choice for interaction with Python code.  
  
The Dash framework, which provides the view and controller structure for this web application, offers a multitude of data analytics tools which are essential for the navigation of such databases for the purposes specified by Grazioso Salvare. These include interactive data tables, geolocation maps, and different types of graphs and charts for effective data visualization.   
  
In order to complete this project, we started by researching the Dash framework and how the functions in our priorly developed CRUD middleware module could be used to view and modify data from the provided AAC database. We then read carefully through the project requirements and used the code from some of our previous projects - as well as some new code written specifically for this project - to meet and exceed the expectations of our client.   
  
We did run into some issues along the way, especially toward the end of development. After having set up all of the widgets, our first implementation of code resulted in a TypeError which rendered the user selection of radio items unable to affect the state of the data frame or the pie chart. After some research and review of the codebase, we realized that we were including the ‘\_id’ column of each document which contains objects of type ‘ObjectID’. Our solution to this error involved adding the following code after each query specification within each ‘if’ statement within the ‘update\_dashboard’ function:   
  
df.drop(columns=['\_id'],inplace=True)  
  
Having dropped the ID column, no object types contained within our database files threw the TypeError. After making this change, the dashboard was then able to meet the requirements of the client.   
  
The Dash framework website provides comprehensive documentation which is incredibly helpful in creating functional dashboards and interfaces. Links to some of these pages can be found below:  
  
<https://dash-leaflet-docs.onrender.com/>   
  
<https://plotly.com/python/pie-charts/>  
  
<https://dash.plotly.com/dash-core-components>   
  
<https://dash.plotly.com/datatable>

## Motivation

International rescue-animal training company Grazioso Salvare has asked us to develop this application to help them navigate the databases of their non-profit partners for the best possible canine candidates for search-and-rescue training.

## Getting Started

### 1) Import your database file

Import your data file into MongoDB (if it has not already been imported).

### 2) Add the CRUD.py file

Add the CRUD.py file to your directory of choice.

### 3) Open the MongoDB shell

Open mongosh by entering the mongosh command into your terminal as depicted below:

mongosh

### 4) Set your connection variables

Check your Dashboard file and make sure that your connection variables are correct including your username (USER), password (PASS), host address (HOST), port number (PORT), database (DB), and collection (COL).

If you are unsure of the appropriate host and port to use, this information will appear after you have opened the mongosh shell, as depicted below:



If this information does not match what is in your dashboard file, you will get an error.  
  
**5)** Navigate to the specified host and port within your browser and start using your K9Recruit dashboard!

## Installation

Install the latest versions of Python and MongoDB.

Instructions can be found at the following links:

