CS-340 Project Two: Dashboard Submission

Name: Stephen Zawatski

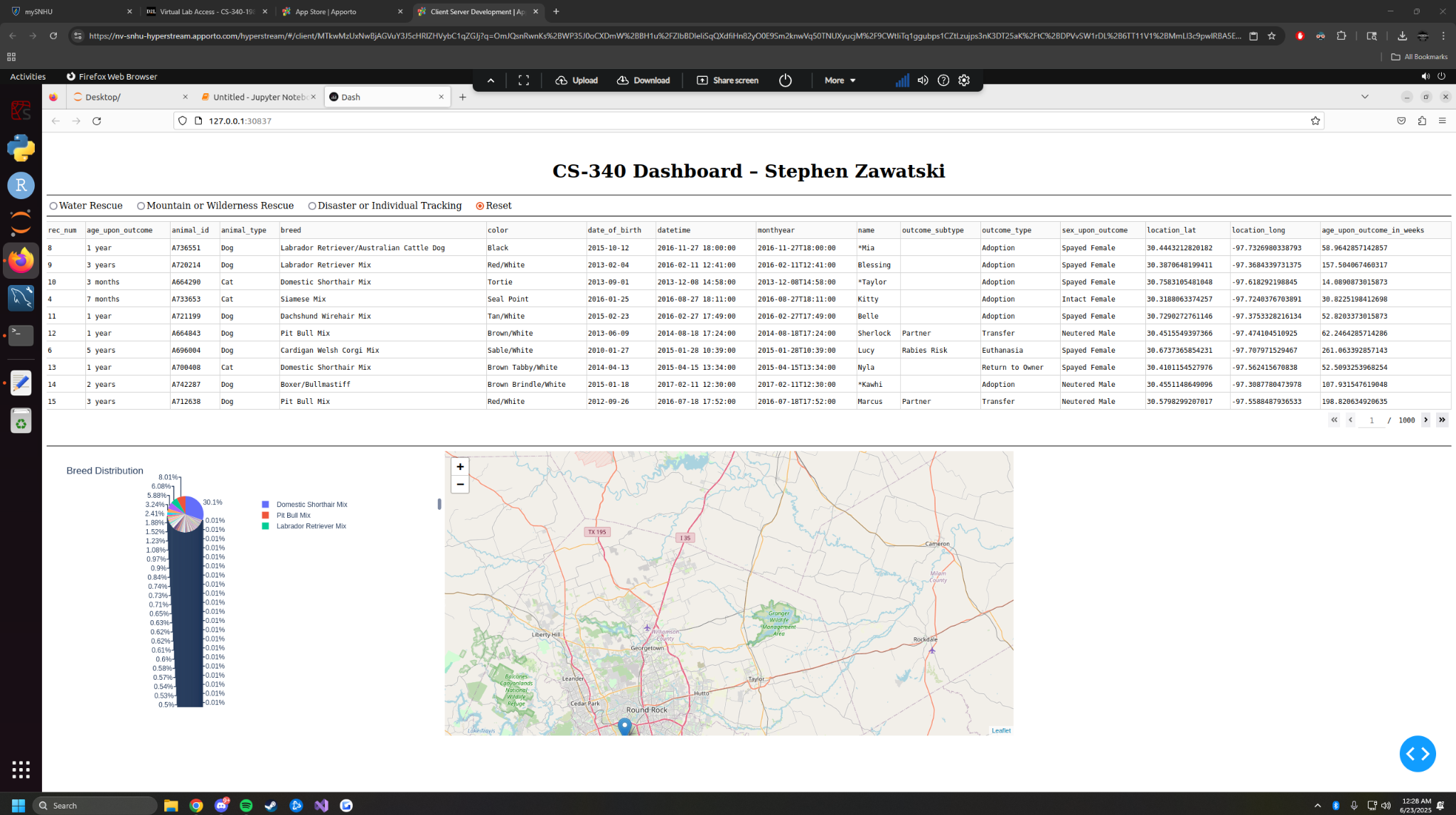
Course: CS-340  
Project: Module Seven – Project Two

