

# **CS-340 Austin Animal Control README**

# **CS-340-X6146 Client/Server Development 23EW6**

**Scott Dixon**

**August 16, 2023**

# **Austin Animal Control Animal Categorization System**

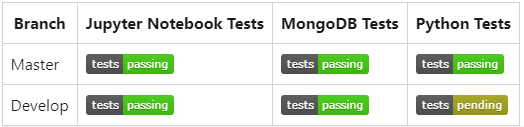
* This project involves developing a system to categorize animals in the care of Austin Animal Control (AAC) based on animal type, breed, age, adoption status, and other factors. The goal is to allow AAC staff to quickly and accurately classify incoming animals to improve care and facilitate adoption.
* The system will utilize MongoDB, and future implementation will use machine-learning models trained on images of animals and data from past intakes along with Dash and Plotly. Staff will be able to upload information along with photos of new animals, and the model will suggest categories and classifications to apply. The categories will allow animals to be housed appropriately and surface key information to potential adopters.
* In addition to assisting with categorization, the system will also track statistics and trends over time related to the types of animals AAC takes in. This data could help inform future operations and outreach. Overall, the system aims to optimize AAC's processes and improve outcomes for animals and the community.

## **Motivation**

* The Austin Animal Center (AAC) is the municipal animal shelter serving Austin, Texas. The AAC takes in over 18,000 animals each year, providing them with care and seeking to find them new homes. Proper categorization and organization of these animals is crucial and can be challenging with limited staff and the large intake volume.
* This project aims to optimize AAC's animal categorization processes through automation. By implementing a machine learning system to classify animal photos and data, staff time spent on this task can be significantly reduced. Employees can instead focus on more critical operations like interacting with animals, potential adopters, and the Austin community.

Austin Animal Center runs the largest no-kill animal shelter in the United States, providing shelter to more than 18,000 animals each year and animal protection services to all of Austin and Travis County. The Austin Animal Center is an open-intake facility where lost and surrendered animals from Travis County needing shelter are accepted regardless of age, health, species, or breed. The goal of the Austin Animal Center is to place all adoptable animals in forever homes. (“Austin Animal Center | AustinTexas.gov”)

# **Build Status**



This shows green passing badges for Jupyter Notebook and MongoDB tests on both the master and development branches, along with a pending badge for Python tests on the develop branch. (The badges will be linked to the actual test reports in the future.)

# **Code Style**

## Python

This project follows the Pep 8 style guide for Python code.

We use:

* 4 space indentation
* CamelCase for class names
* snake\_case for variables and functions
* Classes and variables should be descriptively named

Python code is formatted and linted on each commit to maintain consistency.

