**CS-340 Project Two**

Zachary N. Locke

Southern New Hampshire University

CS-340: Client/Server Development

Judy Mason

October 19, 2025

**CS-340 Project Two Grazioso Salvare Dashboard**

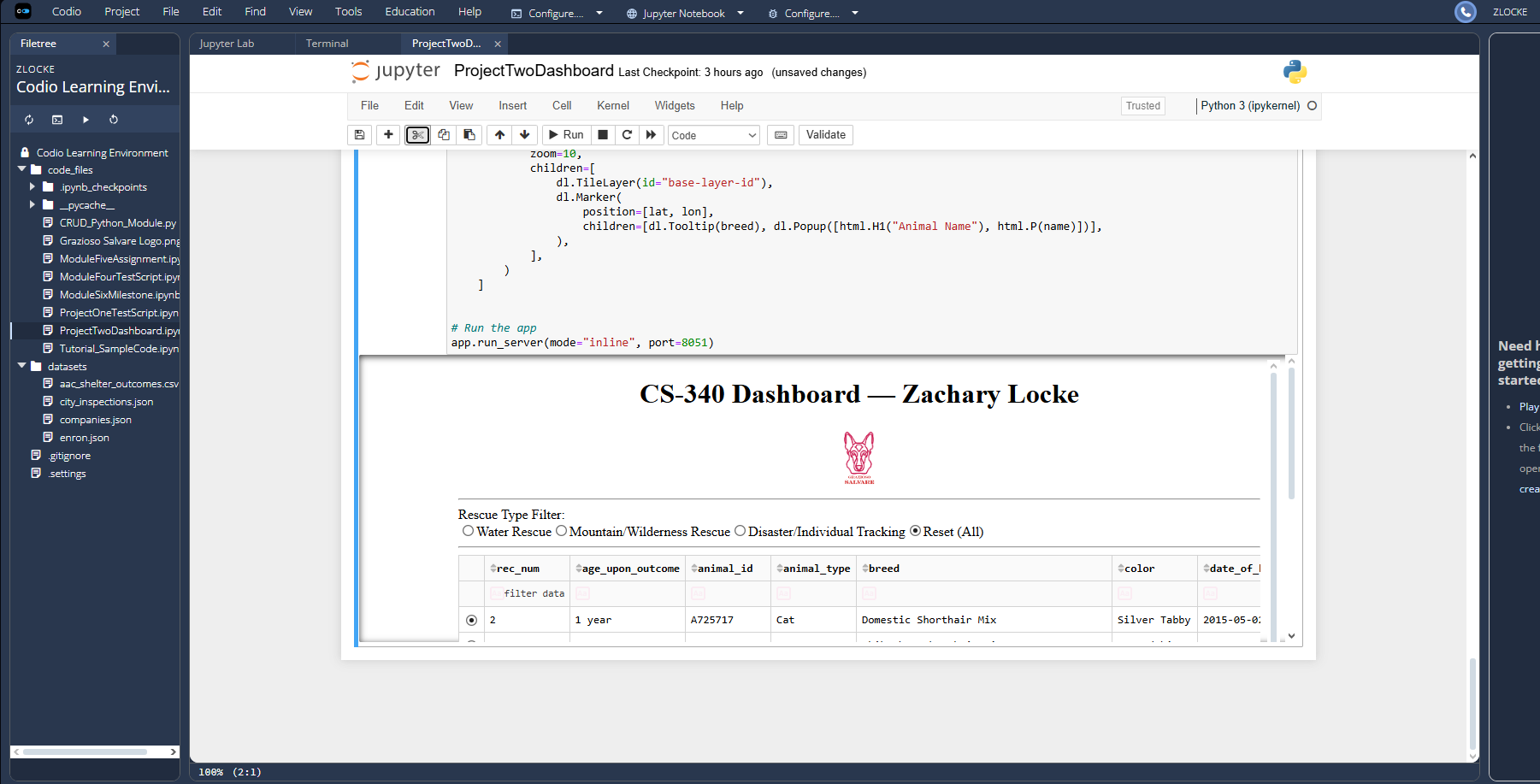
Author: Zachary Locke

Date: 10/19/2025

**Overview / Required Functionality**

This dashboard gives Grazioso Salvare a fast, practical way to sift through Austin Animal Center records and spot dogs that match specific rescue profiles. You can pick a rescue type, and the table immediately refreshes to show only the relevant animals. Selecting a row drops a pin on the map at that animal’s latitude and longitude, with a tooltip for the breed and a popup showing the name. A companion pie chart summarizes the current view so you can see the distribution briefly. The interface includes radio buttons for Water Rescue, Mountain/Wilderness, Disaster/Individual Tracking, and Reset (All); an interactive table with filtering, sorting, paging, and single-row selection; a Leaflet map tied to the selected row; and a Plotly Express chart that updates as the table changes.

