# Client - Server Development Project

## About the Project/Project Title

Developing a software application to help Grazioso Salvare identify and categorize dogs from shelter data for search-and-rescue training, featuring a user-friendly web dashboard.

## Motivation

The project helps Grazioso Salvare, a search-and-rescue group, find the right dogs for training. They need a simple software tool to sort dogs by age and breed, making it easier to find the best ones for specific rescue missions.

## Getting Started

In this project, our first step will be importing our raw data file into MongoDB, specifically the AAC CSV file. Once we have successfully inserted the file into the MongoDB shell, our next task is to create a CRUD (Create, Read, Update, Delete) file that interacts with the database within the MongoDB shell. Finally, we will connect the CRUD file to Dash Plotly using Jupyter, allowing the client to easily interact with the data.

## Installation

The Linux OS, available for download at https://www.linux.org/pages/download/, will serve as the operating system for running the system.

The MongoDB Python Driver (PyMongo), installed via >>pip3 install pymongo, facilitates interaction between Python applications and MongoDB databases.

MongoDB, accessible at https://docs.mongodb.com/manual/installation/, is a popular open-source NoSQL (non-relational) database management system (DBMS) used to store the database for this project.

The text editor gedit, available at https://gedit.en.softonic.com/, is used to create the CRUD file.

For an interactive computing environment, Jupyter can be installed via >>pip3 install jupyterlab. Jupyter serves as the communication hub between the database, CRUD function files, and the client-side interface (Dash Plotly).

Dash and Dash Plotly can be installed via >>pip install dash. These frameworks and component libraries enable developers to create interactive data dashboards.

Leaflet, installed via >>pip install dash-leaflet, is a library that facilitates the integration of interactive maps into Dash web applications.

Reference site: https://plotly.com/python/

## Usage

MongoDB was picked for its ability to handle different types of data effectively, like the information about animals in a shelter. With Python, using tools like PyMongo, it's easy to work with MongoDB, making data operations smooth and efficient.

### Code Example

#CRUD FILE SNIPPET

*from pymongo import MongoClient*

*class AnimalShelter:*

*def \_\_init\_\_(self, USER, PASS):*

*if USER == 'aacuser' and PASS == 'SNHU1234':*

*self.client = MongoClient('mongodb://aacuser:SNHU1234@nv-desktop-services.apporto.com:32014')*

*self.database = self.client['AAC']*

*self.collection = self.database['animals']*

*else:*

*raise ValueError("Invalid username or password")*

*#Create function.*

*def create(...)*

*def read(self, searchdata=None):*

*if searchdata:*

*return list(self.collection.find(searchdata))*

*else:*

*return list(self.collection.find())*

*#Update function*

*def update (...)*

*#Delete function*

*def delete(...)*

*#DASH APP FILE SNIPPET*

*#imports all libraries and files*

*username = "aacuser"*

*password = "SNHU1234"*

*shelter = AnimalShelter(username, password) #create shelter instance*

*df = pd.DataFrame.from\_records(shelter.read({}))*

*df.drop(columns=['\_id'], inplace=True)*

*app = JupyterDash(\_\_name\_\_)*

*#process image for logo and embed URL link*

*#Layout*

*html.Div(id='hidden-div', style={'display':'none'}),*

*image\_with\_link,*

*html.Div([*

*dcc.Dropdown(*

*id='rescue-dropdown',*

*options=[{...}], ) ]),*

*dash\_table.DataTable(*

*id='datatable-id',*

*columns=[{"name": i, "id": i, "deletable": False, "selectable": True} for i in df.columns],*

