# CS 340 - *Austin Animal Center Dashboard* README File

**Daniel Gorelkin**

## About the Project

*The Austin Animal Center project offers the developer the opportunity to easily connect to the ‘AAC’ database in MongoDB and perform Create, Read, Update, and Delete (CRUD) operations seamlessly, without concerns about the connectivity process or the internal implementation of the methods in Python. This project is a full-stack application between the MongoDB database, where the documents for the ‘animals’ collection are stored, and the main user interface from which the data can be queried, rendered, filtered, sorted, and visualized after successful user authentication. The front-end dashboard will enable users to search and filter the various dogs available in the database by the predefined rescue categories, using the specified criteria, and output the filtered results in an interactive and dynamic table. Additionally, the functionality will output a pie chart of all the available dog breeds and display the location of a chosen dog on a geolocation map. The application will also handle errors such as a bad connection to the database, user authentication errors, an empty database, or no search results found.*

## Motivation and Tools

*The purpose of this project is to save valuable development time by building a UI dashboard and connecting your app’s front end to the database while ensuring that the end user utilizes proper authentication and authorization with the minimum permissions necessary to query the database safely and effectively and the pulled data from the database will be easily read filtered and displayed.*

*The project is still in its development phase. Therefore, additional methods and functionality will be implemented later to facilitate convenient use and effective queries. This code is offered as open source; thus, distribution, pull requests, and contributions are welcome. Despite the examples in this document being intended to be used with the ‘AAC’ database within the localhost environment, the setup can be easily modified to fit any similar database locally, or by deploying it into the Atlas environment by MongoDB.*

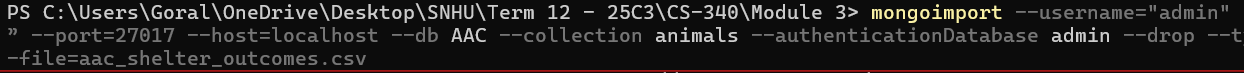
*The tools that were used in this project are MongoDB, which serves as a database for storing and retrieving the data for the project. MongoDB is a schemaless, NoSQL database that stores its documents in collections (tables), in a BSON (JSON-like) format, which perfectly aligns with Python’s dictionary structure. The great benefit of MongoDB for this project is that it can store various attributes from strings to geolocation points, which makes it a perfect fit for the data for the Austin Animal Center. Additionally, it provides great flexibility for running CRUD operations (create, read, update, delete) using the Python programming language. Therefore, choosing and implementing the PyMongo library in our project as a middleware in our AnimalShelter class makes interfacing with Python and integrating the database seamless and efficient.*

*The Dash framework from Plotly, on the other hand, provides the view and controller structure for our web application. It is based on the more robust Flask web framework, and it allows us to create an interactive, browser-based dashboard user interface that can easily handle callback-based user interactions and provide outstanding data visualization with external libraries like Pandas, Leaflet, Plotly Express, and many more. The great benefit of choosing Dash in our application is the simplicity of creating HTML pages and formatting them with a style sheet language such as Bootstrap's CSS framework.*

## Getting Started and Installation

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

*Firstly, you will need to set up your database locally. Assuming that you already have the MongoDB infrastructure on your machine installed, you can simply open your command prompt and run the setup database code and pass the attached CSV file from the* [*GitHub  repo*](https://github.com/Dgoralkin/Portfolio/tree/main/CS_340_ClientServer_Development/Project_1/Project_1_files)*:*

**

*This will create your own local database and load some cleaned data into it. Next, by creating a database user, we can grant him the appropriate level of permissions to the database by running:*

**

*The last line of code will create user access to the ‘AAC’ database, which will later authenticate the user and allow them to perform read and write operations within the database. This step is crucial as it prevents the risk of deleting the database or damaging its performance.*

*To achieve optimal performance, we should consider setting indexes in the database to optimize our queries on the relevant data. Run the following code to create a compound index in the database for the defined filter queries:*

*A screenshot of a computer

AI-generated content may be incorrect.*

*That will ensure that our CRUD functions will receive their commands from the readWrite user and search the database fields according to the indexed key values. This will ensure smooth operation and fast response times.*

*Next, to install the app, we will install several libraries and modules for Python to ensure a smooth start.*

*After downloading the dedicated AnimalShelter package from our GitHub repository, ensure that you import it into your Python code as follows: . Additionally, we will need to install and import several additional libraries and specific modules, such as Dash, dash\_table, html, and dcc, from dash to create the front-end dashboard for the user interface to act as a controller structure for the web application. This includes dependencies like Input, Output, and State to facilitate communication between the user and the app. In addition, we will utilize dash\_leaflet to display our geolocation map and plotly.express for data visualization, such as pie charts. Lastly, we must import the Pandas library to prepare and manipulate our data, and we must have the pymongo library to set up communication and control interaction with the MongoDB database.*

*A screen shot of a computer

AI-generated content may be incorrect.*

## Functionality, Usage, and Tests

*To run the app, we can simply open our Jupyter notebook and run the ProjectTwoDashboard.ipynb file. The starting state of our dashboard displays the provided company logo, which serves as a hyperlink to the Austin Animal Center website, along with guidance information that prompts the user to authenticate themselves to connect to the database using the following credentials: username = ‘aacuser’, password = ‘SNHU1234’. Note that while the user is not authorized, the dashboard will hide the rendered data from the database, and the filtering checklist will not be active:A screenshot of a computer

AI-generated content may be incorrect.After successful authentication, our dashboard displays a confirmation message indicating that the data has been successfully loaded and presents the data from the database in a dynamic and interactive table. Below the table, we can find two interactive graphs that visualize the data from the table:*

