# CS 340 README Template

### About the Project/Project Title

**Grazioso Salvare Interactive Dashboard – CS-340 Project Two**  
This project is part of my coursework and focuses on creating an interactive data dashboard using Python, Dash, and MongoDB. The goal was to help the Grazioso Salvare organization identify suitable animals for various rescue training operations (Water, Mountain, or Disaster) by building a live dashboard that filters, visualizes, and maps the data from a MongoDB animal shelter dataset.

### Motivation

The main goal of this project was to practice full-stack data visualization by connecting a live database to a responsive UI dashboard. It builds on my earlier CRUD work by giving users the ability to explore animal data through filtering, graphs, and maps—essentially turning static data into an actionable decision-support tool.

### Getting Started

Here’s what I did to get the dashboard working:

1. **MongoDB Setup**  
   A user account aacuser was created with read and write access to the AAC database and animals collection, hosted on Apporto at port 30214.
2. **Importing the Data**  
   Using the mongoimport tool, I loaded the CSV dataset into the MongoDB collection:

mongoimport --username aacuser --password SNHU1234 --host nv-desktop-services.apporto.com --port 30214 --authenticationDatabase admin --db AAC --collection animals --type csv --headerline --file aac\_shelter\_outcome.csv

1. **Python CRUD Module**  
   I reused and refined the animal\_shelter.py module, which contains the AnimalShelter class. It handles Create, Read, Update, and Delete operations from the dashboard and back end.
2. **Dashboard Development**  
   Using Dash, I built projecttwodashboard.ipynb, which includes:

* Radio buttons to filter animals by training type
* A dynamic data table
* A pie chart for breed distribution
* A map showing animal location based on selection
* Debug and conditional formatting features for user feedback

### Installation

These tools were used in the project:

* **Python 3**
* **Dash** and **JupyterDash** – for building the interactive dashboard
* **pandas** – for data manipulation
* **dash\_leaflet** – for map rendering
* **pymongo** – for database access
* **MongoDB** (pre-installed in Apporto)
* **Jupyter Notebook** – to test and run the application visually

Install via pip (if running locally):

pip install dash dash-leaflet jupyter-dash pymongo pandas

### Usage

Example of how the dashboard loads data and filters it:

# Load all data from the animals collection

df = pd.DataFrame.from\_records(db.read({}))

# Apply filtering based on radio button selection

query = {

"breed": {"$in": ["Labrador Retriever", "Newfoundland"]},

"age\_upon\_outcome\_in\_weeks": {"$lte": 104}

}

filtered = db.read(query)

Users can:

* View all animals in the database
* Apply filters by rescue type
* Click a row to show a map location
* See a chart update live with breed distribution

### Code Example

Example from the interactive map section:

return [

dl.Map(style={'width': '1000px', 'height': '500px'},

center=[lat, lon], zoom=10, children=[

dl.TileLayer(id="base-layer-id"),

dl.Marker(position=[lat, lon], children=[

dl.Tooltip(breed),

dl.Popup([

html.H1("Animal Name"),

html.P(name)

])

])

])

]

This lets the user click a row and immediately see that animal’s breed and location on the map.

### Tests

I verified all core features in Jupyter:

* CRUD operations tested using animal\_shelter.py
* Filtering logic tested via the radio buttons
* Debug console showed selected records
* Map and chart updated in real time based on DataTable row clicks

Everything worked as expected in the Apporto environment.

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

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

* Dashboard with logo and full dataset
* Filter applied (e.g., “Water Rescue”)
* Map + chart view after row selection
* Debug output showing sample record

### Roadmap/Features (Optional)

Ideas for expanding this project include:

* Adding search by animal name or age
* Enabling CSV export of filtered results
* Deploying the dashboard as a standalone web app (Heroku or Flask server)
* Building a login-protected admin panel for CRUD actions

### Contact

**Niaz Khan**  
Course: CS-340 Client-Server Development