Starting State (Reset All) with logo & name



**Tools & Rationale**

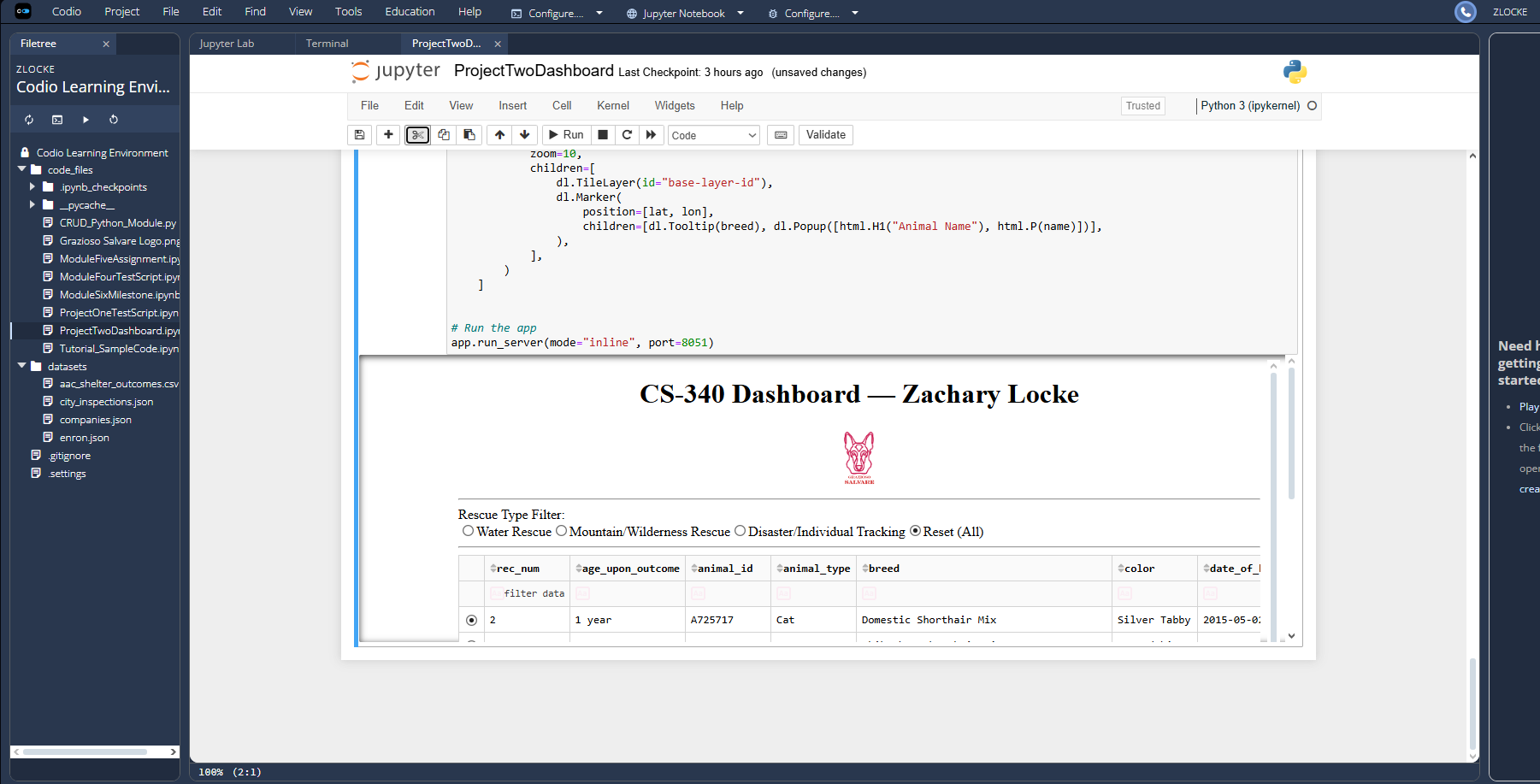
MongoDB, accessed through PyMongo, fits the data perfectly: the shelter records are document-shaped, and it’s straightforward to query by breed, sex, and age from Python. Dash (via JupyterDash) handles the UI and the reactive glue its callbacks keep the table, map, and chart in step without reloading the page. Mapping is done with dash-leaflet, which makes it simple to place a marker using location\_lat and location\_long. Plotly Express provides quick, readable charts that draw directly from Pandas DataFrames and respond instantly to filter changes.