*data=df.to\_dict('records'), ),*

*#Layout for maps and chart*

*#call back for dropdown*

*#callback for piechart*

*#callback for table*

*#callback for geolocation using Leafly*

*app.run\_server()*

### Tests

#TEST FILE SNIPPET

*# To ensure the CRUD file works properly the test script created.*

*import the CRUD file*

*def main():*

*animalList = AnimalShelter('aacuser', 'SNHU1234')*

*data = { }*

*newdata = {}*

*searchdata = {}*

*deteledata = {}*

*if animalList.create(data):*

*print ()*

*else:*

*print ("")*

*search = list(animalList.read( searchdata))*

*print (search)*

*update = animalList.update(newdata)*

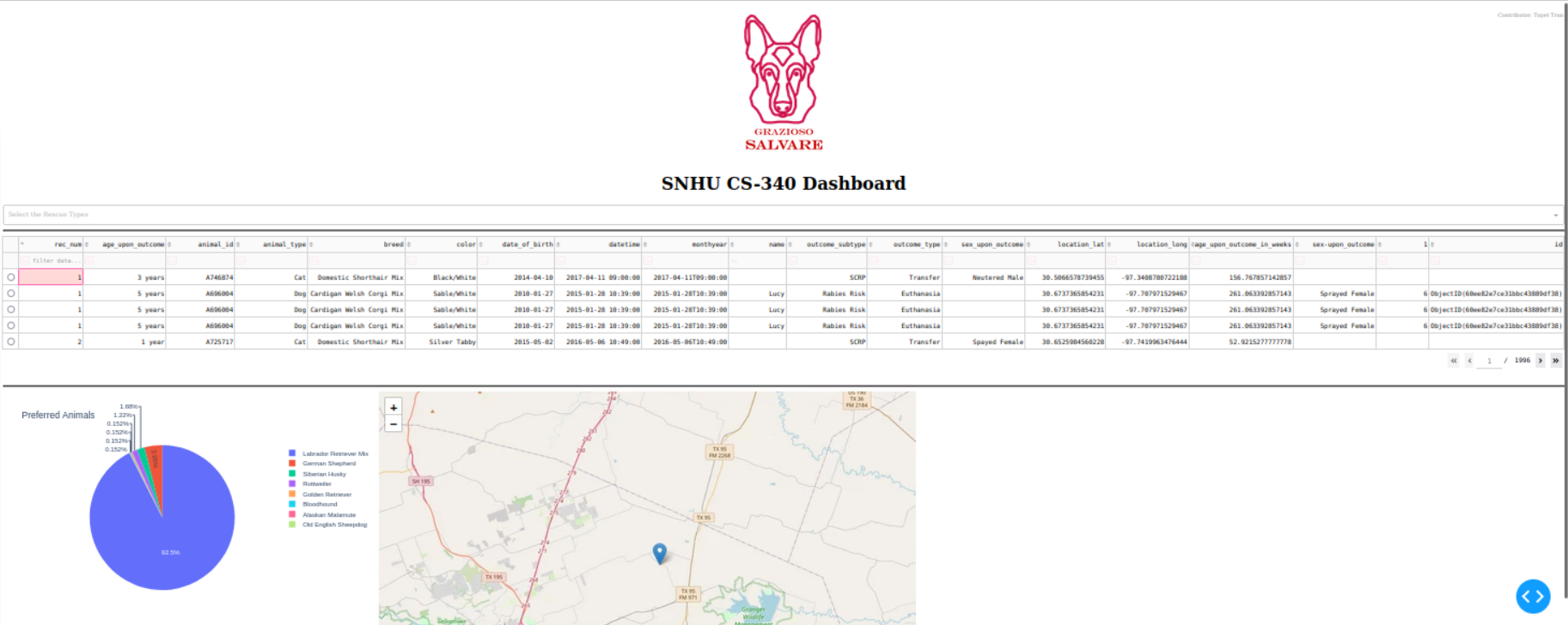
*print (update)*

*delete = animalList.remove()*

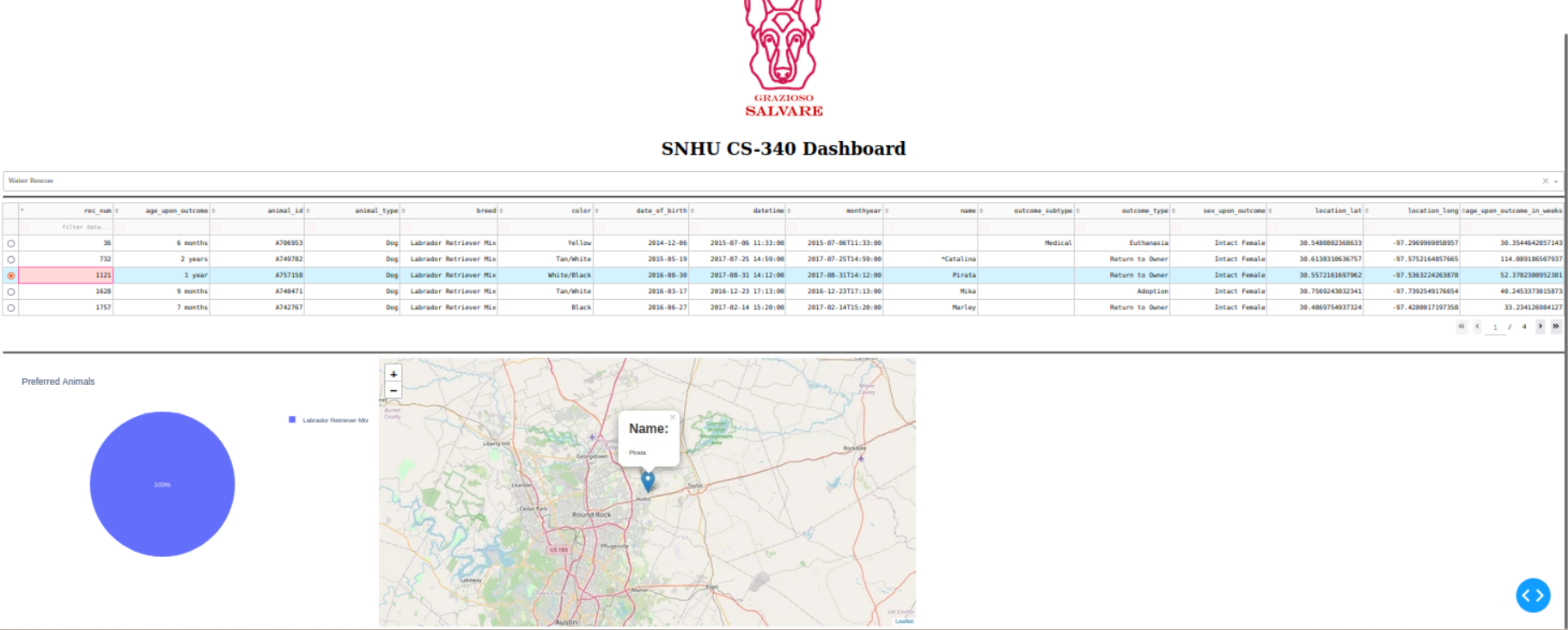
*print (delete)*

### Screenshots

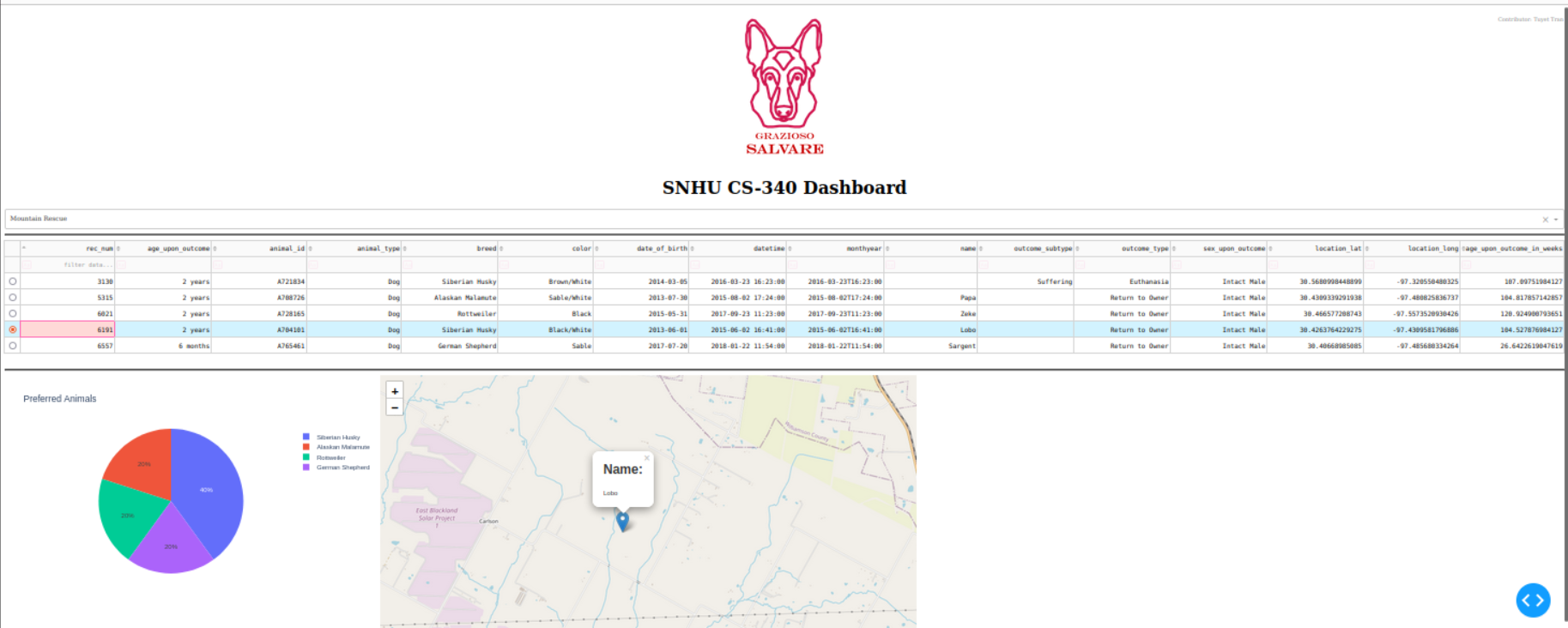
Starting state of dashboard.