*A screenshot of a computer

AI-generated content may be incorrect.*

*The pie chart illustrates the distribution of data across various animal breeds, while the geolocation map displays the location of the chosen animal from the table. As mentioned above, the table and the widgets are interactive and could be controlled and filtered by selecting the desired checkboxes as filters.A black text on a white background

AI-generated content may be incorrect.*

*By default, all filters are unchecked; therefore, the table displays all records from the database, limiting the number of displayed records to 10 rows per page, enabling pagination, scrolling, sorting by column, and filtering the table by the desired column variables. For example, checking the Water Rescue box, searching for all two-year-old dogs, and selecting the Black color will update our table to show all dog records from the table that correspond to the predefined breeds, sex, training age, and the optional typed search parameters. Therefore, we can see that all records were dynamically updated and are displayed on one page, while the pie chart indicates that all corresponding dogs found belong to a single breed. The geolocation map, on the other hand, was updated to display the location, name (if available), and breed of the selected animal from the table.*

*A screenshot of a computer

AI-generated content may be incorrect.*

*Similarly, our dashboard will react to choosing any other filter rubric, or even several rubrics at once, and update the pie chart with all available breed types from the table:*

*Filter Mountain or Wilderness Rescue: Filter Mountain and Disaster Rescue animals:*

*A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.*

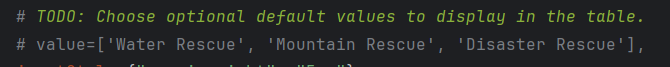
*To reset the filter to its default state and view all available animals from the dataset, simply uncheck all the checkboxes.*

*Note that our dashboard is capable of skipping the user authentication process by uncommenting the following line of code:*

*A screenshot of a computer program

AI-generated content may be incorrect.*

*It also allows us to predefine the default state of the filter checkboxes, adjusting the initial table data results.*



### Testing and Error Handlers

*To test our app’s error handling behavior, we will run several test cases and expect the following app output for such an error case as follows:*

*No username and/or password output: Incorrect username and/or password input:*

*A screen shot of a computer

AI-generated content may be incorrect.A close-up of a login

AI-generated content may be incorrect.*

*In case no available records exist in the database, or no animals were found in the requested filter group, our dashboard will hide the plots and acknowledge the user with an error message as follows:*

*A screenshot of a computer

AI-generated content may be incorrect.*

*Additionally, if a user manually filters the table with a search parameter that does not correspond to any of the animals in the database, our dashboard will hide the graphs and reply with the following error message, showing that the data was successfully loaded, some filters were applied, but no records were found:*

*A screenshot of a computer

AI-generated content may be incorrect.*

## Project Development Process and Challenges

*Just like Rome, the project wasn’t developed overnight. The development phase began by creating a local database using the Mongosh shell and uploading all the data records for Austin Animal Center from the provided aac\_shelter\_outcomes.csv file. Later, we created a unique user access, including setting the credentials and permissions for that user, and indexed the database to optimize our queries. As the database was established, we proceeded to build our AnimalShelter.py model in the MVC paradigm, which will communicate with the AAC animal shelter collection. After successfully testing our model, we shifted our focus to constructing the UI skeleton and creating the app’s logic. In this phase, we first imported all the required libraries and integrated our middleware model through which we will communicate with the database. Next, we moved on to outlining the interface visualization aspect and created buttons, input fields, and containers to form the layout for our app. Lastly, we adjusted our queries to align with the predefined requirements by our stakeholders. And finally, we polished the project by adding some exception handlers and ensured that the visualization and user experience (UX) would be friendly, efficient, responsive, and robust. After testing our completed project and receiving feedback from the stakeholders, we modified some areas and deployed the code to the pipeline.*

## Contacts

Daniel Gorelkin  
[Daniel.Gorelkin@SNHU.edu](mailto:Daniel.Gorelkin@SNHU.edu)  
<https://github.com/Dgoralkin>

**References:**

*Alex Giamas. (2022). Mastering MongoDB 6.x : Expert Techniques to Run High-volume and Fault-tolerant Database Solutions Using MongoDB 6.x. Packt Publishing.*

*errors – Exceptions raised by the pymongo package - PyMongo 4.9.1 documentation. (n.d.).* [*https://pymongo.readthedocs.io/en/4.9.1/api/pymongo/errors.html*](https://pymongo.readthedocs.io/en/4.9.1/api/pymongo/errors.html)

*DAsh DataTable | DAsh for Python Documentation | Plotly*. (n.d.). <https://dash.plotly.com/datatable>

*DASH Documentation & User Guide | Plotly*. (n.d.). <https://dash.plotly.com/>

*Dash Leaflet*. (n.d.). <https://dash-leaflet-docs.onrender.com/>

*Pie*. (n.d.). <https://plotly.com/python/pie-charts/>

Team, M. D. (n.d.). *Users in Self-Managed deployments*. Database Manual v6.0 - MongoDB Docs. <https://www.mongodb.com/docs/v6.0/core/security-users/>

Team, M. D. (n.d.-b). *Run commands*. Mongosh - MongoDB Docs. <https://www.mongodb.com/docs/mongodb-shell/run-commands/>

*Tutorial - PyMongo 4.13.0 documentation*. (n.d.). <https://pymongo.readthedocs.io/en/stable/tutorial.html>

*TutorialsPoint. (2025, March 25). DASh Framework for Python Web Development.* [*https://www.tutorialspoint.com/python\_web\_development\_libraries/python\_web\_development\_libraries\_dash\_framework.htm*](https://www.tutorialspoint.com/python_web_development_libraries/python_web_development_libraries_dash_framework.htm)