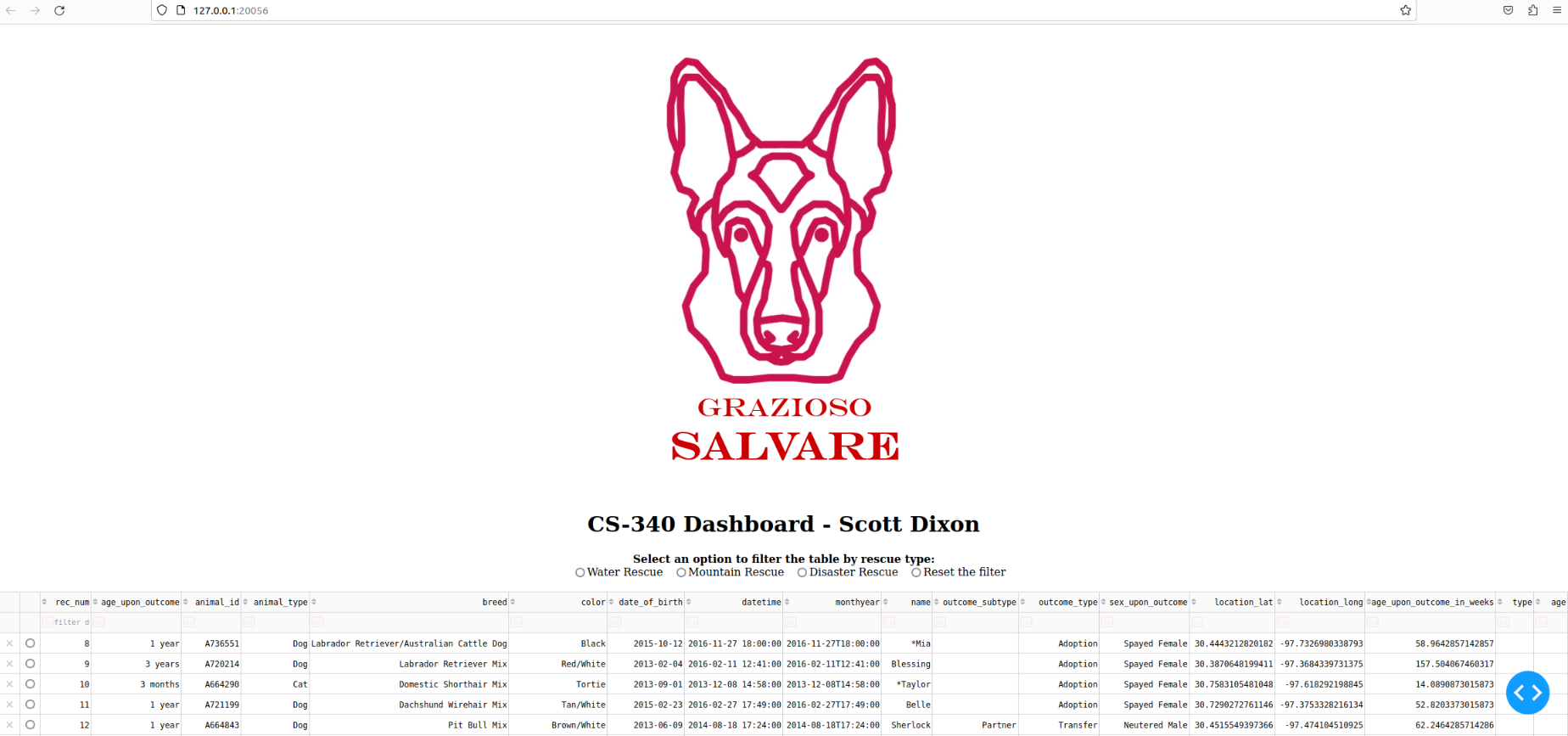
## **Configuration (Links to Documentation.)**