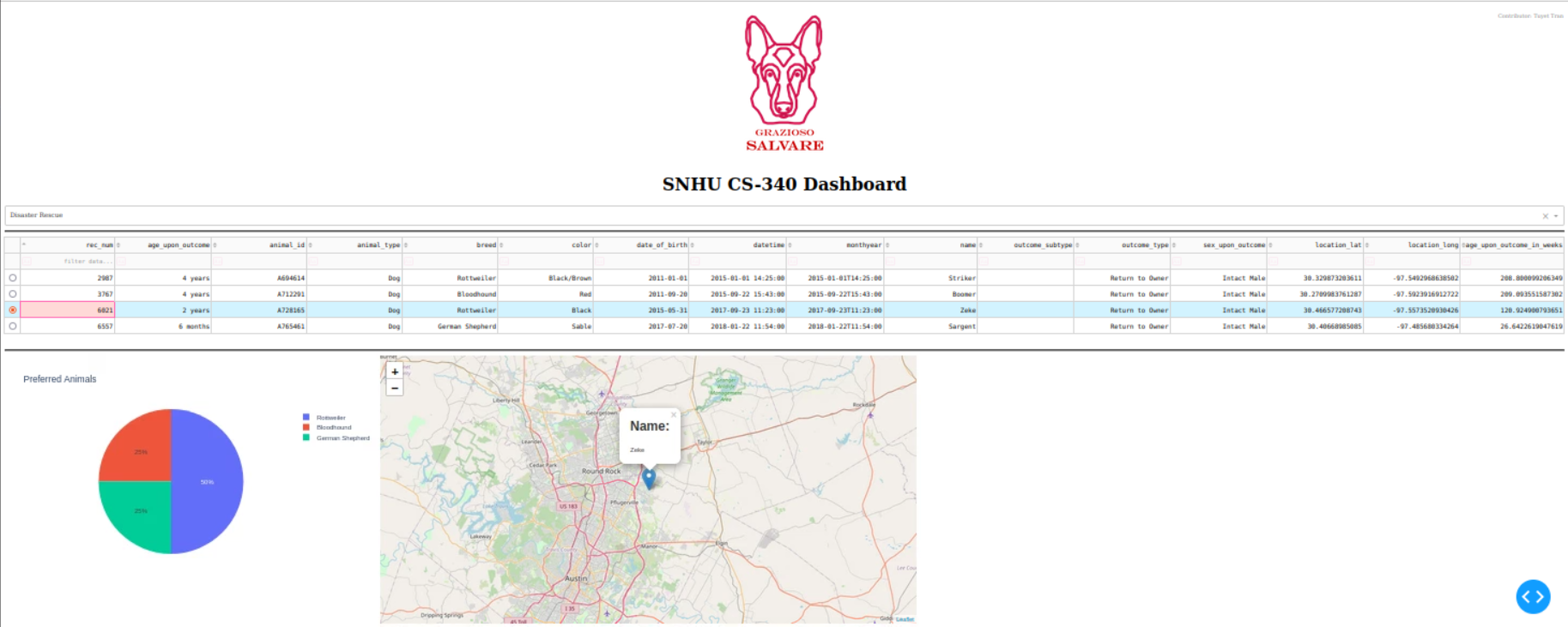
Water rescue drop-down dashboard.



