# CS 340 Animal Shelter Full Stack Application

## About the Project/Project Title

*Project Two, Grazioso Salvare Austin Animal Shelter Dashboard and Database*

The The *Grazioso Salvare Austin Animal Shelter Dashboard and Database* was created for CS-340 as a website to view and search the database of the associated fictional company including various key search features as requested by the company, and infographics about displayed information.

Features include:

A CRUD (Create, Read, Update, Destroy) class for the Animals collection of the AAC database in MongoDB written in Python.

A dashboard created in DASH with simple infographics such as a map and two pie charts to display data in organized segments.

Multiple radio sorting objects to search the database with specific logical MongoDB queries.

## Motivation

This website was created as part of the course requirements of CS-340, and was designed as such to satisfy course requirements.

It functions as the minimum viable product (MVP) for a fictional company by the name of Grazioso Salvare, a fictional Animal Shelter with a database of sheltered animals and their outcomes.

## Getting Started

First, follow CS-340 Module 3 requirements, as can be found at the [User Management in MongoDB](https://learn.snhu.edu/content/enforced/1918326-CS-340-10069.202551-1/course_documents/CS%20340%20User%20Management%20in%20MongoDB.pdf?isCourseFile=true&ou=1918326) document. This ensures you have the proper AAC user instantiated in your MongoDB interface for the class to function properly.

From there, download both the python class and the ipnyb files as seen in the Documents directory. There, you can run the *AnimalShelterTest.ipnyb* file which includes and utilizes the *AnimalShelter.py* CRUD class.

Once you ensure AnimalShelter.ipnyb successfully runs, you may install ProjectTwoDashboard. Utilizing ProjectTwoDashboard is as simple as running the associated ipnyb file in the Jupyter notebook tab as AnimalShelter.ipnyb did – you should have access to your own local version of the Project!

## Installation

Tools:

JupyterNotebook, which can be found at the [Jupyter Notebook Install](https://jupyter.org/install) page.

MongoDB, which can be found on it’s respective [MongoDB Install](https://www.mongodb.com/docs/manual/installation/) page. MongoDB was utilized for it’s excellent compatibility with both individual query through terminal and compatibility with Python at a higher level, allowing manipulation from outside sources like the CRUD module.

Python, a modular programming language and several respective libraries, which can be found at it’s [install page](https://www.python.org/downloads/).

[Plotly’s DASH](https://dash.plotly.com/), a modular Python framework used to produce, build, and deploy efficient HTML documents with built-in infographics.

## Usage

*Project Two, Grazioso Salvare Austin Animal Shelter Dashboard and Database*

This dashboard is used to navigate and manage the Austin Animal Shelter database in a user-friendly manner, more so than the primitive CRUD class. See attached video for a demonstration of features and usage:

**

*CRUD CLASS*

This class is utilized to provide an abstract overhead CRUD interface to the Animals collection of the AAC database with specified user credentials, and port information to provide ease of access to potential users or developers.

User is authenticated through the set python module, just modify user credentials:

A screenshot of a computer

AI-generated content may be incorrect.

*A screenshot of a computer program

AI-generated content may be incorrect.*

### Tests

To run a CRUD operation, simply call its respective method:



Expected Output:

True, operation successful.

Example: Each method with respective queries: A screenshot of a computer program

AI-generated content may be incorrect.

## Steps

This project was created in a series of steps, initially through creating the CRUD module and associated user credentials for the MongoDB instance. Once that was completed, the program was developed utilizing the Dash framework to produce the MVP.

## Challenges Overcome

This project posed numerous challenges which required significant refactoring of not only the program but also the source code.

First, Jupyter Notebook depreciated Jupyter\_Dash in favor of the superior Dash framework, and as such had several unsolved bugs that required me to refactor Jupyter\_Dash to the now updated Dash framework. Such bugs included a TypeError in the Jupyer\_Dash directory among other things upon import.

I also faced significant issues with the Map rendering, specifically out of bounds indexes – this is likely because when changing the objects and removing them from the datatable through sorting you are no longer accessing that index. I got around this by adding more conditional checks, and getting rid of previously redundant ones.

Lastly, the entire csv file was imported via mongoimport incorrectly on my aacuser instance as I didn’t know how to properly import a CSV with proper column headers at the time, and this resulted in all search queries always returning an empty array. I didn’t realize this at the time, but to solve this problem I reimported the csv file with updated headers and removed my logic to just add them in post, allowing my CRUD module to successfully read the database and return proper results.

* How do you write programs that are maintainable, readable, and adaptable? Especially consider your work on the CRUD Python module from Project One, which you used to connect the dashboard widgets to the database in Project Two. What were the advantages of working in this way? How else could you use this CRUD Python module in the future?

Maintainable code is written widely through documentation and utilizing important principles such as Object-Oriented Design and DRY, (Don’t repeat yourself). Through DRY and Object-Oriented Design, developers can create efficient code that is maintainable through one segment as opposed to multiple repeat actions. The CRUD Python module is an excellent example, utilizing Object Oriented Design via four core access functions that are easily modifiable through parameters, hard-coded login credentials modifiable at the file level, and more. In the future, the Python CRUD module may be useful in various applications, especially those with a MongoDB context.

* How do you approach a problem as a computer scientist? Consider how you approached the database or dashboard requirements that Grazioso Salvare requested. How did your approach to this project differ from previous assignments in other courses? What techniques or strategies would you use in the future to create databases to meet other client requests?

As a software engineer, each problem is first an architectural problem, and a code problem second. Through this class we were given complex architectural tools such as DASH, MongoDB, and Python Libraries supporting each to effectively create a simplistic architecture that accomplished a deeply complex task. To approach the database or dashboard requirements I first had to learn how to utilize each library effectively and produce a maintainable architecture that utilized those libraries to solve the greater problem. In the future, I could utilize knowledge I learned to produce a better MVP with a deeper knowledge of each library.

* What do computer scientists do, and why does it matter? How would your work on this type of project help a company, like Grazioso Salvare, to do their work better?

Software engineers, and Computer Scientists in general are the forefront engineers for software problems. By proposing architectural solutions to modern problems, engineers provide a bridge between vision and accomplishment and help to ensure the end goal is met. Without engineering, ideas cannot come to fruition. Work on full-stack projects such as the Grazioso Salvare Animal Shelter dashboard help engineers like myself learn more about the tools and components we have at our disposal to solve problems easier and more efficiently for each client.

## Contact

Thomas Bryant