# Final Dashboard Code

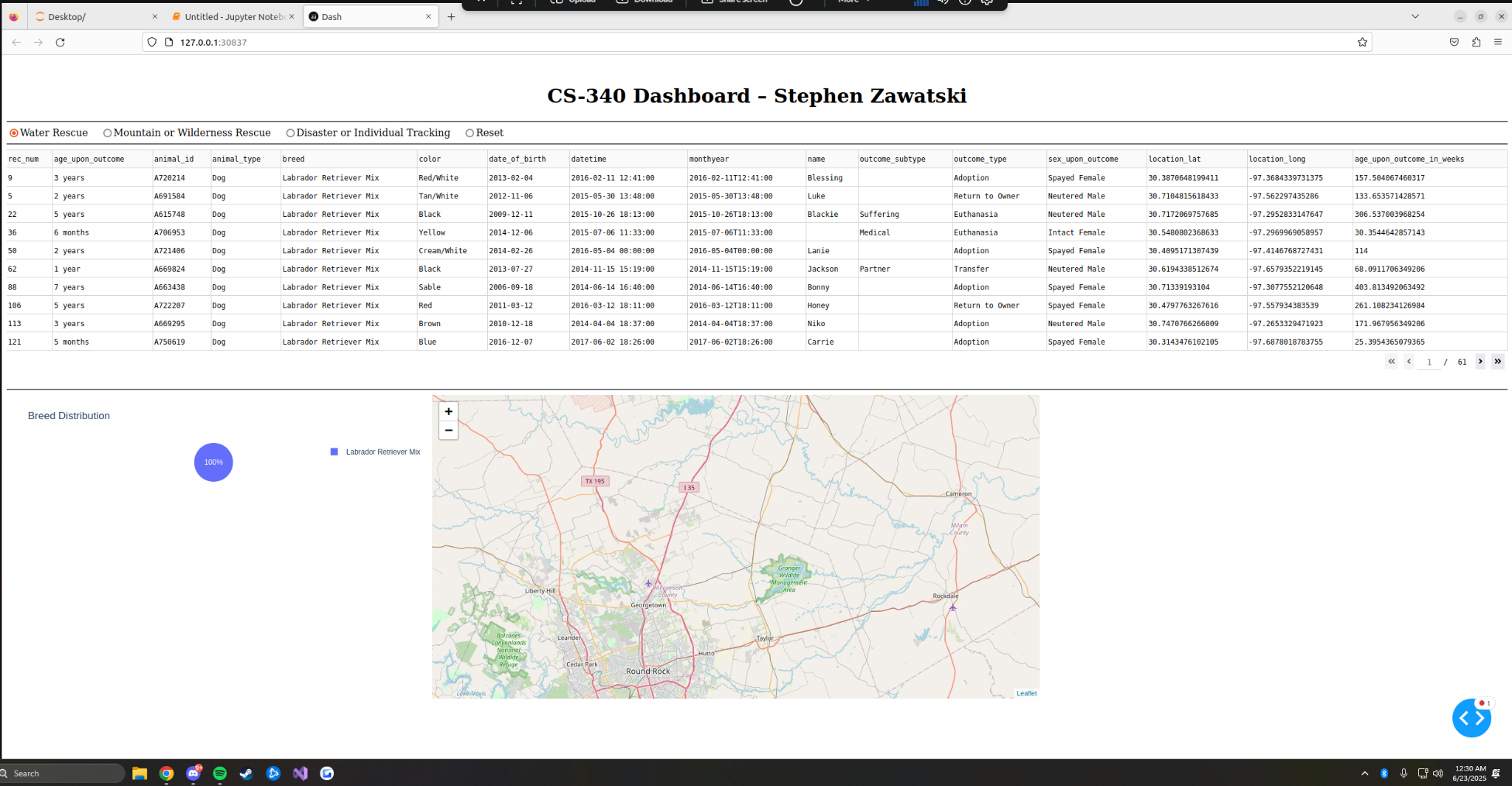
# Setup the Jupyter version of Dash  
from jupyter\_dash import JupyterDash  
import dash\_leaflet as dl  
from dash import dcc, html, dash\_table  
from dash.dependencies import Input, Output, State  
import plotly.express as px  
import base64, os, numpy as np, pandas as pd  
  