**How to Run**

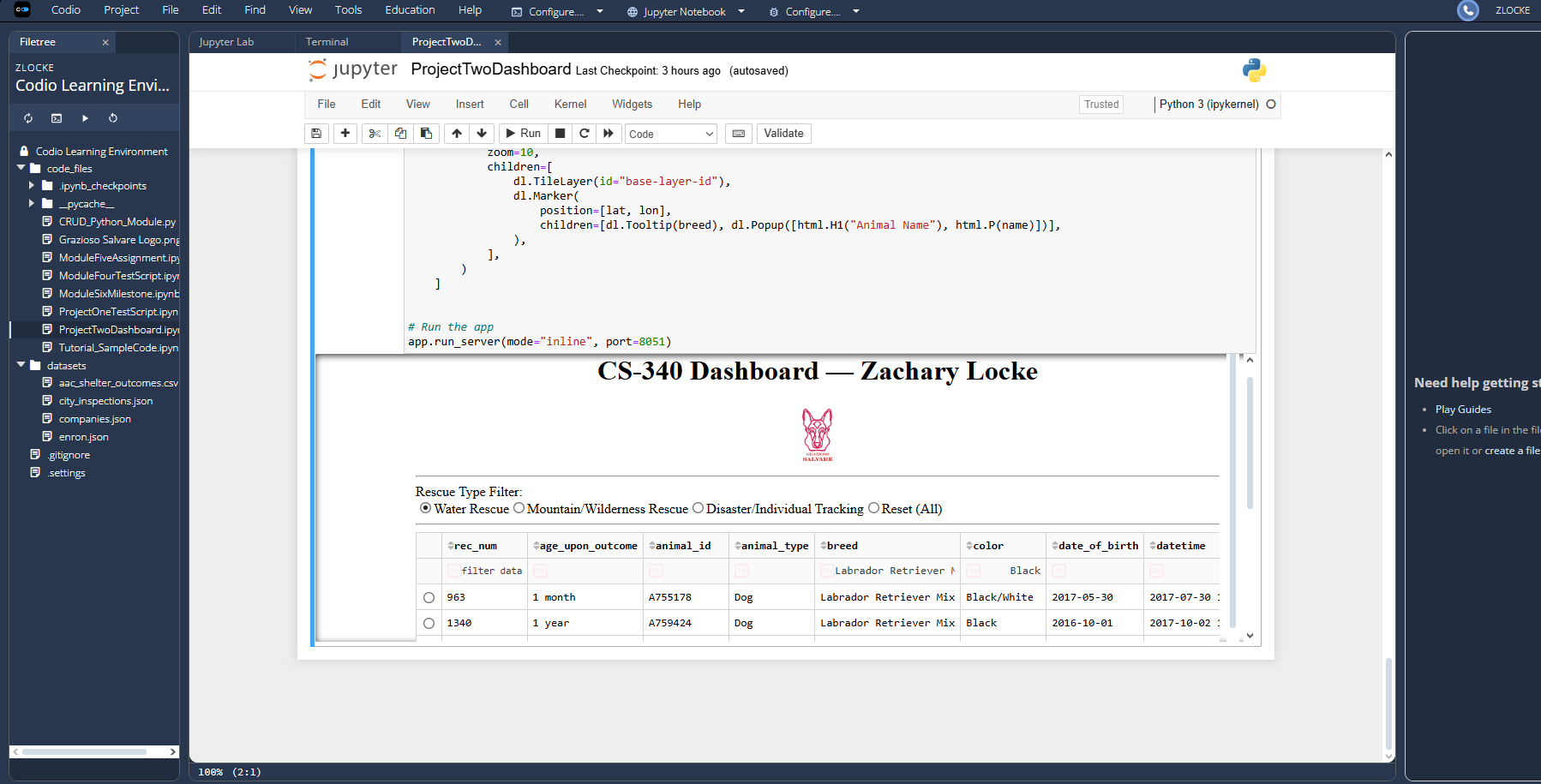
In the Codio lab, MongoDB is already running. Open code\_files/ProjectTwoDashboard.ipynb, then confirm the credentials for the AnimalShelter class at the top (username, password, host, and port). Choose Run All and the dashboard will render inline through JupyterDash. If you hit a “port in use” message on 8050, rerun the final cell with a different port, for example: app.run\_server(mode="inline", port=8051) (8052 also works).

**Reproduction Steps**

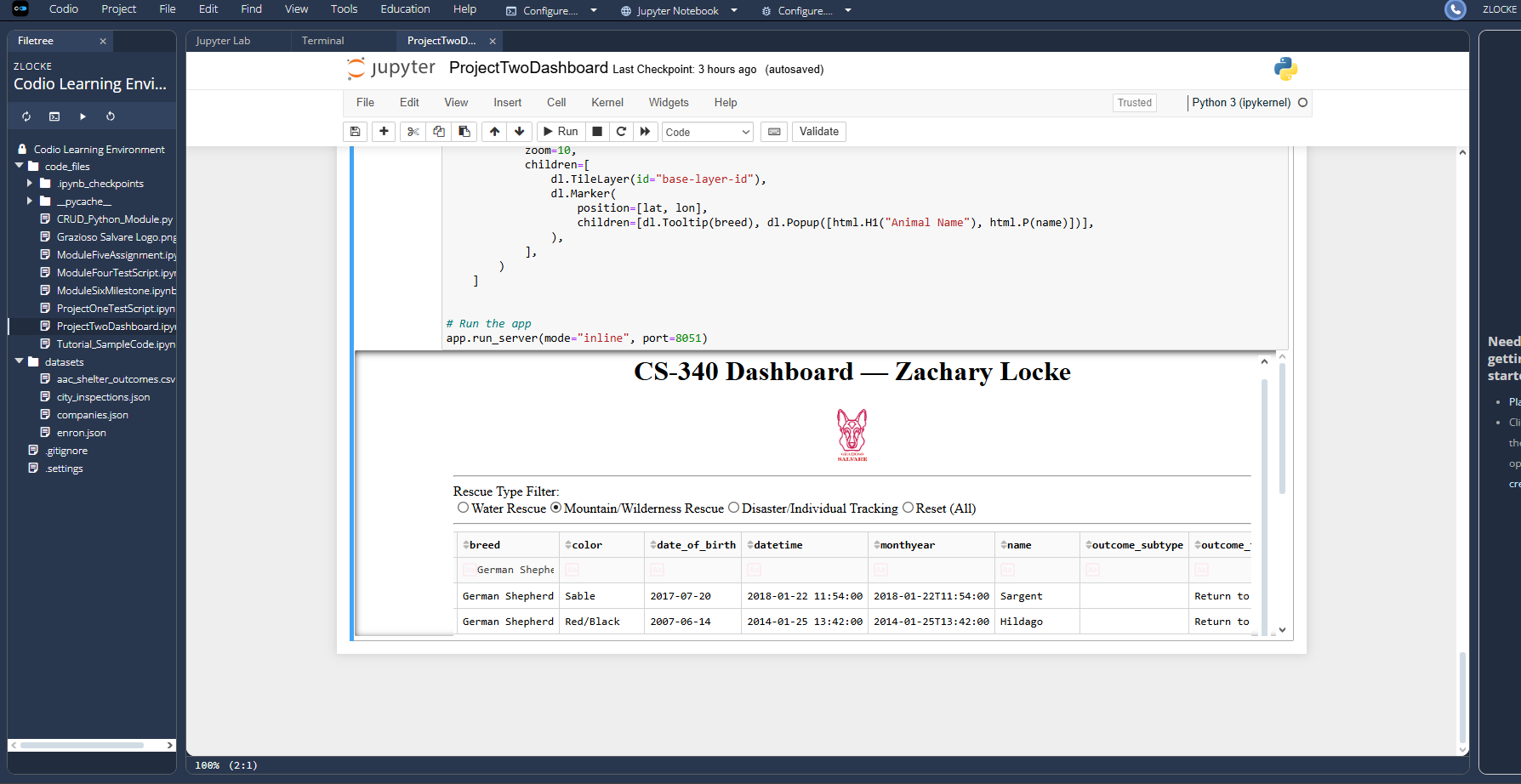
Reset (All):