### Install MongoDB on Linux:

<https://www.mongodb.com/docs/manual/administration/install-on-linux/>

### Install MongoBD on Windows

<https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/>

### Install MongoDB on Mac:

<https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-os-x/>

### Install Python on Linux:

<https://www.geeksforgeeks.org/how-to-install-python-on-linux/>

### Install Python on Windows:

<https://www.python.org/downloads/>

### Install Python on Mac:

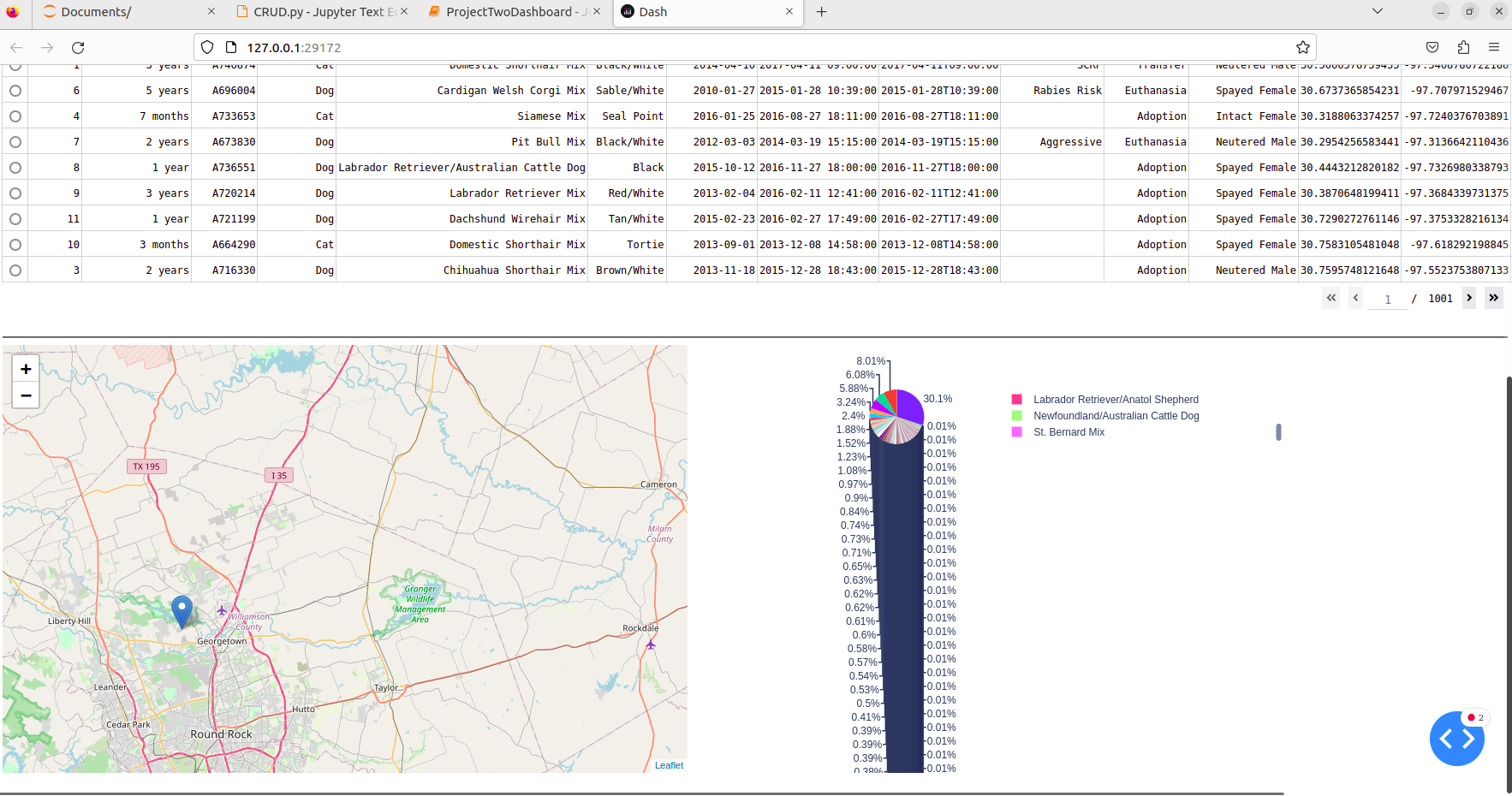
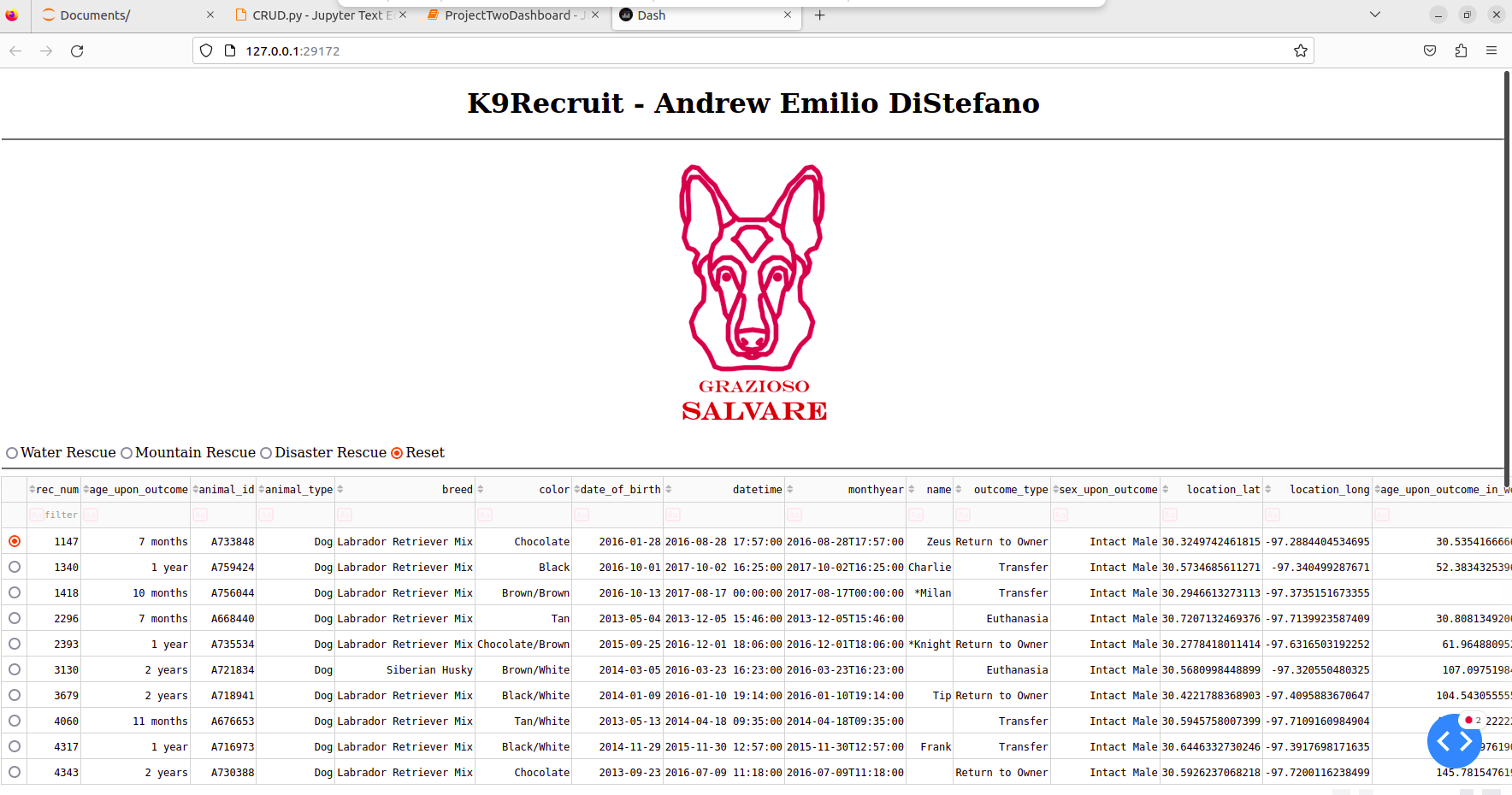
