README – Grazioso Salvare Dashboard

In order to transform the MongoDB-based CRUD system from Project One into a fully interactive dashboard, the Grazioso Salvare Dashboard was created for CS-340 (Project Two). Users at Grazioso Salvare can browse and filter animal data pertinent to rescue-training decisions using the application, which connects to the Austin Animal Center outcomes database hosted in MongoDB Atlas. Pandas handles data transformation, and the Dash and Plotly Express frameworks are used to write the dashboard fully in Python. The AnimalShelter class, which was initially developed in Project One, is used for all CRUD activities.

When the application is run, it retrieves the animal outcomes collection and uses the student's credentials to create a secure connection to MongoDB. Interactive controls, such as drop-down filters for Animal Type and Outcome Type, a text box for Breed search, and a multi-select menu for Sex Upon Outcome, are then rendered by a Dash online application. A paginated, sortable DataTable with native column filtering displays the filtered data. An instant overview of adoption, transfer, and other outcome trends is given by the dashboard's dynamically updated bar chart that visualizes the Outcome Type Distribution. Real-time communication between all filters and MongoDB ensures that only records that meet the chosen criteria are retrieved.

The system can be used as a standalone Dash server (python3 ProjectTwoDashboard\_FIXED\_v3.py) or inside of Jupyter (app.run\_server(mode="inline")). The public URL needed to access the dashboard is generated by Project → Box URL → Open Port 8050, which users can use to launch the application in Codio from the terminal. After a successful connection, the application prints a confirmation message that reads, "✅ Connected and fetched initial data." Users should run inline in Jupyter Lab or open the proxied Codio URL (which ends in app.codio.io) if the page times out on 0.0.0.0:8050.

The dashboard automatically adjusts to various AnimalShelter class versions. To ensure that the DataTable never crashes, it checks to see if the constructor takes credentials, locates the appropriate "read-like" function (read, retrieve, search, etc.), and normalizes missing columns. This adaptability guarantees that even if the CRUD module changes between environments or projects, the code will still function properly. Additionally, the data is verified to have necessary columns like outcome\_type, animal\_type, and \_id.

Numerous difficulties arose throughout development. The "Open Port 8050" feature was used since direct access to 0.0.0.0:8050 was blocked by Codio's network sandbox. An adaptive import procedure was used to fix constructor mismatches between assignments that the CRUD module also produced. Pandas preprocessing and type casting were used to address data formatting discrepancies in MongoDB. A stable, cross-compatible dashboard environment was guaranteed by each of these fixes.

All things considered, the finished dashboard satisfies every condition of the rubric: it has modular code architecture, good documentation, an interactive table and chart, live filtering, and a MongoDB connection. Other features that could be added to the project include mapping animal locations by latitude and longitude, exporting filtered data to CSV, and putting authentication in place for restricted access. An effective tool for Grazioso Salvare's data-driven decision making is provided by the combination of MongoDB's flexible querying and Dash's web interactivity.

Screenshot: