Grazioso Salvare Animal Finder Application Readme

Keith Breazeale

Southern New Hampshire University

20-November-2023

Version 2.0

## About the CS340 Grazioso Salvare Animal Finder Application

This application is two-fold. For the backend, the data was imported from a csv and functionality was added to Create, Read, Update, and Delete records from the dataset with the help of Python and Mongo. For the frontend, the data is imported into a dashboard, allowing the users to review the dataset in a user-friendly manner. The data can be filtered by the type of rescue that needs to take place and, ultimately, provide two visuals for review. The first review is a pie chart, allowing the user to see the percentage of animal breeds that are currently in the dataset that qualify for such a rescue. The second is a geo-locational visual, placing a marker for where the animal is in the United States.

## Installation

Python Linux: [Installing Python 3 on Linux — The Hitchhiker's Guide to Python (python-guide.org)](https://docs.python-guide.org/starting/install3/linux/)

Python Windows: [How to install Python 2.7 and 3.6 in Windows 10 [add python PATH] | by Jose Miguel Arrieta | Data Science](https://datascience.com.co/how-to-install-python-2-7-and-3-6-in-windows-10-add-python-path-281e7eae62a) (Note: The instructions for Python 3.6 will also work for 3.7)

Jupyter Notebook (Linux and Windows): [How to Use Jupyter Notebook in 2020: A Beginner’s Tutorial (dataquest.io)](https://www.dataquest.io/blog/jupyter-notebook-tutorial/)

MongoDB Linux: [Install MongoDB Community Edition on Ubuntu — MongoDB Manual](https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-ubuntu/)

MongoDB Windows: [Install MongoDB Community Edition on Windows — MongoDB Manual](https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/) (highly recommend installing MongoDB as a Service and the usage of MongoDB Compass)

**Optional but Highly Recommended**

To harden the security of MongoDB it is recommended to change the default listening port which is usually 43981.

For Linux: Open /etc/mongod.conf with your favorite code editor and search for the following lines:

Net:

Port: 43981

Update the port number to a different port number.

Save and then restart the service with sudo service mongodb restart.

For Windows:

Open a terminal and type: net stop MongoDB.

Update the mongod.cfg file located at C:\mongo (or wherever you chose to install MongoDb).

Find Port: 43981 within the configuration file. Update the port number to a different port number.

Save and then restart the service with net start MongoDB.

Within the animal\_shelter.py file, update line 16 to use the updated port number.

## Getting Started

Make sure you have the csv file containing the data before attempting to set up the dataset.

1. Make sure the Mongo Service is running.
2. Import the dataset.
   1. Text

      Description automatically generated
3. Type in “mongo” from the prompt and verify that the AAC dataset and animals collection is present
   1. Text

      Description automatically generated
4. Create a breed index
   1. Text

      Description automatically generated

## Usage

**Below are the examples of code coming from animal\_shelter.py. It is highly recommended to have this file for reference.**

### Code Example

Authentication:  
 *def \_\_init\_\_(self,username,password):*

*# init to connect to mongodb without authentication*

*# self.client = MongoClient('mongodb://localhost:43981') #Update the port number as needed*

*# init connect to mongodb with authentication*

*self.client = MongoClient('mongodb://%s:%s@localhost:43981/?authMechanism=DEFAULT&authSource=AAC' % ('aacuser','aacuser')) # Update the port number as needed*

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

Create Method:

*def create(self, data):*

*if data is not None:*

*self.database.animals.insert(data) # data should be dictionary*

*return True*

*else:*

*raise Exception("Nothing to save, because data parameter is empty")*

*return False*

Read Method:

*#Read Method*

*def read(self, data):*

*if data is not None:*

*return self.database.animals.find\_one(data)*

*else:*

*print('Nothing to read, because data parameter is empty')*

*return False*

Update Method:

*#Update Method*

*def update(self, data, updateData):*

*if data is not None:*

*result = self.database.animals.update\_many(data, {"$set" : updateData})*

*else:*

*return "{}"*

*print ("Record Updated!")*

*return result.raw\_result*

Delete Method:

*#Delete Method*

*def delete(self, data):*

*if data is not None:*

*result = self.database.animals.delete\_many(data)*

*else:*

*return "{}"*

*print("Record Deleted")*

*return result.raw\_result*

**End of code examples from animal\_shelter.py**

### Tests

Using a Jupyter Notebook, execute the below test:

from animal\_shelter import AnimalShelter

shelter = AnimalShelter("aacuser","aacuser")

data = {"1": 3,

"age\_upon\_outcome" : "4 years",

"animal\_id" : "27-Mar-2023\_test",

"animal\_type" : "Dog",

"breed" : "Bluetick Coonhound",

"color" : "Black and White",

"date\_of\_birth" : "2019-01-01",

"datetime" : "2019-01-01 00:00:00",

"monthyear" : "2019-01-01T00:00:00",

"name" : "Courage",

"outcome\_subtype" : "",

"outcome\_type" : "Adoption",

"sex\_upon\_outcome" : "Female",

"location\_lat" : 35.651150,

"location\_long" : -78.746020,

"age\_upon\_outcome\_in\_weeks" : 220}

*#Create Test*

shelter.create(data)

print("Animal added successfully!")

Graphical user interface, text, application

Description automatically generated

*#Read Test*

#Read Test

result = shelter.read({"name" : "Courage"})

print(result)

A picture containing text

Description automatically generated

Update Test

#Update Test

result = shelter.update({"name" : "Courage"}, {"sex\_upon\_outcome" : "Spayed Female"})

print(result)

#Confirm update

result = shelter.read({"name" : "Courage"})

print(result)

A picture containing application

Description automatically generated

Delete Test

#Delete Test

result = shelter.delete({"name" : "Courage"})

print(result)

#Confirm deletion

result = shelter.read({"name" : "Courage"})

print(result)

Graphical user interface, text, application

Description automatically generated

**Below are the coding examples from the AnimalDashboard.ipynb Jupiter Notebook. It is highly recommended to have this file for reference. The below is not the full code.**

app = JupyterDash('Grazioso Salvare Animal Finder Dashboard')

image\_filename = 'GSLogo.png' # replace with your own image

encoded\_image = base64.b64encode(open(image\_filename, 'rb').read())

app.layout = html.Div([

html.Center(html.B(html.H1('Grazioso Salvare Animal Finder Dashboard'))),

html.Center(

html.Img(src='data:image/png;base64,{}'.format(encoded\_image.decode()))),

# Unique identifier

html.Center(html.H1('Developed by Keith Breazeale 7-November-2023')),

html.Hr(),

#Enhancement Comment - Dropdown Filter Setup

html.Div([

dcc.Dropdown(

id='dropdown',

options=[

{'label': 'Water Rescue', 'value': 'WR'},

{'label': 'Mountain or Wilderness Rescue', 'value': 'MR'},

{'label': 'Disaster or Individual Tracking Rescue', 'value': 'DR'},

{'label': 'Reset', 'value': 'reset'}

],

value='Reset'

),

html.Div(id='dd-output-container')

]),

html.Hr(),

#Enhancement Comment - Interactive Table Settings

dt.DataTable(

id='datatable-id',

columns=[

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

],

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

editable=False,

filter\_action="native",

sort\_action="native",

sort\_mode="multi",

column\_selectable=False,

row\_selectable=False,

row\_deletable=False,

selected\_columns=[],

selected\_rows=[0],

page\_action="native",

page\_current=0,

page\_size=10,

),

html.Br(),

html.Hr(),

#This sets up the dashboard so that your chart and your geolocation chart are side-by-side

html.Div(className='row',

style={'display': 'flex'},

children=[

dcc.Graph(

id="pie-chart",

className='col s12 m6',

),

html.Div(

id='map-id',

className='col s12 m6',

)

])

])

### Screenshots

Screenshot Example from the Animal\_Shelter.py file

A screenshot of a computer program

Description automatically generated

Screenshot Example of the AnimalDashboard Jupiter Notebook

A screenshot of a computer

Description automatically generated

Screenshots of the Dashboard

First Launch / Reset

A screenshot of a computer

Description automatically generated

Select the dropdown and make a choice

Water Rescue

A screenshot of a computer

Description automatically generated

Mountain or Wilderness Rescue

A screenshot of a computer

Description automatically generated

Disaster or Individual Tracking Rescue

A screenshot of a computer

Description automatically generated

## Contact

James “Keith” Breazeale

James.breazeale@snhu.edu