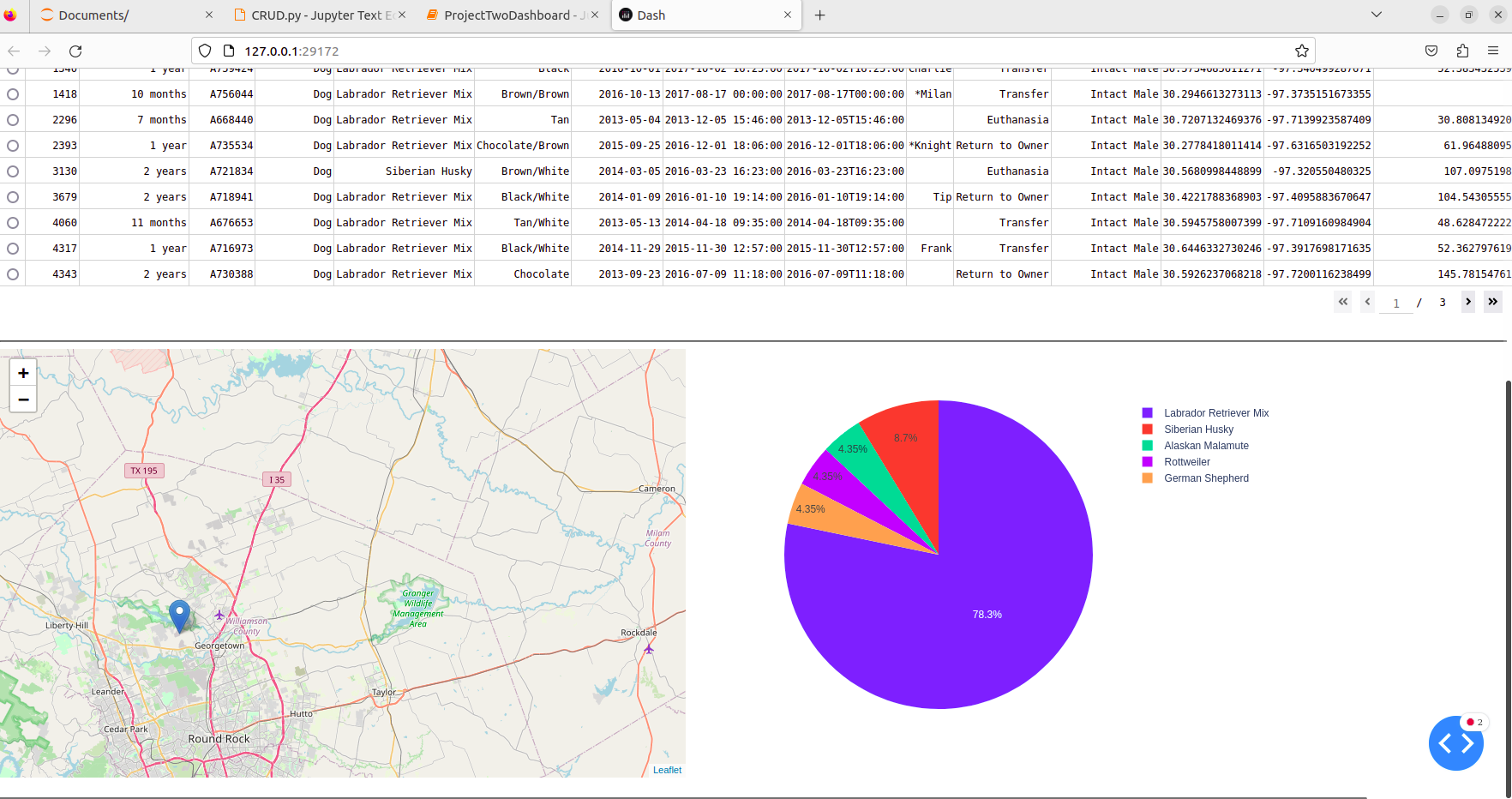
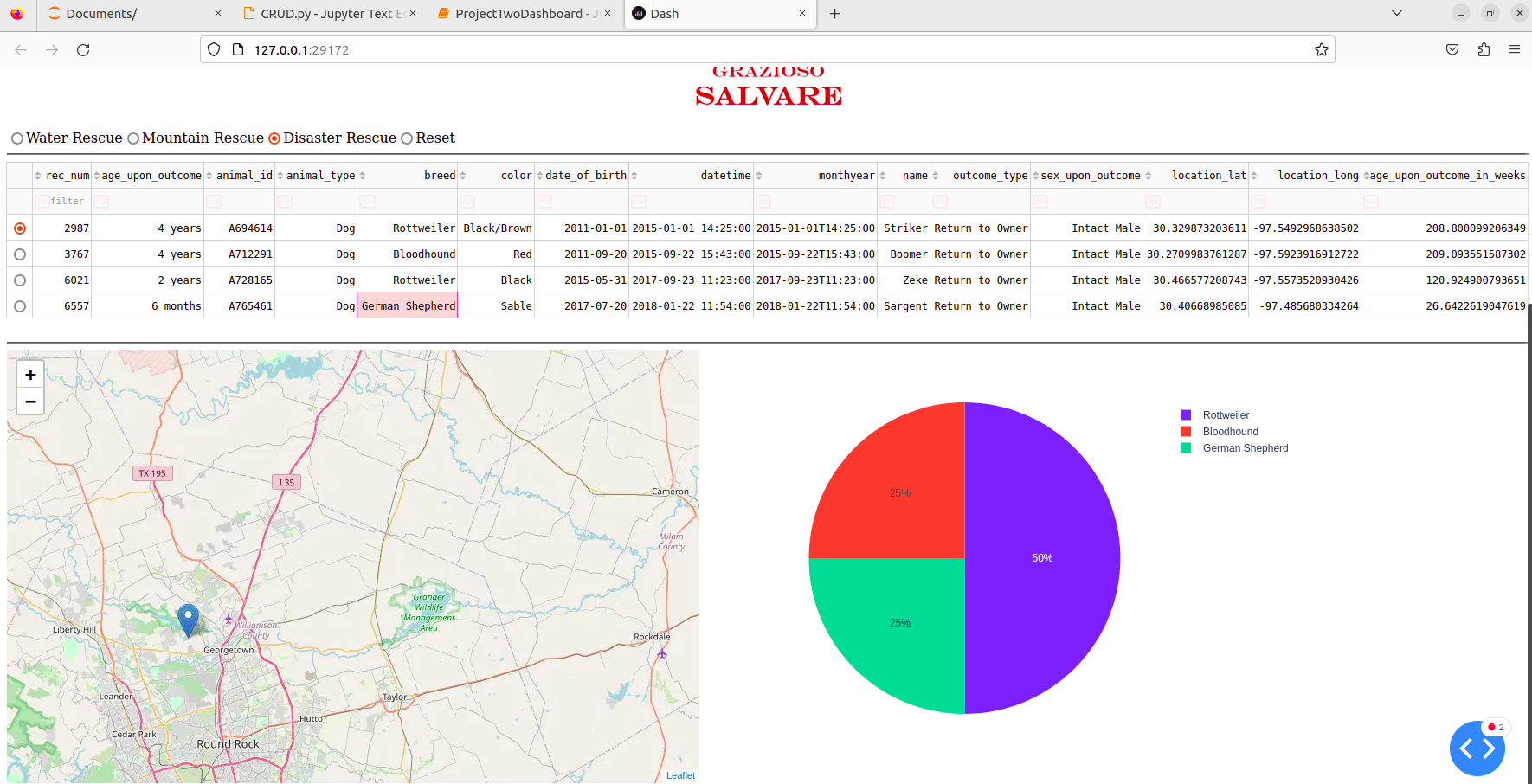
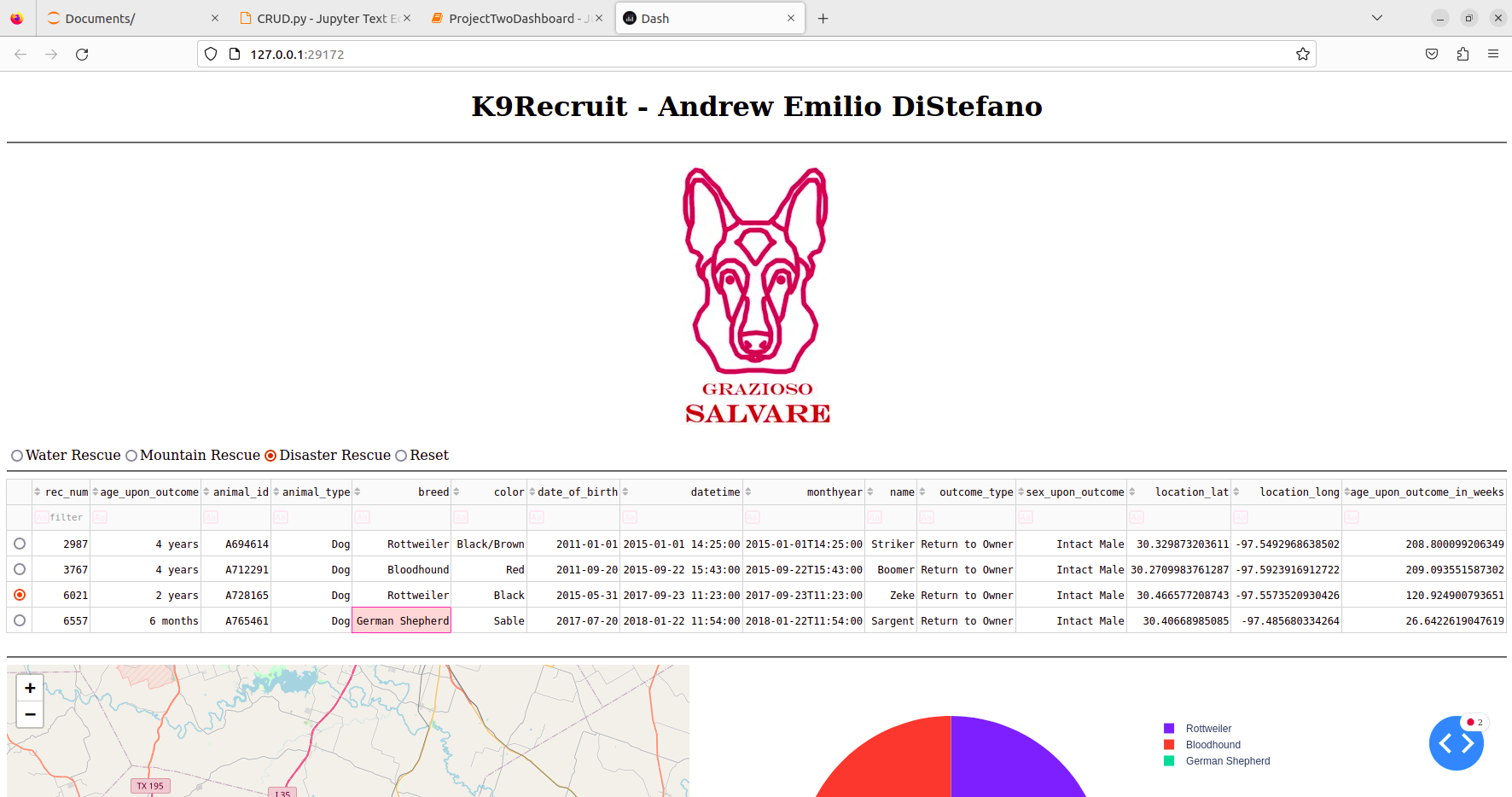
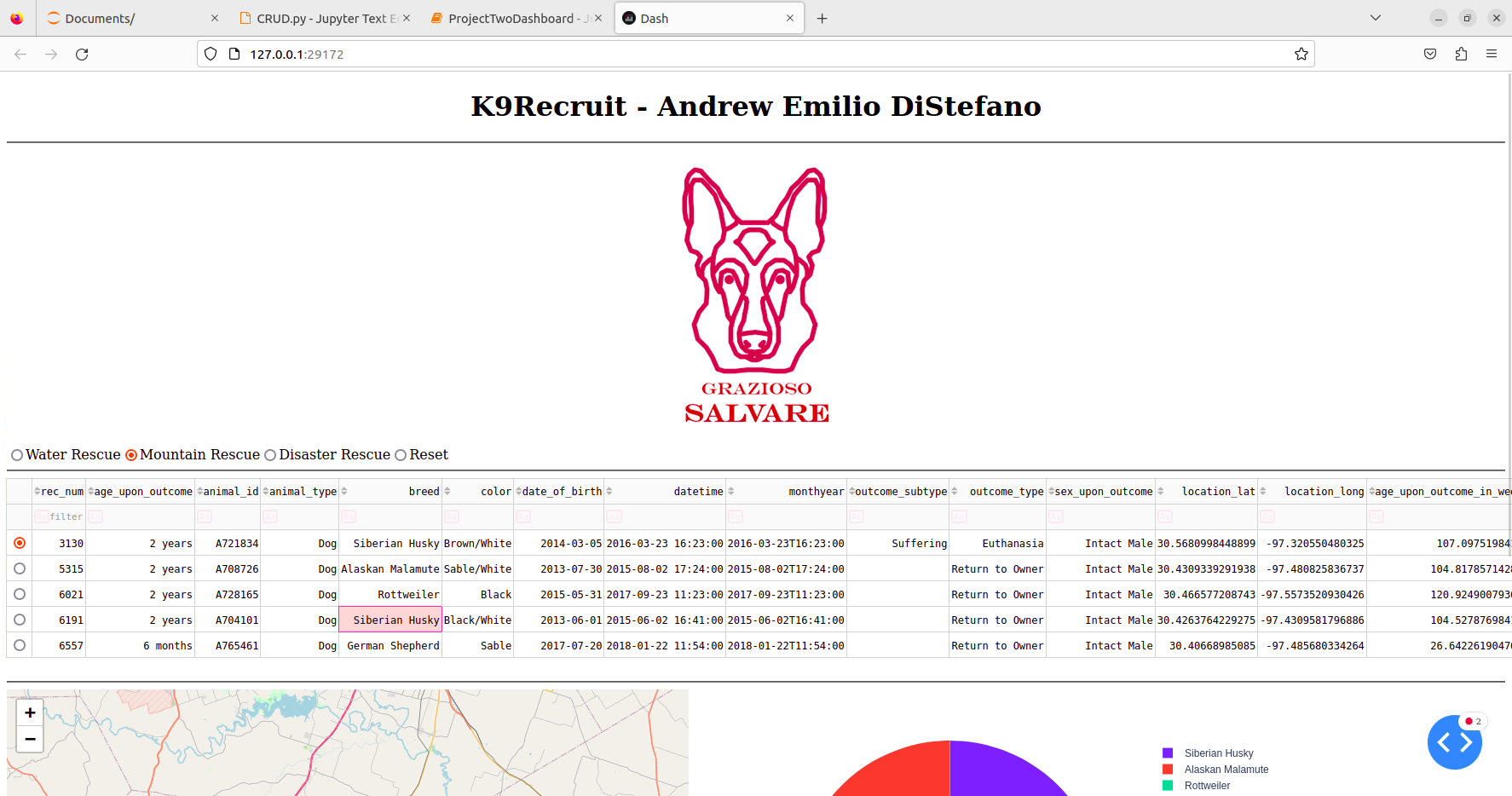
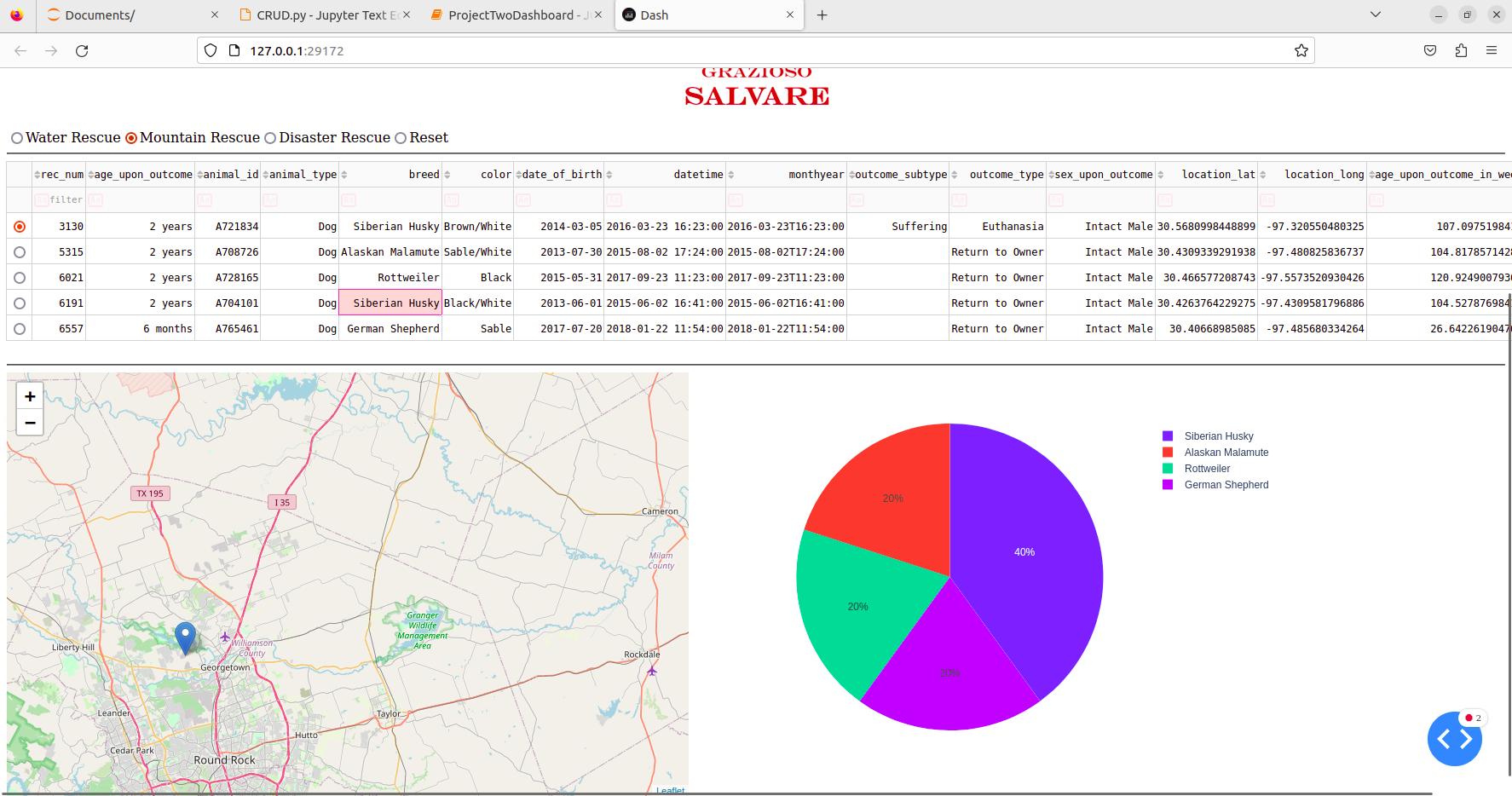