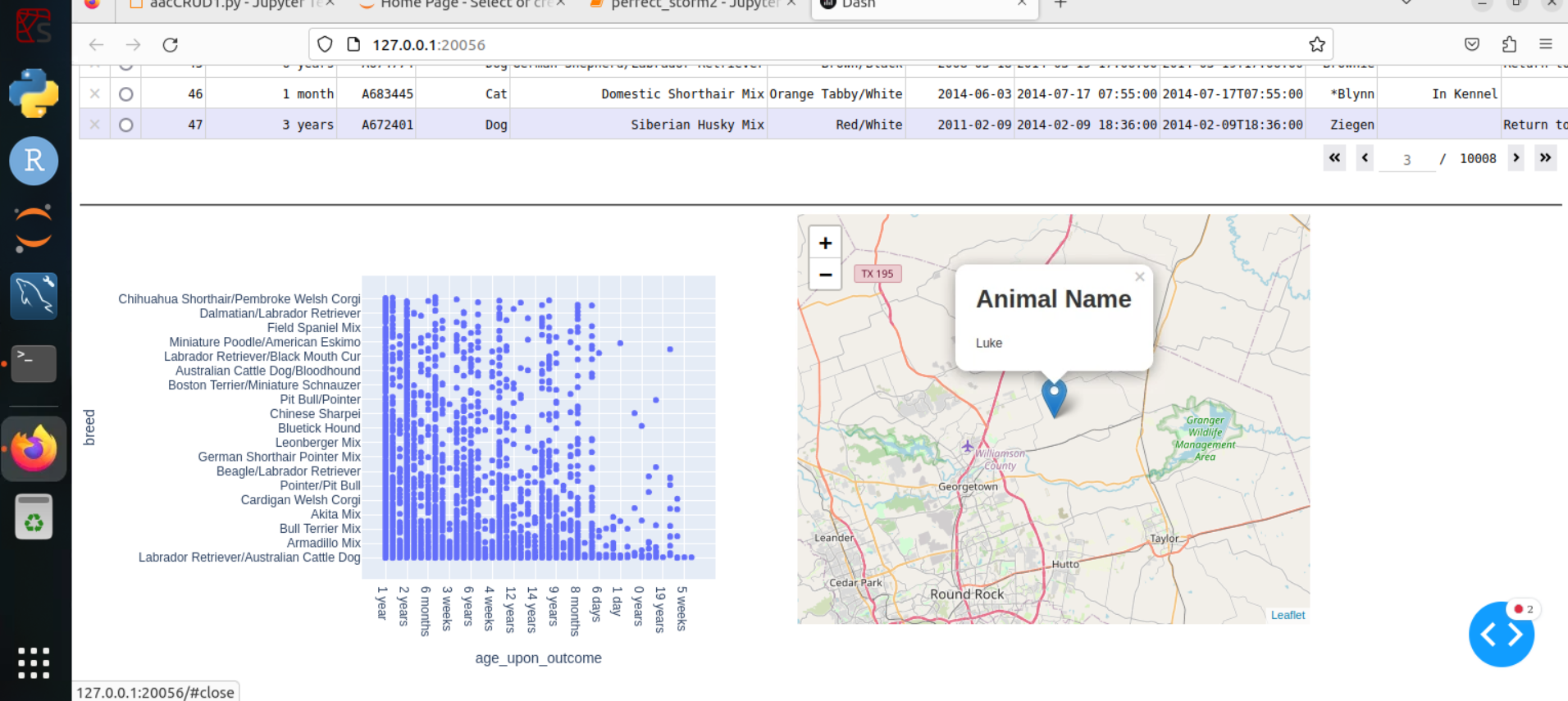
* Code formatting and linting can be configured via <https://pypi.org/project/flake8/>

* See the [Black docs](https://black.readthedocs.io/en/stable/the_black_code_style/current_style.html) and [Flake8 docs](https://flake8.pycqa.org/en/latest/user/configuration.html) for details on customization.

## **Screenshots**

The following screenshots demonstrate the user interface after being linked to the aac.animals API.





