**CS 340 README**

**About the Project/Project Title**

*This project includes a CRUD module that interacts with an animal shelter's database and an .ipynb file which utilizes the CRUD module’s functionality in order to display data in the form of a data table, a graph, and an interactive map. Employees of Grazioso Salvare can use this application in order to identify and categorize different types of search and rescue animals.*

**Motivation**

*The module was created to be imported in order to be used to easily organize animal shelter information about the animals at the shelter. The application as a whole was created both to assist the employees of Grazioso Salvare with their mission to identify dogs that could be good candidates for search and rescue training and to benefit shelters by finding jobs for the dogs there so that shelters have more space for other animals.*

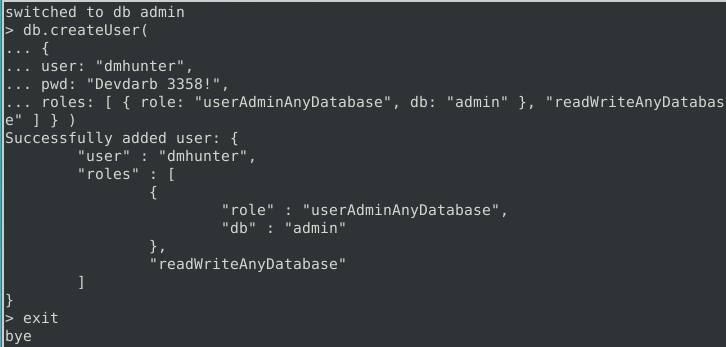
**Installation and Getting Started**

*Users will (at this time) need MongoDB installed locally on their device. The port # is currently hardcoded into the module. The database will need to be reflective of the hardcoded database name "AAC", the collection will need to be "animals", and the user will have needed to create an admin account in MongoDB with the appropriate username and password with access to the "AAC" database and "animals" collection.*

*You will need to import a .csv file called “aac\_shelter\_outcomes.csv” in order to start working with the animal data. This can easily be done by writing the following command in MongoDB:*

*Mongoimport -d DB\_NAME -c COLLECTION\_NAME —file aac\_shelter\_outcomes.csv*

*Because the username and password are hardcoded into the .ipynb file (this should never be done in the real world) you will have to create the appropriate account in MongoDB. You can do that like this:*

**

*Be sure to have MongoDB running locally prior to inserting, reading, updating, or deleting documents.*

*The module has already been imported into the .ipynb file and instantiated and the appropriate methods have been called in order to appropriately search and return search and rescue animal information.*

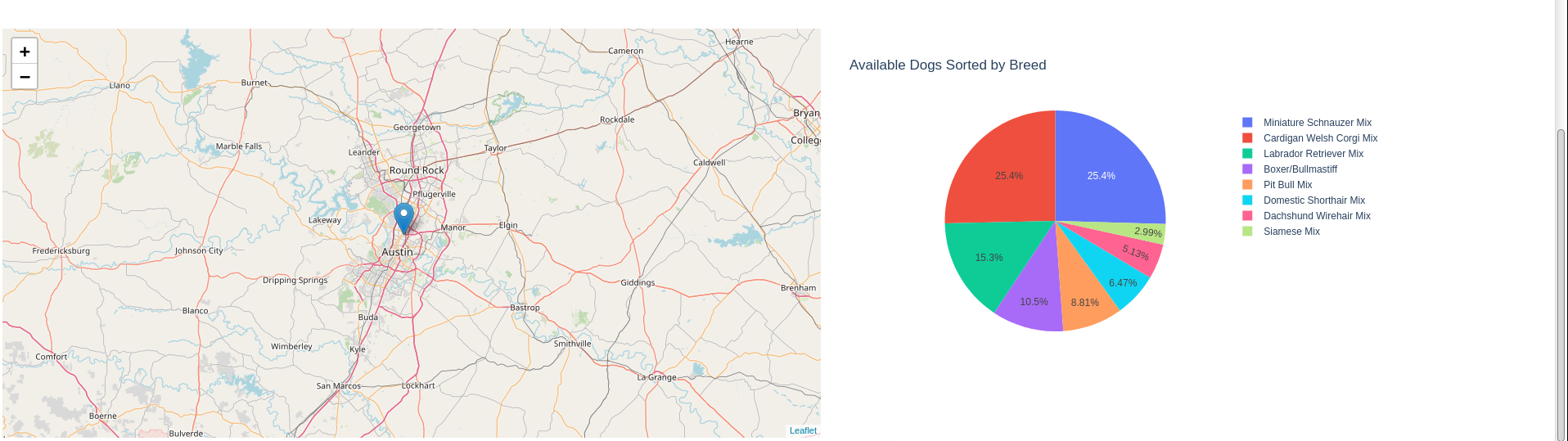
**Usage**

*When the user runs the application, they will be met with the following interactive dashboard:*

*1/2 Dashboard screenshot:*

**

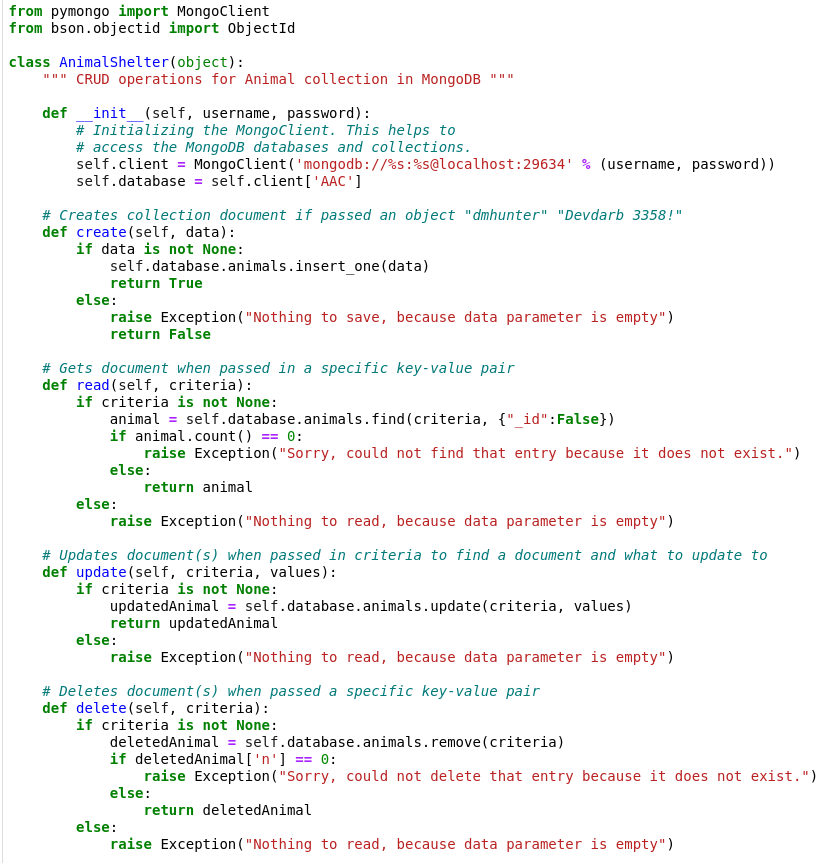
*2/2 Dashboard screenshot:*

**

*Users can select the type of rescue animal from the dropdown menu above the data table. Upon selection the interactive map and the pie chart will change to reflect the user’s selection. You’ll notice there are check boxes next to each row within the table. Selecting a checkbox will show the location of the selected animal on the interactive map. Users can also sort by using the arrows next to each field in the data table.*

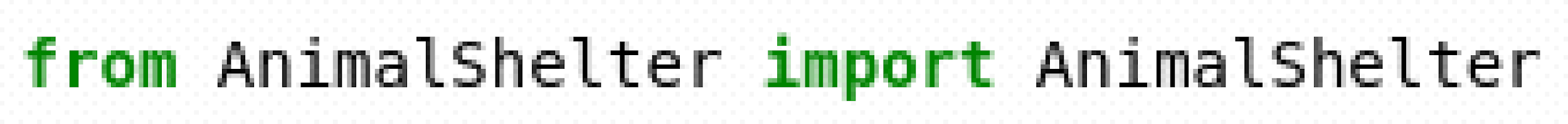
**Code Example and Screenshot:**

Below you’ll find a screenshot of the code that makes up the module that the application imports and uses in order to interact with the database.

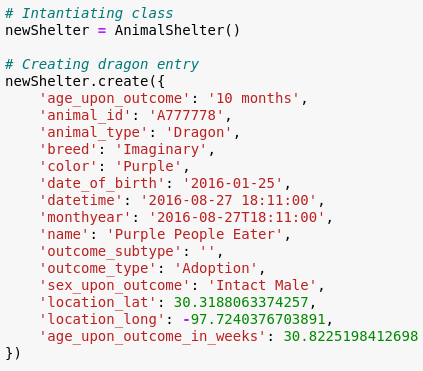
**

**How can I use the AnimalShelter module in my own project?**

*Here is an example of how to import the module into your own project:*



*Here is an example of how to instantiate the class and insert a document:*

**

*Here is an example of how to read the document that you just created:*

**

*Here is an example of how to update a document (or documents):*

**

*Here is an example of how to delete a document (or documents):*

**

**Description and justification of tools used to create the interactive shelter dashboard:**

*MongoDB -*

*This project uses MongoDB in order to store data. This data is then read, created, updated, or deleted with assistance from the Python module PyMongo, which provides an API for performing CRUD operations on MongoDB databases and collections.*

*PyMongo -*

*A Python module widely used in the Python community for working with MongoDB. PyMongo provides a straightforward interface for developers to easily interact with MongoDB databases.*

*Dash -*

*An open-source web framework for building interactive web apps with Python. Dash allows access to components for creating interactive web interfaces, including graphs and tables like the ones featured in this project. The components Dash provides make it easy for the user to interact with the web app.*

**Steps taken to complete the project:**

1. *Installed MongoDB locally.*
2. *Created a Mongodb database called “AAC”.*
3. *Created a collection within that database called “animals”.*
4. *Imported a csv file full of animal data into the animals collection.*
5. *Enabled user authentication.*
6. *Created the importable module “AnimalShelter” which handles CRUD operations.*
7. *Created the .ipynb file which imports and instantiates the AnimalShelter module in order to allow users to interact with the database via an interface within an easy-to-use web app.*

**Challenges and solutions:**

*I initially had a difficult time getting the AnimalShelter module to interact with the database at all until I realized that I hadn't started the MongoDB server as a background process. I also had a difficult time getting the data table to change based on the dropdown menu option the user selected. I was able to overcome this by using the correct parameter in the “update\_dashboard()” function. Another thing I struggled with was just working with the Jupyter Notebook itself. The reason for this is that it will tell you the exact line of code that it believes an error is on, but the code in the Jupyter Notebook is not labeled accordingly so you have to search through your code squinting until you find the area that it is talking about. I never found a solution for this, but I’m sure if I worked with Jupyter Notebook more I could find one.*

**Contact**

Devin Hunter