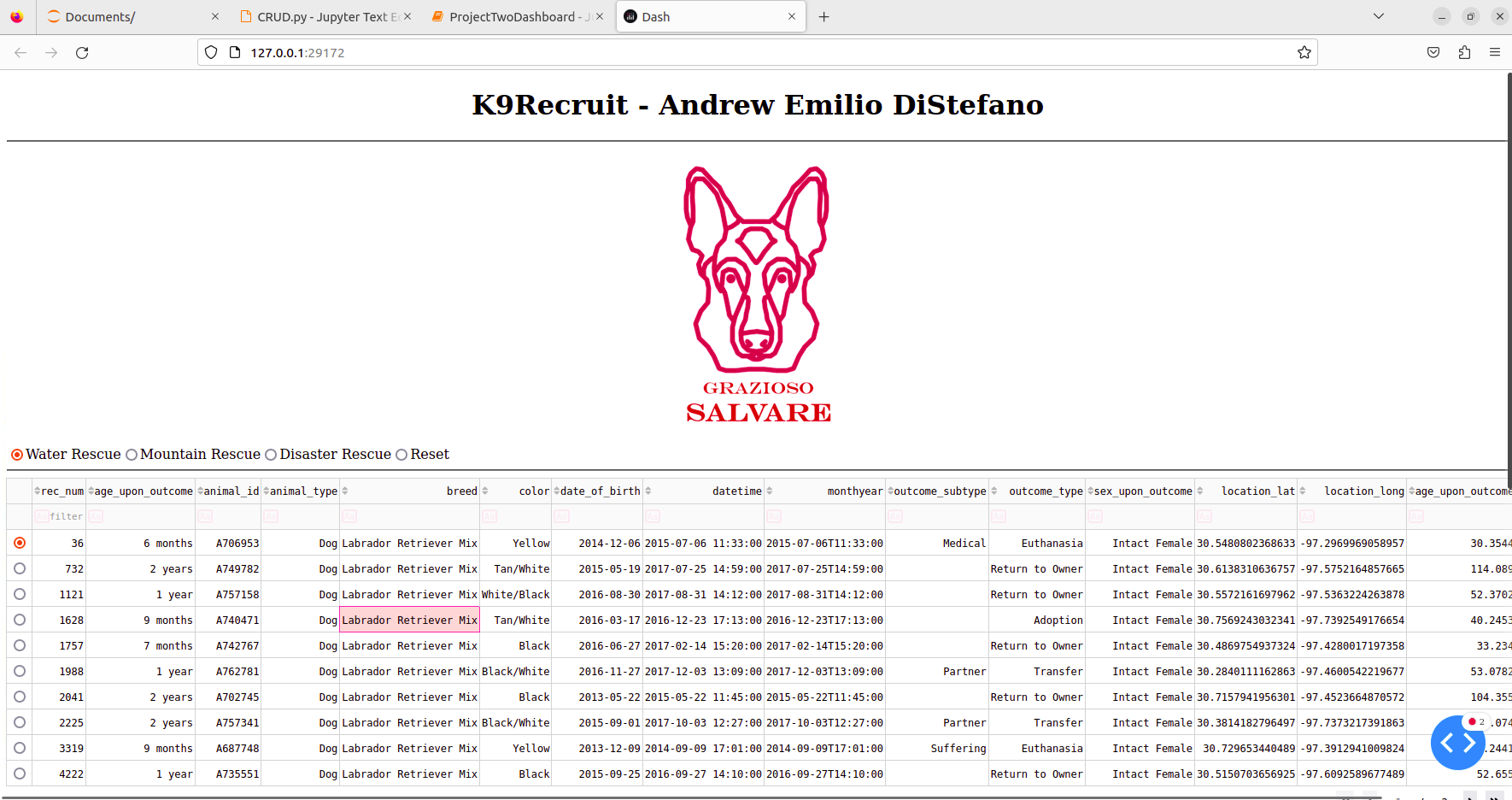
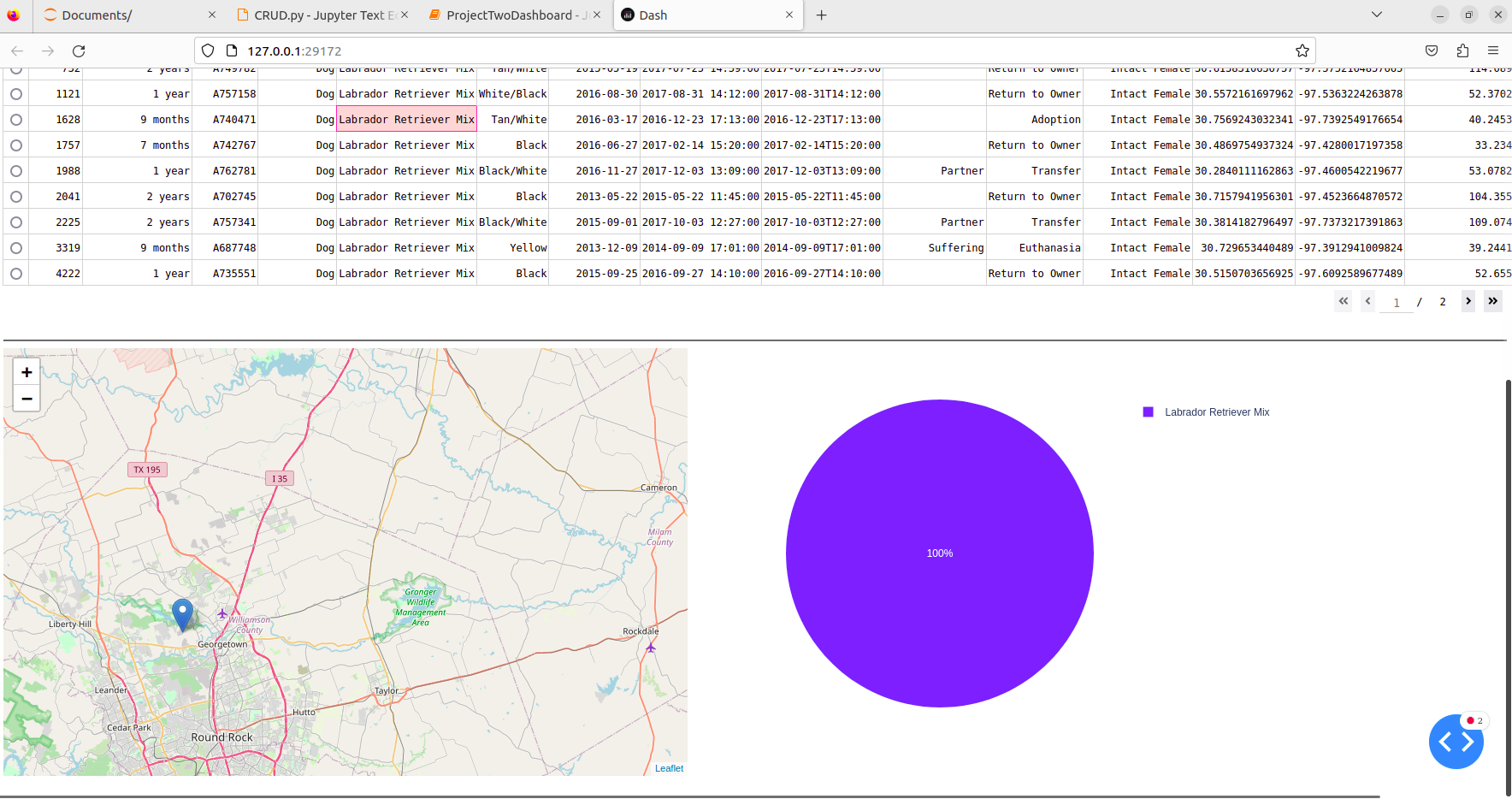
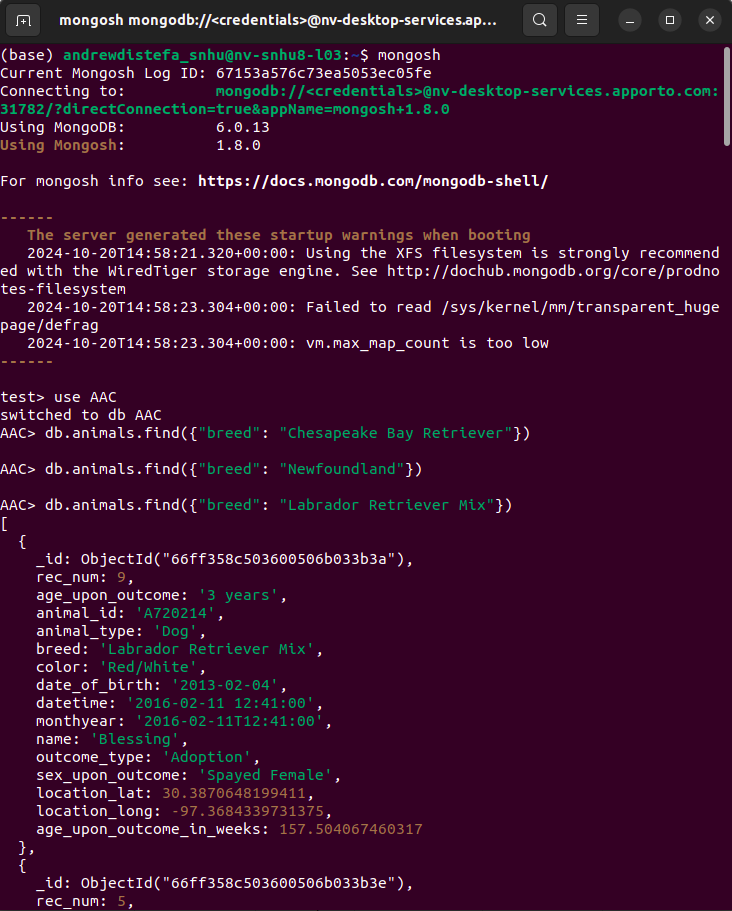
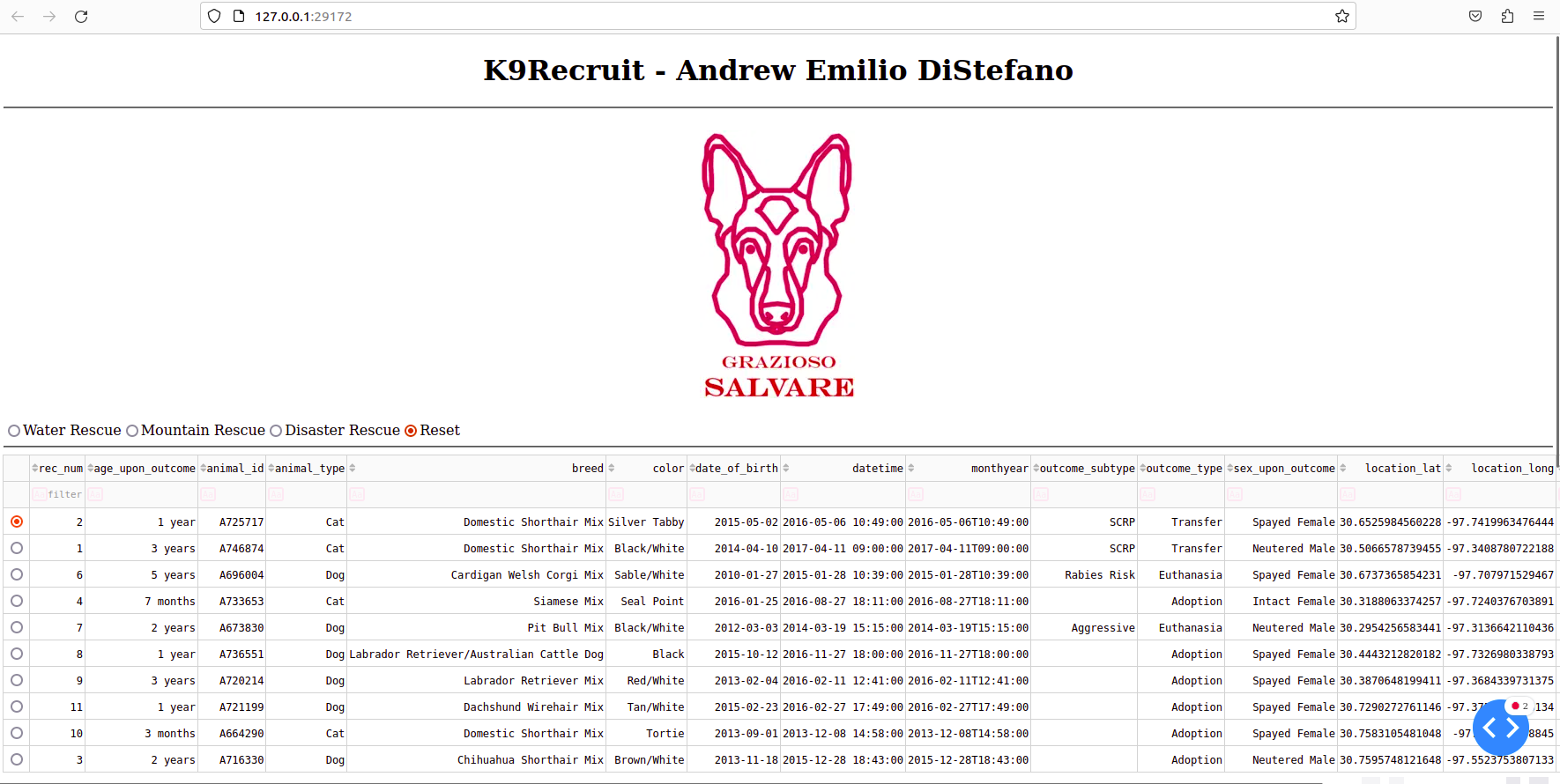
