Jean Agnus  
CS 499 Milestone 4 Narrative  
Professor: Nembhard  
Date: June 13, 2025

**Artifact Description**

The chosen artifact for comparison is the Grazioso Salvare Animal Dashboard, a Python-written web application made with the Dash Framework and MongoDB. As a previous assignment from CS 340 Advanced Programming Concepts, this dashboard provided a simplistic display for animal shelter data but had little functionality and no discernible backend architecture.

**Original Limitations:**

* It only supported **read** operations (viewing records from the MongoDB database).
* There were **no forms or UI elements** to allow users to **add, edit, or delete** data.
* There was **no input validation**, making it vulnerable to faulty inputs or security risks.
* All logic (UI, database queries, and callbacks) was combined in a **single script**, making the system hard to extend or test.

**Enhancements Implemented:**

To improve the application and bring it to a professional level, I added:

* Full **CRUD functionality**: Users can now Create, Read, Update, and Delete animal records directly from the UI.
* A separate module (crud\_module.py) to handle all MongoDB interactions, improving **modularity** and **separation of concerns**.
* Proper **input validation** and **error handling** to ensure secure and reliable database updates.
* Enhanced the UI with **Dash input components** and **interactive messages** confirming database actions.
* Modularized code to support future features like filtering and visualizations.

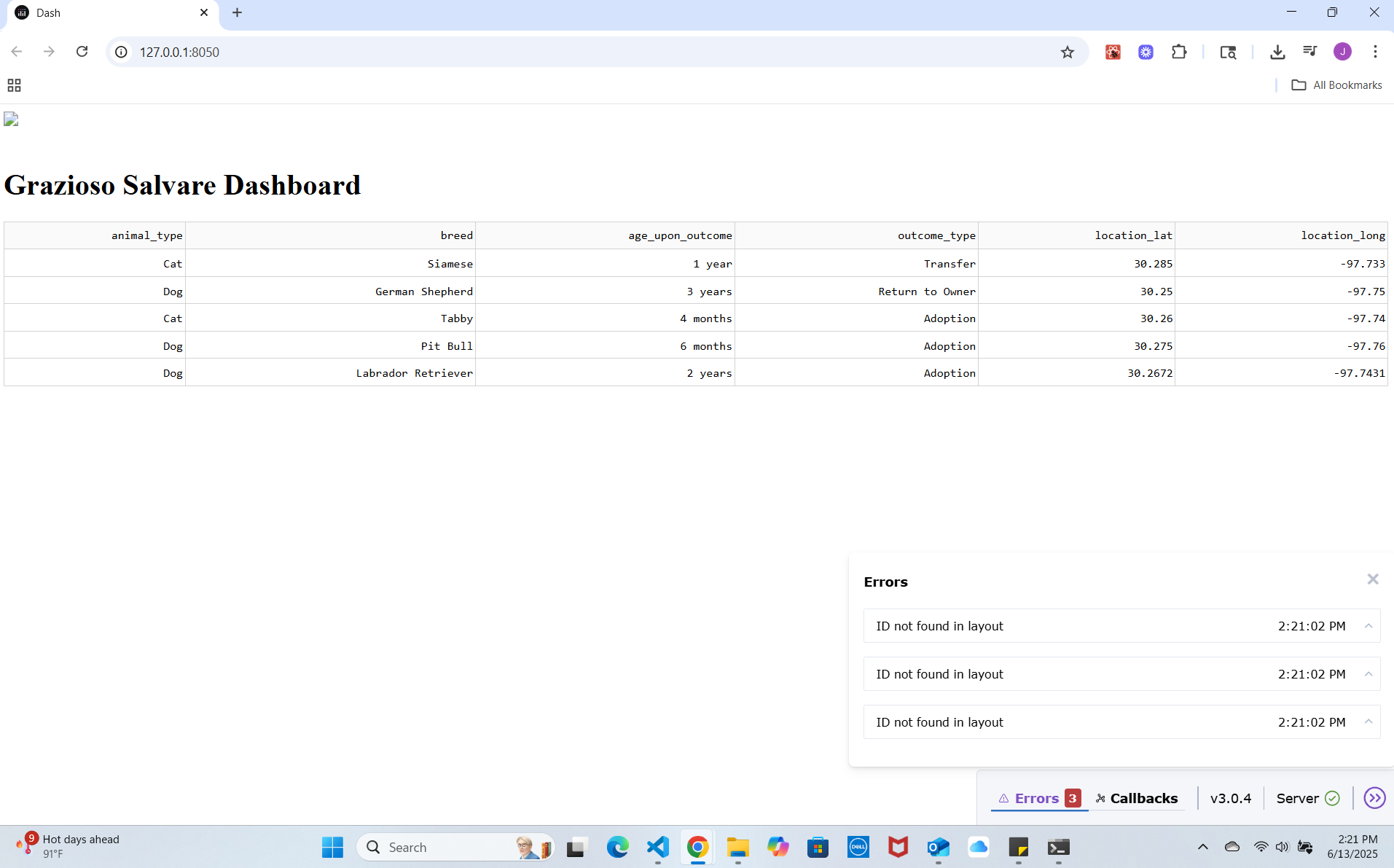
**Before vs. After:**

* **Before**: Static data table pulled from MongoDB. No user input or live updates.
* **After**: Dynamic dashboard with add, edit, and delete inputs, real-time updates to the data table, and input validation to prevent bad or insecure data.