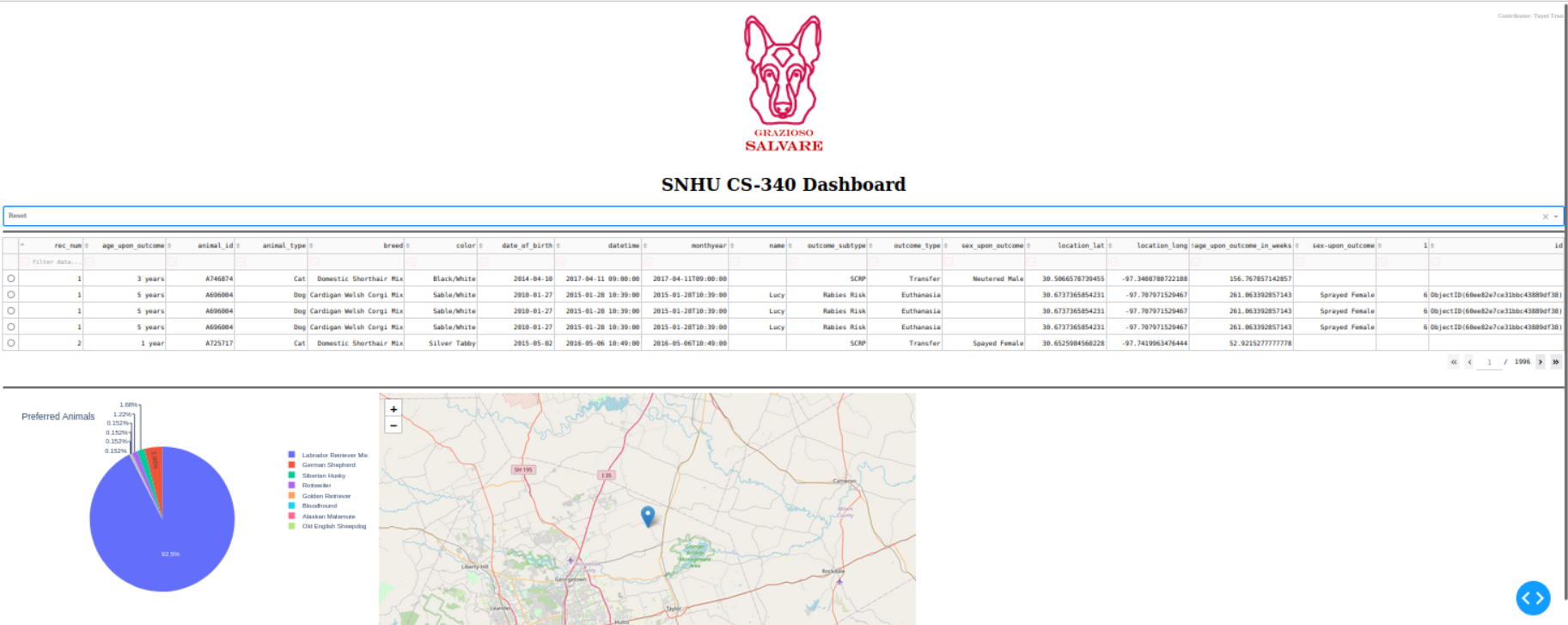
Mountain rescue drop-down dashboard.



Disaster Rescue drop-down dashboard.

**

Reset Functionality Dashboard



## Roadmap/Features:

The user interface enables users to select dropdown options with different rescue types. Upon selecting a rescue type, the dropdown allows users to filter dog breeds suitable for that rescue type. This action triggers changes in the pie chart to reflect the distribution of animal breeds and updates the table to display filtered data according to the client's requirements. Additionally, selecting a Radio Item activates geolocation, displaying the animal's location on the map.

## Challenges:

During the development process, I encountered challenges in understanding Leaflet and implementing it into the system. To overcome this hurdle, I thoroughly reviewed the code and researched to ensure that Leaflet would generate the correct input.

## By:

Tuyet Tran