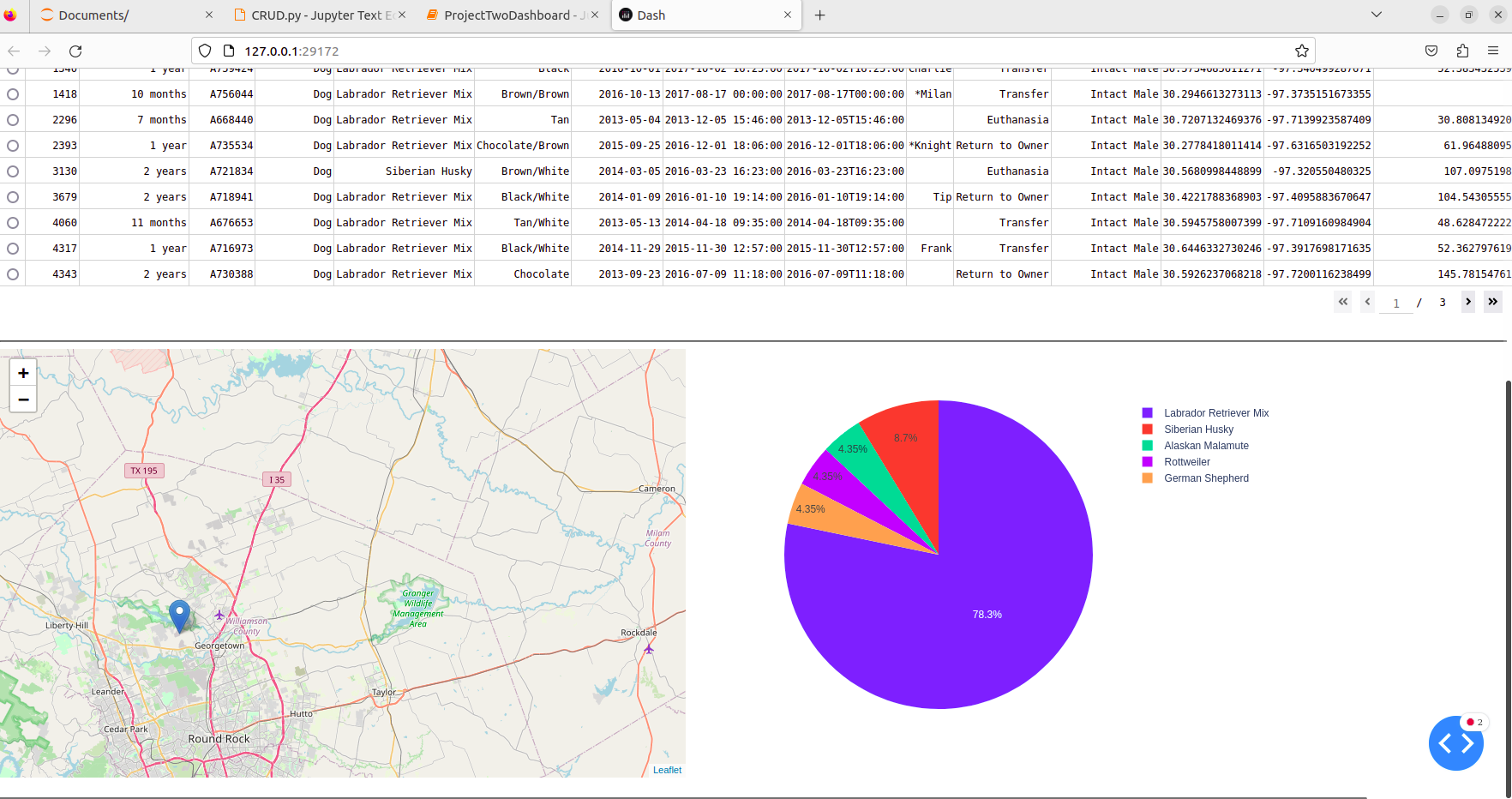
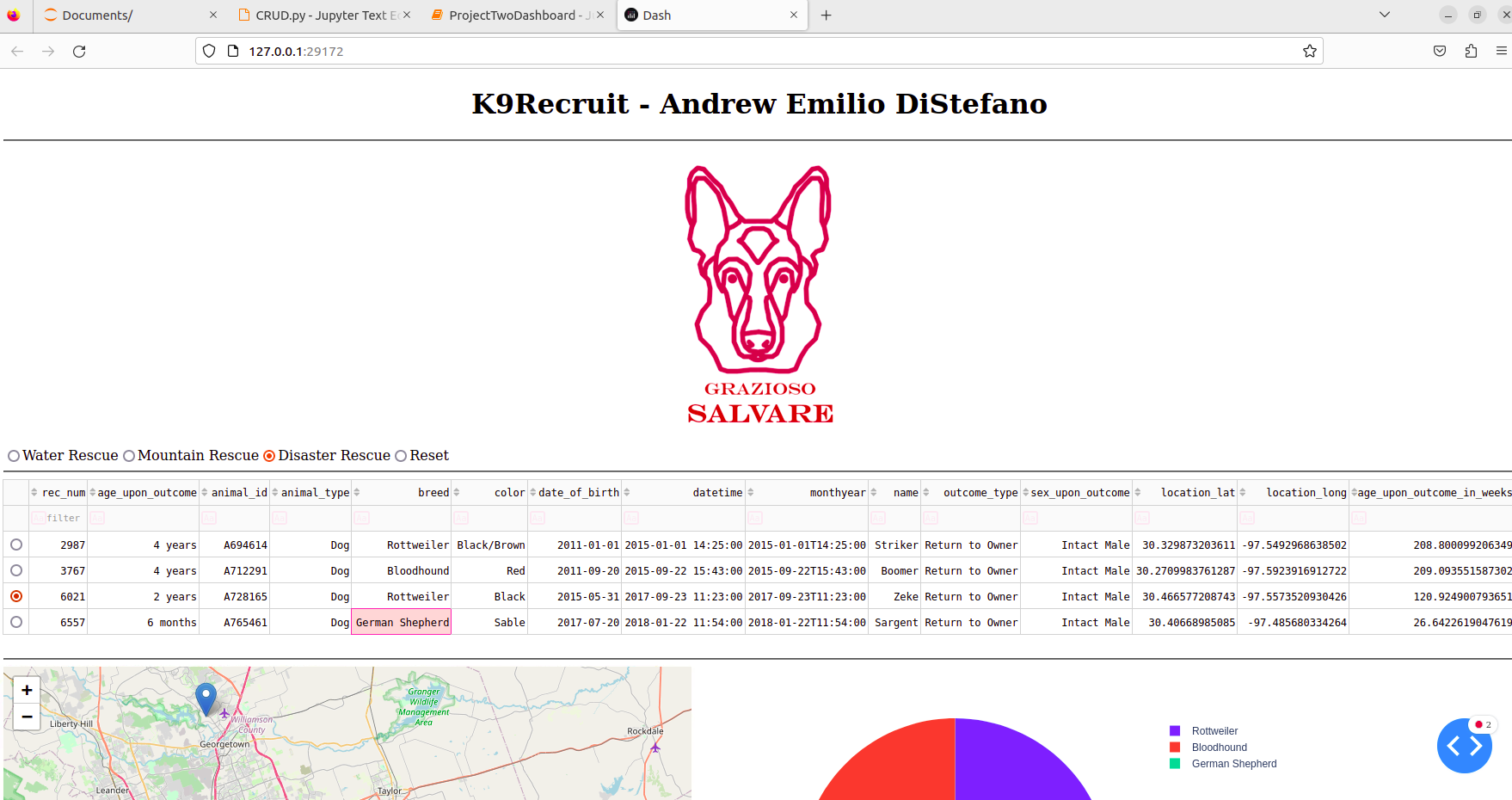
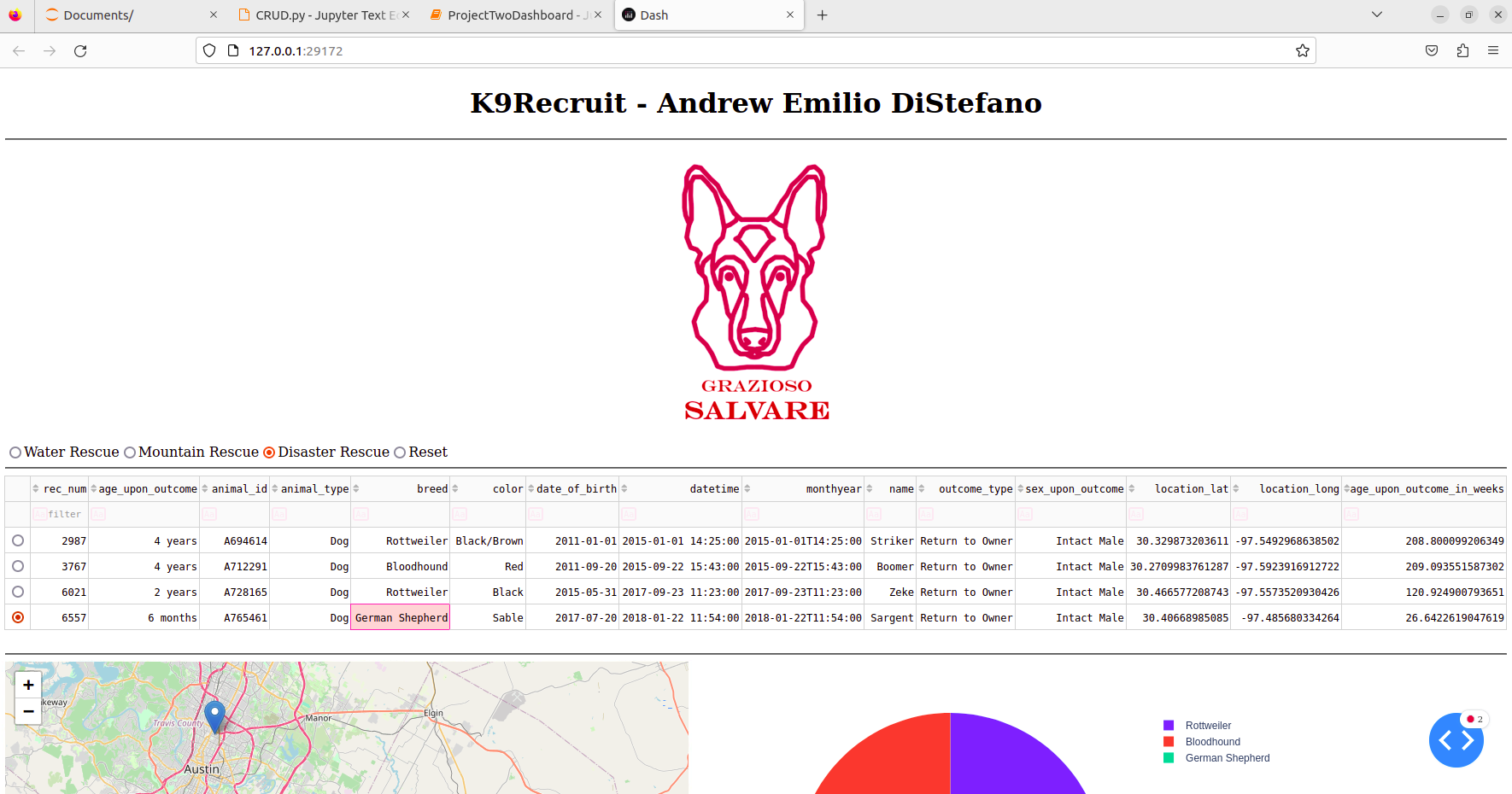
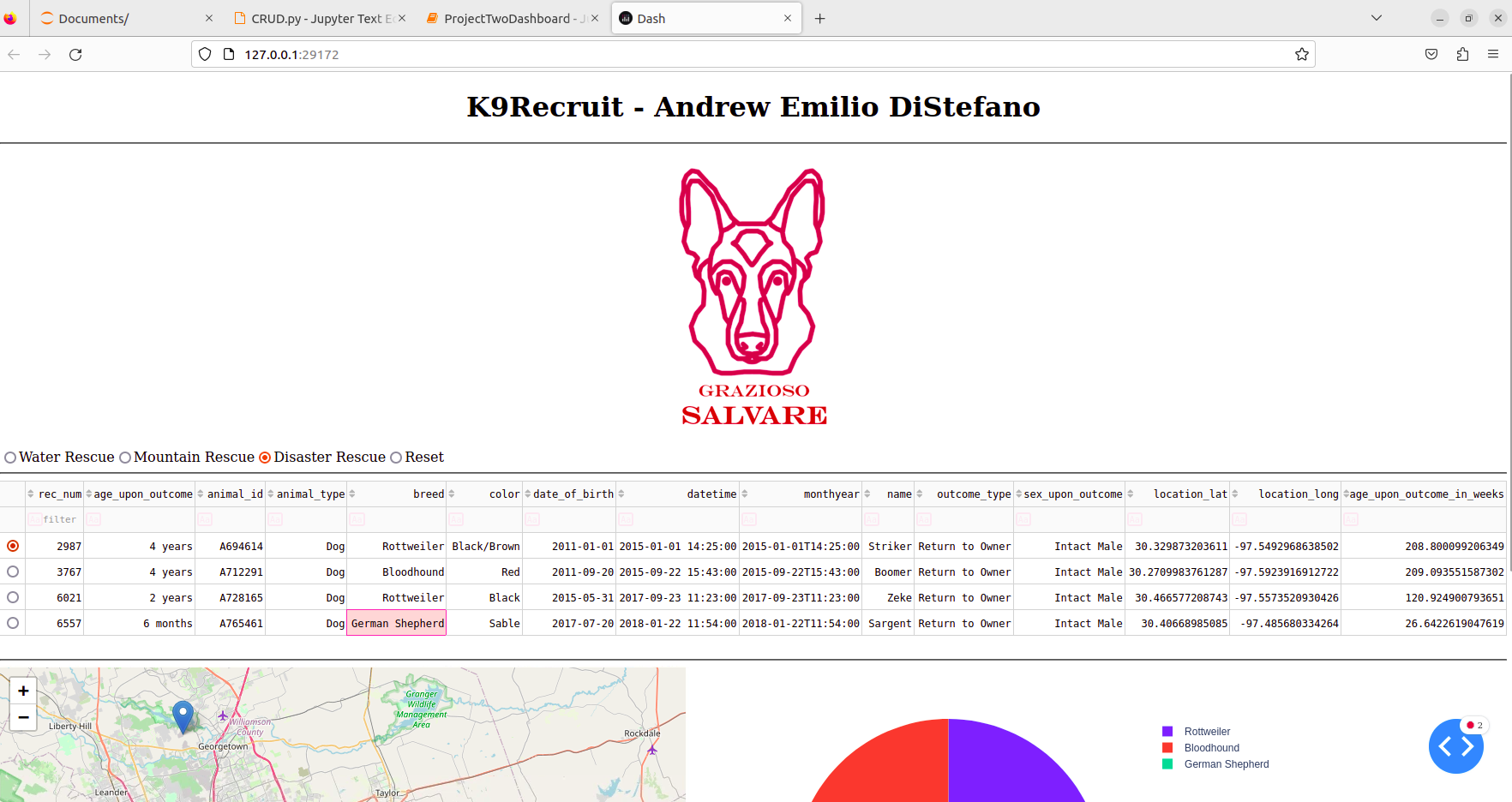
<https://www.python.org/downloads/macos/>

## Usage

In order to use this project to interact with MongoDB databases, our CRUD module uses the create(), read(), update() and delete() functions to create, delete, and update objects and to query the database.

### Code Examples

### The following code makes up the callback function and the query specification for the radio items which specify filtering for dogs of breeds which are optimal for ‘water rescue’ ‘mountain rescue’ and ‘disaster rescue’ training: As you can see, each of the ‘if’, ‘elif’ and ‘else’ statements contains a dictionary with the appropriate breeds and other characteristics specified for each rescue scenario mentioned in the Grazioso Salvare Dashboard Specification Document. The ‘reset’ radio item allows the data frame and pie chart to show every animal in the database as requested by our client, but uncommenting the commented-out code within the above ‘else’ statement will make it so that the dashboard only shows animals of breeds which would be of interest to Grazioso Salvare. Tests

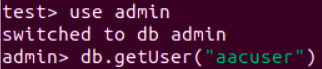
Upon entering the dashboard, the database has not yet been filtered. Two Images (one of the top of the screen and one of the bottom) of what the dashboard looks like while managing the unfiltered data are shown below:   
  
  
This is of course not optimal since Grazioso Salvare trains dogs rather than cats, and since many of the dogs in the database are of breeds which are not optimal for search-and-rescue operations. The pie chart is also rendered virtually unreadable if the entire database is completely unfiltered due to the sheer number of different types of animals stored in the database.   
  
Rather than including all of these animals in the unfiltered dashboard, it would make more sense to filter by default for all of the dog breeds which would be viable for search-and-rescue operations. Two screenshots depicting the starting state of the dashboard with these filters implemented by default are shown below:   
  
   
This is a much more readable format. Since animals other than dogs and dog breeds which are not viable for search-and-rescue operations do not appear in the datatable or pie chart, both are much more readable and focused on the goals of the Grazioso Salvare company.   
  
Upon selecting the ‘Disaster Rescue’ radio item, the dashboard will appear as follows (with the current state of the AAC database):   
   
The fourth dog’s breed has been highlighted to showcase that capability of the interface. Upon clicking on a specific cell of the dataframe, that cell becomes highlighted in red.  
  
Upon selecting the ‘Mountain Rescue’ radio item, the dashboard will appear as follows (with the current state of the AAC database):   
  
   
  
Upon selecting the ‘Water Rescue’ radio item, the dashboard will appear as follows (with the current state of the AAC database):   
  
  
As you can see, all of the dogs which were found with the given query criteria belong to the breed 'Labrador Retriever Mix’. In order to confirm that no dogs of the other two breeds included in the search criteria exist in the database (and therefore confirm that this result was not due to error, we queried the database from within the MongoDB shell (mongosh) to search for the breeds specified in the Grazioso Salvare Dashboard Specification Document.   
  
As you can see, no animals of the breed ‘Chesapeake Bay Retriever’ or ‘Newfoundland’ exist in the database. The third query for gods of the ‘Labrador Retriever Mix’ breed yielded many results, confirming that we are successfully connected to the database.   
  
Upon selecting the ‘Reset’ radio item, the dashboard reverts back to its starting state and appears as follows (with the current state of the AAC database):   
  
  
  
The geolocation of an animal which has been selected will also appear in the geolocation chart based on the latitude/longitude coordinates of each animal in the database as shown in the screenshots below:   
  
  
  
  
The functions of this application will surely save our client Grazioso Salvare time and resources while searching for the best canine search-and-rescue candidates among the animals kept by their animal shelter partners and associates.

## Common Problems and Solutions

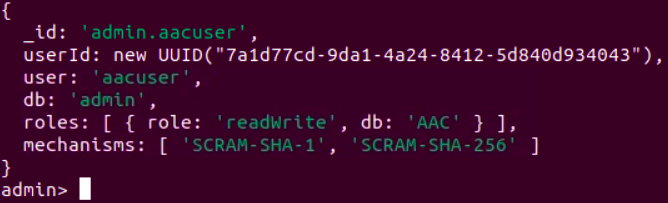
### Authentication Errors

Always make sure that your user has the appropriate privileges to avoid authentication errors.

Enter your mongosh shell and confirm that your user has both readWrite privilege in the database which you will be using by entering the following command as depicted below:



You should see output similar to the following:



In our example, the username is ‘aacuser’ and ‘AAC’ is the name of our database (db) of choice.

If your user's roles do not include the correct permission(‘readWrite’) and database name, change the user’s permissions and authorization or create a new user with the appropriate permissions and authorization for use in the database in which you would like to perform CRUD operations with this library.

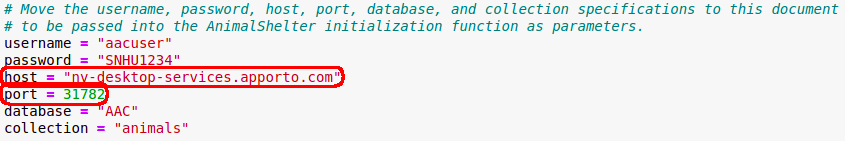
### ServerSelectionTimeoutError

Always make sure that your connection variables are assigned the correct values including your username, password, host address, port number, database, and collection. These variables are declared in the Dashboard file rather than in the CRUD middleware file.

If you are unsure of the appropriate host and port to use, open mongosh in your terminal and look for the “Connect to:” section toward the top of the output. The host and port number for this example are shown in the image below.



The values circled in red above should be assigned as the values of the variables circled in red as depicted below:



Your host and port will almost certainly be different from those in the example, so be sure to add the appropriate host and port as the values of your HOST and PORT variables.

## Roadmap

We are currently working on adding more functionality and widgets to this dashboard for more effective and comprehensive querying and data analytics. Follow us on social media and visit our GitHub to see if any new versions or functions have been added to the project.

**Contact:**

Emilio DiStefano