# CS 340 Project 2

Adam Conger, CS 340

## About the Project/Project Title

* In the CRUD Python module, we developed a web application that connects to the client-side user interface (the dashboard) to a database by implementing the Python syntax. The project uses an animal shelter database that imports 1000 documents to the mongo shell. This Project is meant to show students how to use PyMongo or Python interacting with the MongoDB database. This project shows us how to import databases, creating documents (C – Create), querying information (R – Read), make changes to information (U – Update), delete records (D – Delete), authentication, and automation (scripting). This project helps the student to understand the CRUD method for databases.

In addition, in project two, we were asked to develop queries that allow users to filter the database, build interactive options through which users will activate the filters. Project 2 allow me to build skills for adding widgets to a web application, that are easy to use for the user.

## Motivation

* This web application was developed, so that eventually it will be used to connect the user interface component to the database component of the dashboard in Project Two. The motivation behind this project is to develop a portable Python module that enables CRUD functionality for a data connection with the Austin Animal Center data set. Full stack development helps developers become more well-rounded. For a developer to be capable of developing front end and back end will help them to become more valuable for companies in the future.

## Getting Started

* The first thing that needs to be done when getting started is starting up the mongo shell with the authorization code*.* This needs to be done so that the Juptyer starts to connect with the .py file and the mongo shell for testing.

## Installation

* It is recommended to have the mongo shell and Jupyter Notebook in order to create a Python testing script that imports your CRUD Python module to call and test the create and read instances of CRUD functionality. In order to make sure that the mongo and Ju

## Usage

### Code Example

* It is recommended that the developer creates classes in Python syntax that will be calling the documents and function in the mongo shell. This is important to do so that the two software’s can communicate effectively between one another. In addition to coding communication, we will also need to code condition statements to be able to test if a particular document or file is found and if it is then the iteration will return true. Else if the document is not found then the condition in the code iteration will return a false result.

Below is some of the code that was used to create the Project2CompletedDashboard.ipynb file. This is the file that is used to call the create class, create the pie chart, as well as the clickable buttons (interactive options) that a user can toggle through to show different information on the dogs that were rescued. The code created in the file also created the geolocation chart that is interactive for the user.

### Tests

### *Describe and show how to run the tests with code examples.*

### Graphical user interface, text, application, Word Description automatically generated

The above screenshot displays the Dashboard that was created when running the python code. The code that was altered is shown below.

**app = JupyterDash('SimpleExample') //**

**image\_filename = 'Grazioso Salvare Logo.png' # replace with your own image //**

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

### *Graphical user interface, text, application Description automatically generated*

### *Graphical user interface, text, application Description automatically generated*

The code that was used to display the above dashboard with the interactive radio buttons is shown here.

**app.layout = html.Div([**

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

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

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

**html.Center(html.P("by Adam Conger")),**

**html.Hr(),**

**html.Div(**

**dcc.RadioItems( #Radio buttons for filtering**

**id = "filter",**

**options = [**

**{"label" : "Water Rescue", "value" : "wr"},**

**{"label" : "Mountain or Wilderness Rescue", "value" : "mtn"},**

**{"label" : "Disaster or Individual Tracking", "value" : "dis"},**

**{"label" : "Reset", "value" : "RESET"}],**

**value = "RESET",**

**labelStyle = {"display" : "inline-block"}**

**)**

**),**

### *Graphical user interface Description automatically generated*

### Above is a screen shot of the pie chart along with the interactive geolocation chart.