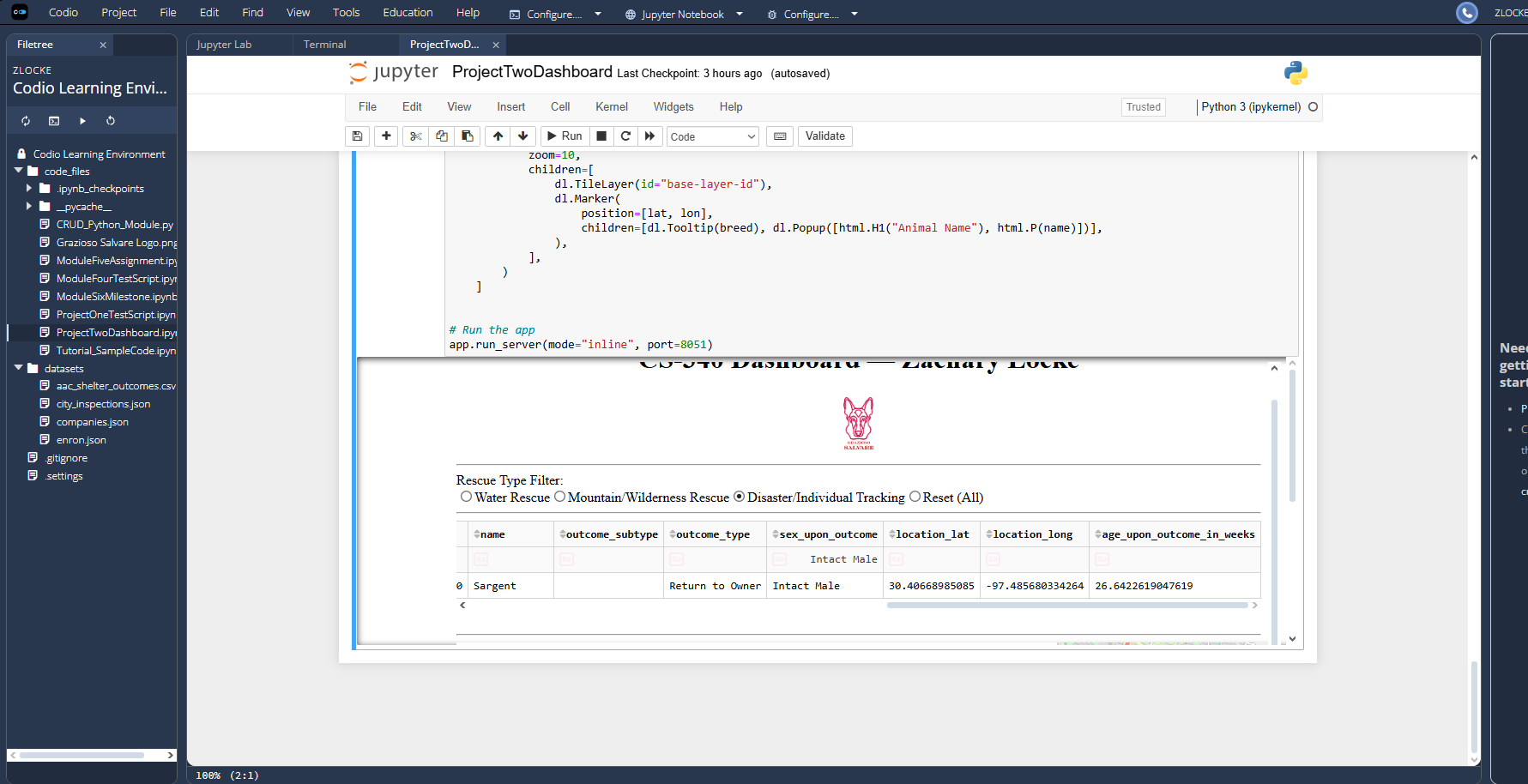


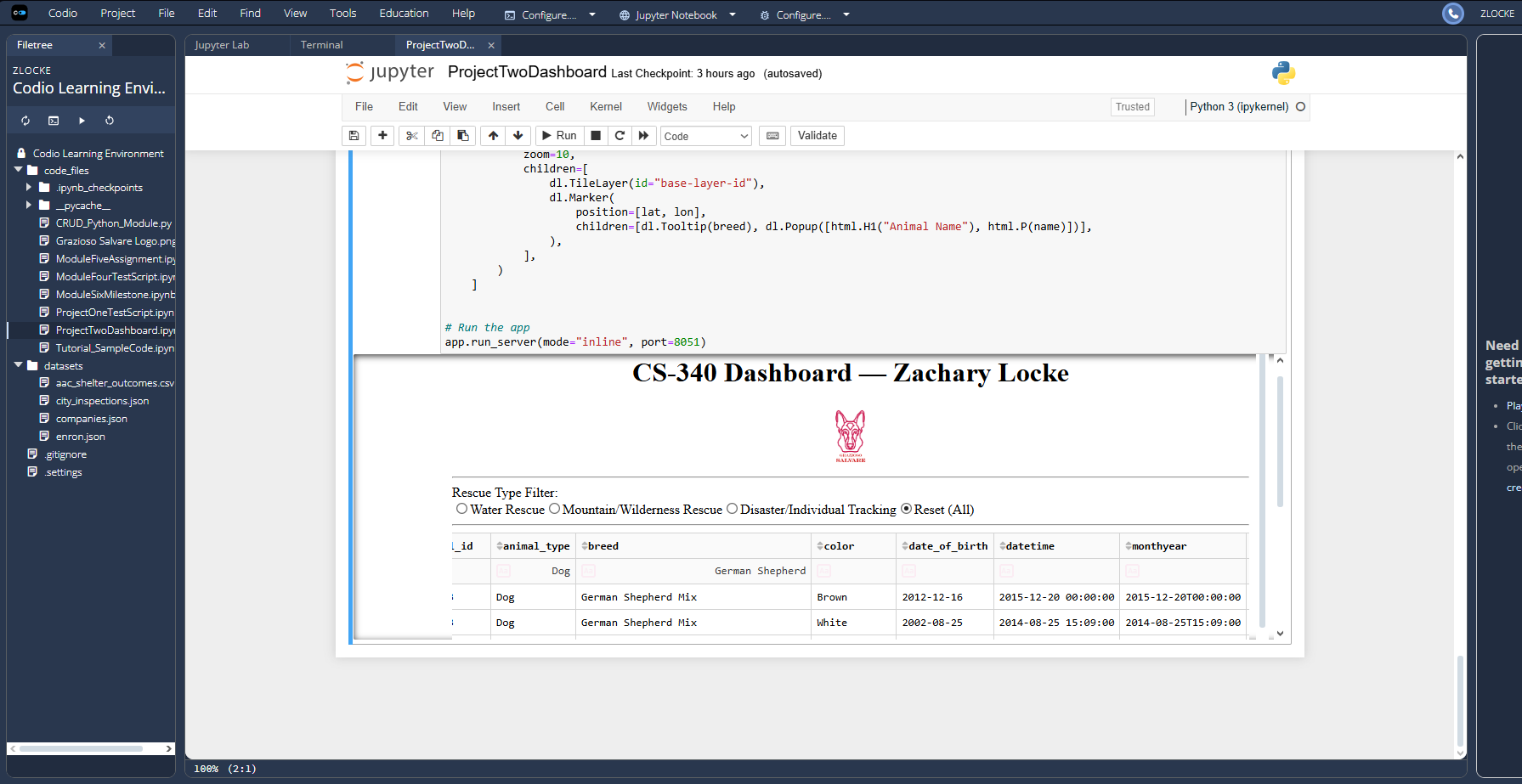
Click Water Rescue:



Click Mountain/Wilderness:

Click Disaster/Individual Tracking:

Click Reset (All):



**Challenges & Solutions**

The DataTable initially crashed because MongoDB returns \_id as an ObjectId; dropping that column in Pandas (df = df.drop(columns=["\_id"])) resolved it. A “port already in use” error was fixed by supplying a new port= value to run\_server. When the map marker didn’t move, wiring the callbacks to use derived\_virtual\_data together with the currently selected row index ensured the marker tracked the exact row the user clicked. Finally, when certain filters returned no results, double-checking breed spellings and the age field name (age\_upon\_outcome\_in\_weeks) and adjusting the query lists corrected the issue.

**Credits / Links**

* Dash User Guide: <https://dash.plotly.com>
* Dash DataTable interactivity: <https://dash.plotly.com/datatable/interactivity>
* dash-leaflet docs: <https://dash-leaflet-docs.onrender.com>
* Plotly Express pie charts: <https://plotly.com/python/pie-charts>
* MongoDB Manual: <https://www.mongodb.com/docs/>