Southern New Hampshire University

Module Seven – README Final

Emily Nagorski

Satish Penmatsa

CS340-R1895 – Client/Server Development

10/14/23

# CS 340 README Template

## About the Project: Animal Database, Grazioso Salvare

This program was created for Austin Animal Center (AAC), so that employees of the animal center can access animal information within the database. With the addition of the create, read, update, and delete functions, users can add new animal profiles to the database, search by desired parameter, update any existing animal profiles and delete any existing ones as well.

## Motivation

The application will help the client quickly maneuver through the database, being able to search and find animals that meet desired parameters. Implementing an easier way for the user to interact with the animal center database will benefit the client and improve their business functions.

## Getting Started

To get a local copy up and running, follow these simple example steps.

1. Open a terminal window and access MongoDB
2. Import the CSV file “aac\_shelter\_outcomes.csv”, this holds all of the database entries
3. Once imported, enter user credentials to gain access to functions that will have previously been set up for you by the admin. If you do not have user credentials, contact the administrator before moving on from this step.
4. Ensure crud.py has been saved to your documents. Should you need to make any additions or edits to the python file, it can be accessed and edited in a separate terminal window with vim.
5. From a separate terminal window, run command “python3 crud.py”

A screenshot of a computer

Description automatically generated

## A screenshot of a computer Description automatically generated

Getting started note: the Create portion of the python code is written as such, so that when a new pet profile is created, if the entry is successful the terminal will print ‘true’, a value of ‘false’ will be returned for any unsuccessful entries. The Read function exists as a loop and will loop through and return all results that match the request. This may mean thousands of results are returned so it is advised to add multiple parameters if possible. New functionality will continue to be added, stay tuned for updates.

1. Launch Jupyter file (should automatically open in a web browser). Once in Jupyter notebook, any layout changes and additional features can be added or manipulated.
   1. Radio Buttons for pre-set queries:
      1. New buttons need to be created and assigned a value in the app layout section of the Jupyter file
      2. Once a button has been created and assigned a value, you can move down to update\_dashboard(filter\_type) to assign the search parameter to a new elif statement. (be sure your parameters are declared exactly as they appear within the data; any variation or misspelling will not produce the desired results)
2. The map widget will automatically update with the first row of data displayed on the table.
   1. This occurs because in the update\_map section I declared that if index is None row = 0, else row would equal whatever is selected by the user. This can certainly be adjusted; however, I recommend initialing the map with a declared row to avoid errors.
3. The pie chart widget will always show whatever data is currently being held in the table. This means that when the page is initially populated and reset, the pie chart will attempt to show you breed data for every animal held in the database. This creates a fairly useless table to start. As the search results are narrowed with the radio buttons we created previously, the pie chart will update and supply to user with breed data included in their query.
   1. If the client has a desire for a pie chart that shows data besides breed, this would need to be updated from the Jupyter file in the update\_graphs(viewData) section. You would just need to update the ‘names’ and ‘values’ dff value to the requested trait.

## Installation

Libraries needed for python functionality:

* from pymongo import MongoClient
  + pymongo is a MongoDB driver that helps connect MongoDB to a python application. Pymopngo is an official driver created and supported by MongoDB which is why I chose to use it for the purposes of this project.
* from bson.objectid import ObjectId

Additionally, you will need to ensure access to the MongoDB, as it is the database program at the core of our product. MongoDB was selected for its cross-platform availability and its clear ability to fill the needs of our client at this time. MongoDB is bet for data entering, storage and editing which is the main functionality requested by the client.

Libraries needed in Jupyter for Dash

* from jupyter\_dash import JupyterDash
* import dash\_leaflet as dl
* from dash import dcc
* from dash import html
* import plotly.express as px
* from dash import dash\_table
* from dash.depdendencies import Input, Output, State
* import base64
* import os
* import numpy as np
* import pandas as pd
* import matplotlib.pyplot as plt

Dash framework integrates really well with python libraries and framework. It also includes many pre-built components that directly relate to data tables and manipulation making it a desirable choice for the client’s purposes.

It is possible that you may run into additional issues when attempting use of MongoDB and python integration, below I will include a few commands that may address roadblocks you come across.

* ‘update\_many’ when used in MongoDB this will have the program find the document and then update if found, or return a message that the document was not found
* ‘show collections’ is a command that can be used in MongoDB if you are wanting a list directory of the collections available (hint: we mainly access the collection ‘animals’
* ‘printenv | grep -I mongo’ is helpful when needing to discover the host and port value when logged in as a specific user, the port number must be up to date in the python file
* From your command line in a new window enter 'cd Documents/’ to display the available documents, you should see crud.py listed, from there enter ‘vim crud.py’ which will then show you the contents of that py file
* When in the file ‘i’ allows you to insert and edit code within the file, when you get to a point where you would like to exit and save hit the escape key and then ‘:wq’ to write and quit the vim editor, when back in the command line enter ‘python3 crud.py’ to run the script

## Usage

### Code Example

*A screenshot of a computer

Description automatically generated*

Above is an example of how to run the python file from the command line. The program notifies the user when each function is running and then provides any additional output as defined in the test. For example, the print line ‘true’ after a new entry has been added to the database and then the read command displays the specifically requested information.

### A screenshot of a computer program Description automatically generated

### Above is a screenshot of the python file to display the CRUD operations. Create is used to add new animal entries to the animals collection. This is achieved by using the ‘insert\_one’ command and returning ‘true’ once complete. Read searches the database based on the parameters requested by the user, and then prints any results matching those parameters. Update takes new values and a query and updates the specified values or the query by utilizing the ‘update\_many’ command, returning a message stating the number of documents updated upon completion. Delete takes the entered value, and by using the ‘delete\_one’ command, will remove one matching document from the collection.

### *A screenshot of a computer Description automatically generated*Tests

Above is the python code that tests the CRUD functionality.

## A screenshot of a computer Description automatically generatedFrontend Features :

Above is the initial landing page when you access the front-end of the Grazioso Database. You will see a choice of four radio buttons, a table that shows the first ten results and the ability to change the page number, a pie chart and a map with the location of the selected animal pinned.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedAbove is the displayed results from the “Water Rescue” search query. Below is the “Mountain Rescue” query results.

A screenshot of a computer

Description automatically generated

## Above is the “Disaster Rescue” query results, below displays the “Reset” functionality.

## A screenshot of a computer Description automatically generated

## Contact

Your name: Emily Nagorski