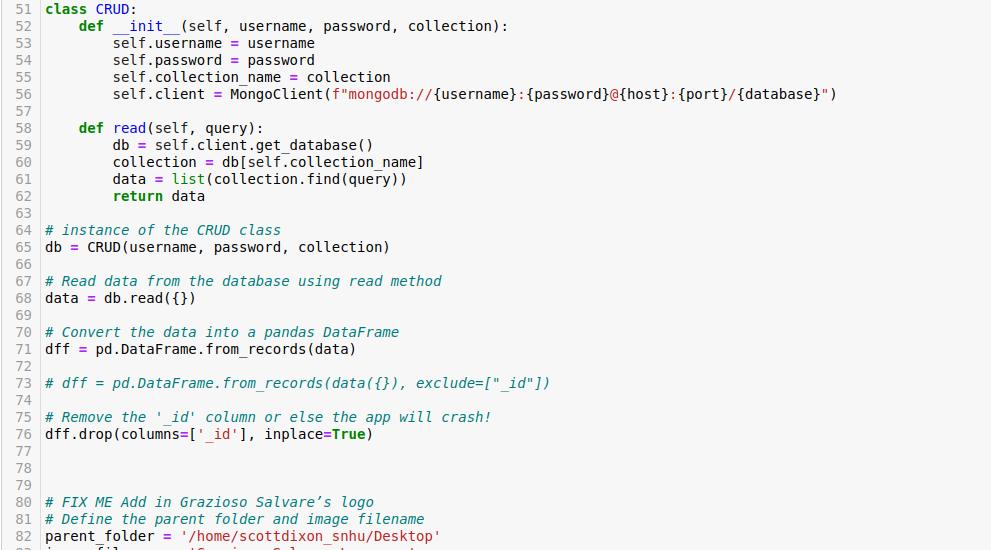
**Built with**

* MongoDB
* Python3
* Jupyter Notebook

## **Features**

* Framework deploys a user-friendly web interface for non-technical users.
* Integrates with external services through a modular API.
* Built with a focus on privacy and data security throughout.
* Packaged in a reusable library that can be easily integrated into any codebase.
* Provides extensive customization options to adapt to diverse use cases.

## **Code Example**



## **Installation**

# **Setting Up the Development Environment**

To start developing with this project, first clone the repository:

git clone <https://github.com/scottdixon-github>

**Install Dependencies**

Install project dependencies by running:

**npm install**

* Dash- Front-end framework
* Plotly - State management
* Apache JMeter - Testing framework

## **API Reference**

# **API Reference**

The Animal Shelter API provides endpoints for managing animals, adoptions, medical records, and more.

Base URL: https://data.austintexas.gov/widgets/wter-evkm?mobile\_redirect=true

## **Animals**

### Get All Animals

**GET /animals**

[ { "id": "12345", "name": "Luna", "species": "Dog", "breed": "Beagle", "age": 3, "status": "adoptable" }, { "id": "67890", "name": "Pepper", "species": "Cat", "breed": "Domestic Shorthair", "age": 2, "status": "adoption pending" } ]

## **Tests**

# **Testing AAC Animal Shelter App with HTTP(S) Test Script Recorder**

1. Install JMeter on your machine.
2. Launch JMeter and add a Thread Group to your Test Plan.
3. Under the Thread Group, add an HTTP(S) Test Script Recorder.
4. In the recorder, set the following:

* **Port**: Some browsers ignore proxy settings for **localhost** or **127.0.0.1**; try using the local hostname or IP (Internet Protocol) instead.
* **Include Pattern.** \*\.py (to only record Python requests)

1. Start the HTTP(S) Test Script Recorder.
2. In your browser, navigate to https://data.austintexas.gov/widgets/wter-evkm?mobile\_redirect=true As you use the site, the recorder will log all PHP requests.
3. Interact with key site functionality:

* View animal listings
* View animal profile
* Create an adoption application
* Log in as staff using Mongosh command: use admin.

## **Challenges**

* Data cleaning and standardization was a difficult aspect of this application when trying to deal with the Apporto Virtual Framework.
* Optimizing aggregations to rapidly summarize large collections was also challenging since the speed of the application was quite slow.
* Implementing filtering and callbacks in Dash. I still have a couple of errors in the callback function that I need to debug.

I overcame these by writing data processing scripts, adding MongoDB indexes, and implementing callbacks.

# **Credits**

This project was inspired by:

* <https://github.com/scottdixon-github> - For the project structure and implementation pattern.
* <https://github.com/scottdixon-github>- For the algorithm used in the analysis module.

Contributors:

* scott.dixon@snhu.edu - Core maintainer, educational developer.

Thanks to the many other [contributors](https://github.com/exampleuser/awesomeproject/graphs/contributors) for this project. Your help is appreciated!

# **License**

This project is licensed under the MIT License - see the file for details.

The MIT License is a permissive open-source license that allows users to freely use, modify, and distribute the software with proper attribution and without warranty.

MIT © Creative Web Developers

References,

<https://www.austintexas.gov/>

*Austin Animal Center | AustinTexas.gov*, https://www.austintexas.gov/department/austin-animal-center-0.

<https://www.austintexas.gov/department/about-austin-animal-center>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Additional Screenshots: 