# Import CRUD module  
from animal\_shelter import AnimalShelter  
  
# Connect to MongoDB  
username = "aacuser"  
password = "SNHU1234"  
db = AnimalShelter(username, password)  
df = pd.DataFrame.from\_records(db.read({}))  
df.drop(columns=['\_id'], inplace=True)  
  
# Create app  
app = JupyterDash(\_\_name\_\_)  
  
# Layout  
app.layout = html.Div([  
 html.Center([  
 html.H1('CS-340 Dashboard – Stephen Zawatski')  
 ]),  
 html.Hr(),  
 dcc.RadioItems(  
 id='filter-type',  
 options=[  
 {'label': 'Water Rescue', 'value': 'water'},  
 {'label': 'Mountain or Wilderness Rescue', 'value': 'mountain'},  
 {'label': 'Disaster or Individual Tracking', 'value': 'disaster'},  
 {'label': 'Reset', 'value': 'reset'}  
 ],  
 value='reset',  
 labelStyle={'display': 'inline-block', 'margin-right': '20px'}  
 ),  
 html.Hr(),  
 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'),  
 page\_size=10,  
 style\_table={'overflowX': 'auto'},  
 style\_cell={'textAlign': 'left'}  
 ),  
 html.Br(),  
 html.Hr(),  
 html.Div(className='row', style={'display': 'flex'}, children=[  
 html.Div(id='graph-id', className='col s12 m6'),  
 html.Div(id='map-id', className='col s12 m6')  
 ])  
])  
  
@app.callback(Output('datatable-id', 'data'), [Input('filter-type', 'value')])  
def update\_dashboard(filter\_type):  
 if filter\_type == 'water':  
 dff = db.read({"breed": {"$in": ["Labrador Retriever Mix"]}})  
 elif filter\_type == 'mountain':  
 dff = db.read({"breed": {"$in": ["German Shepherd"]}})  
 elif filter\_type == 'disaster':  
 dff = db.read({"breed": {"$in": ["Belgian Malinois"]}})  
 else:  
 dff = db.read({})  
 df = pd.DataFrame.from\_records(dff)  
 df.drop(columns=['\_id'], inplace=True)  
 return df.to\_dict('records')  
  
@app.callback(Output('datatable-id', 'style\_data\_conditional'), [Input('datatable-id', 'selected\_columns')])  
def update\_styles(selected\_columns):  
 if selected\_columns is None:  
 return []  
 return [{'if': {'column\_id': i}, 'background\_color': '#D2F3FF'} for i in selected\_columns]  
  
@app.callback(Output('graph-id', "children"), [Input('datatable-id', "derived\_virtual\_data")])  
def update\_graphs(viewData):  
 dff = pd.DataFrame(viewData)  
 if 'breed' in dff:  
 fig = px.pie(dff, names='breed', title='Breed Distribution')  
 return [dcc.Graph(figure=fig)]  
 return []  
  
@app.callback(Output('map-id', "children"),  
 [Input('datatable-id', "derived\_virtual\_data"),  
 Input('datatable-id', "derived\_virtual\_selected\_rows")])  
def update\_map(viewData, index):  
 if viewData is None or index is None:  
 return []  
 dff = pd.DataFrame.from\_dict(viewData)  
 row = index[0] if index else 0  
 return [  
 dl.Map(style={'width': '1000px', 'height': '500px'}, center=[30.75, -97.48], zoom=10, children=[  
 dl.TileLayer(id="base-layer-id"),  
 dl.Marker(position=[dff.iloc[row, 13], dff.iloc[row, 14]], children=[  
 dl.Tooltip(dff.iloc[row, 4]),  
 dl.Popup([html.H1("Animal Name"), html.P(dff.iloc[row, 9])])  
 ])  
 ])  
 ]  
  
app.run\_server(debug=True)

# Dashboard Screenshots

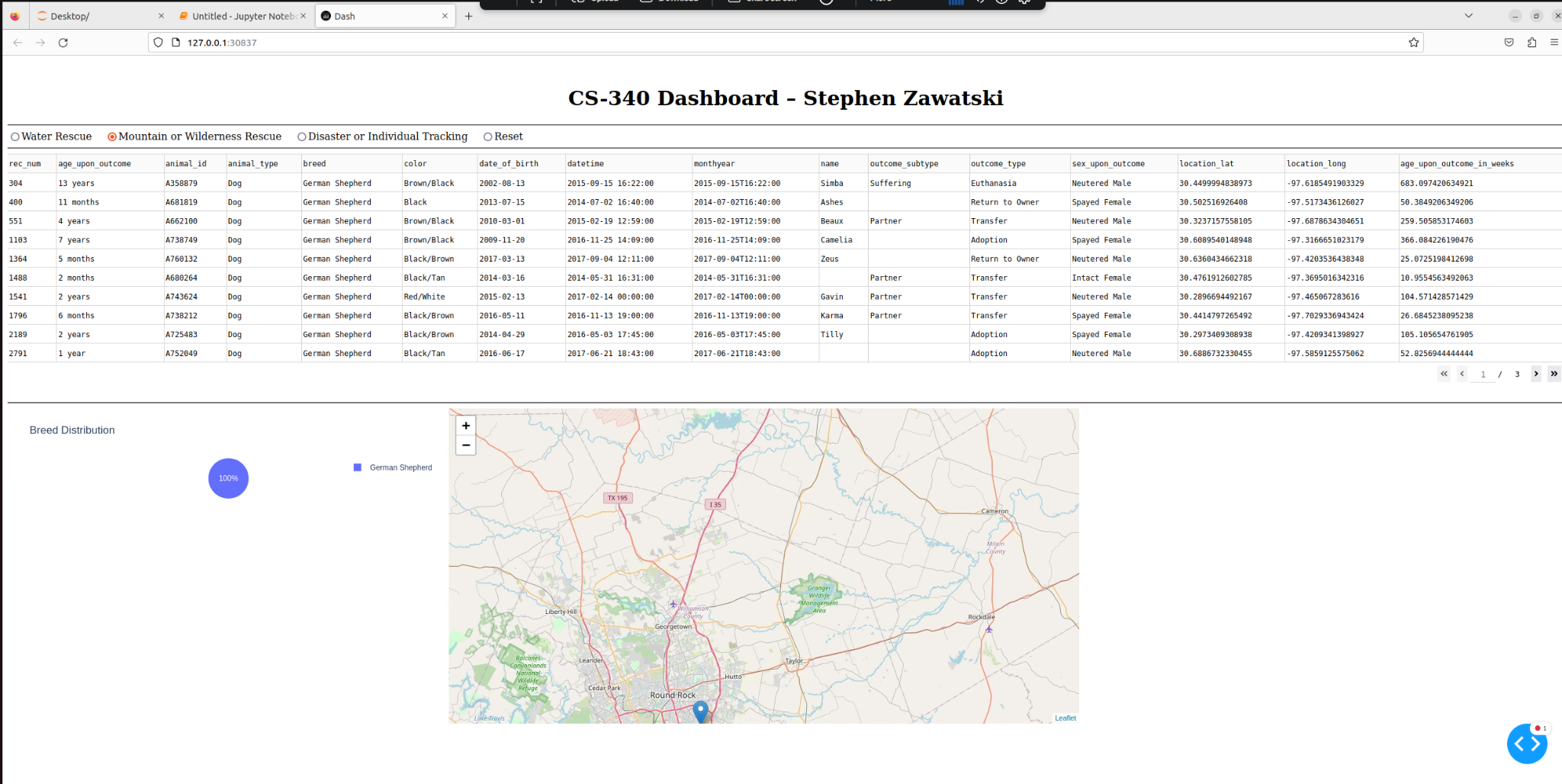
* Dashboard with filters and DataTable



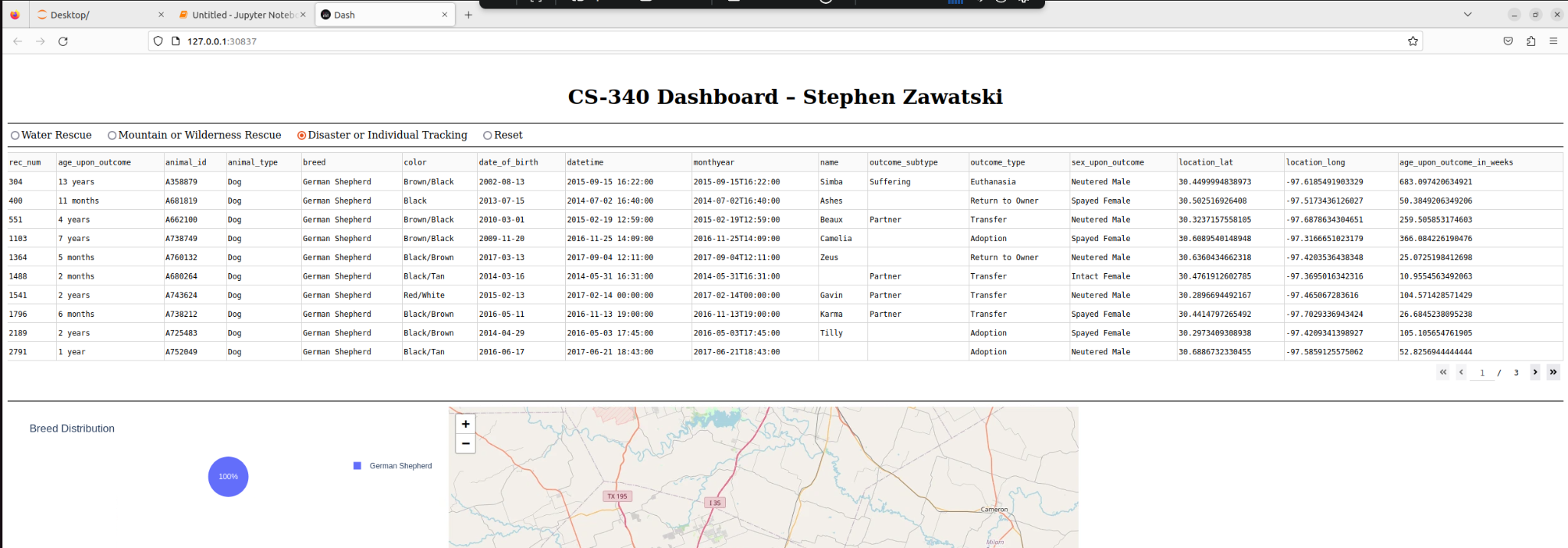
* Water Rescue filter applied



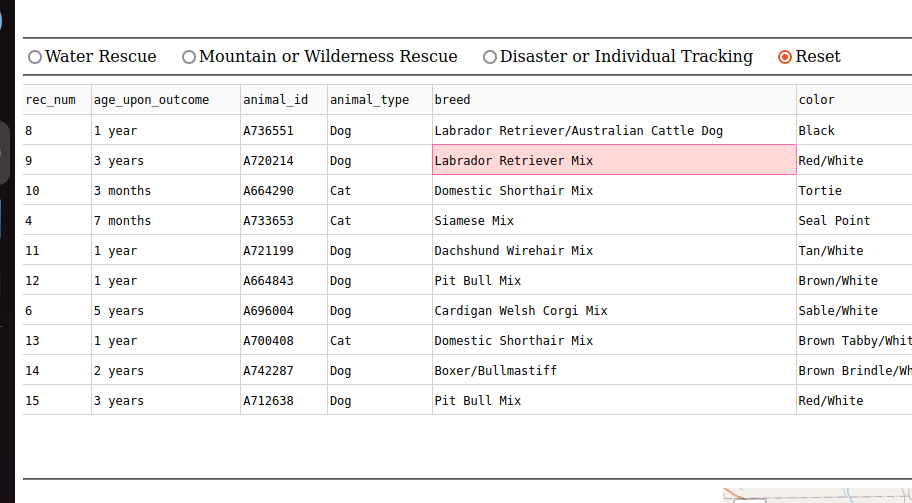
* Mountain/Wilderness Rescue filter applied



* Disaster/Individual Tracking filter applied



* Pie chart showing breed distribution



* Geolocation map displaying animal marker