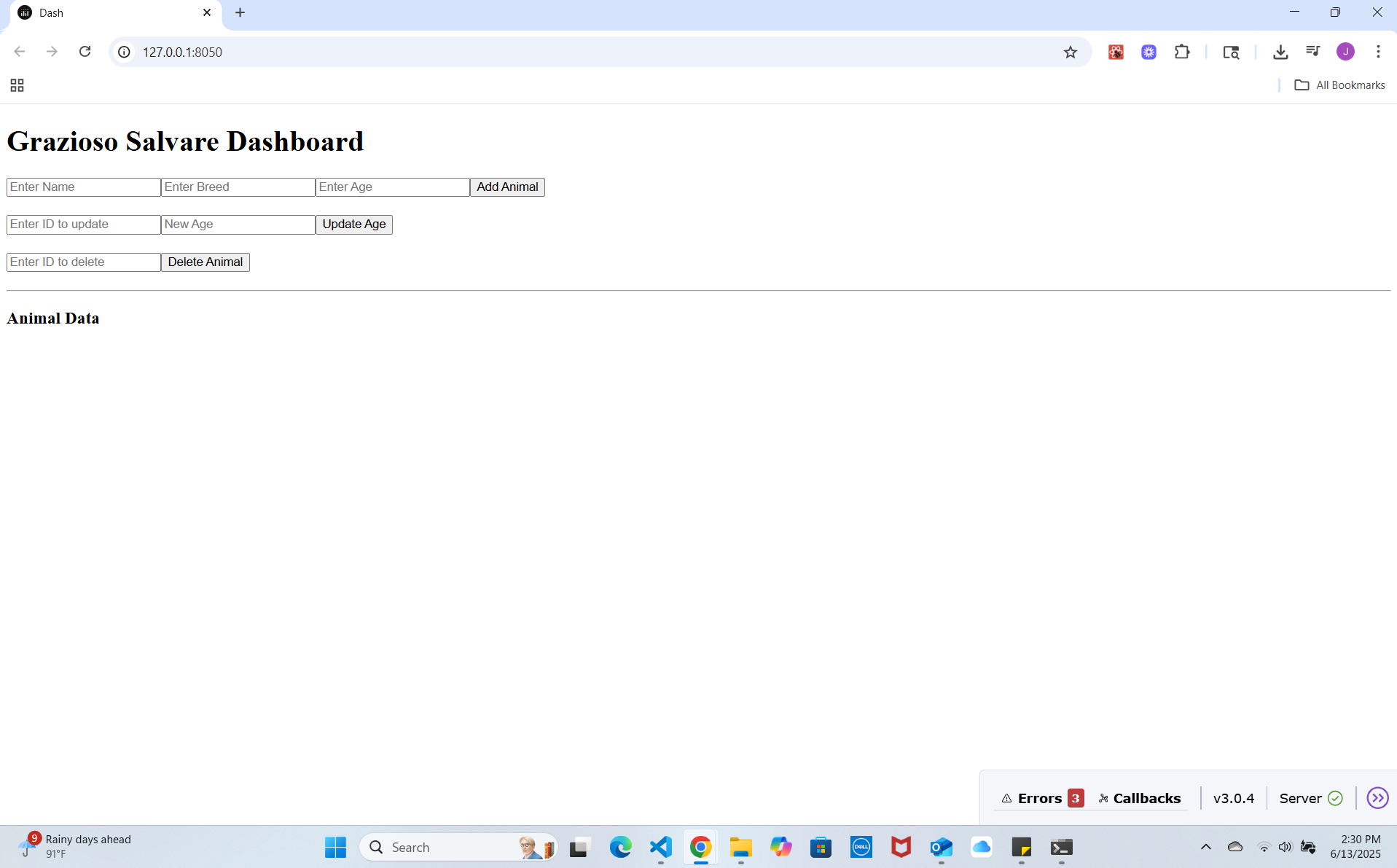
Note: In my final ePortfolio and code review video, I’ve included side-by-side visuals to clearly show these changes in action.

1. **Homepage UI – Before vs After**

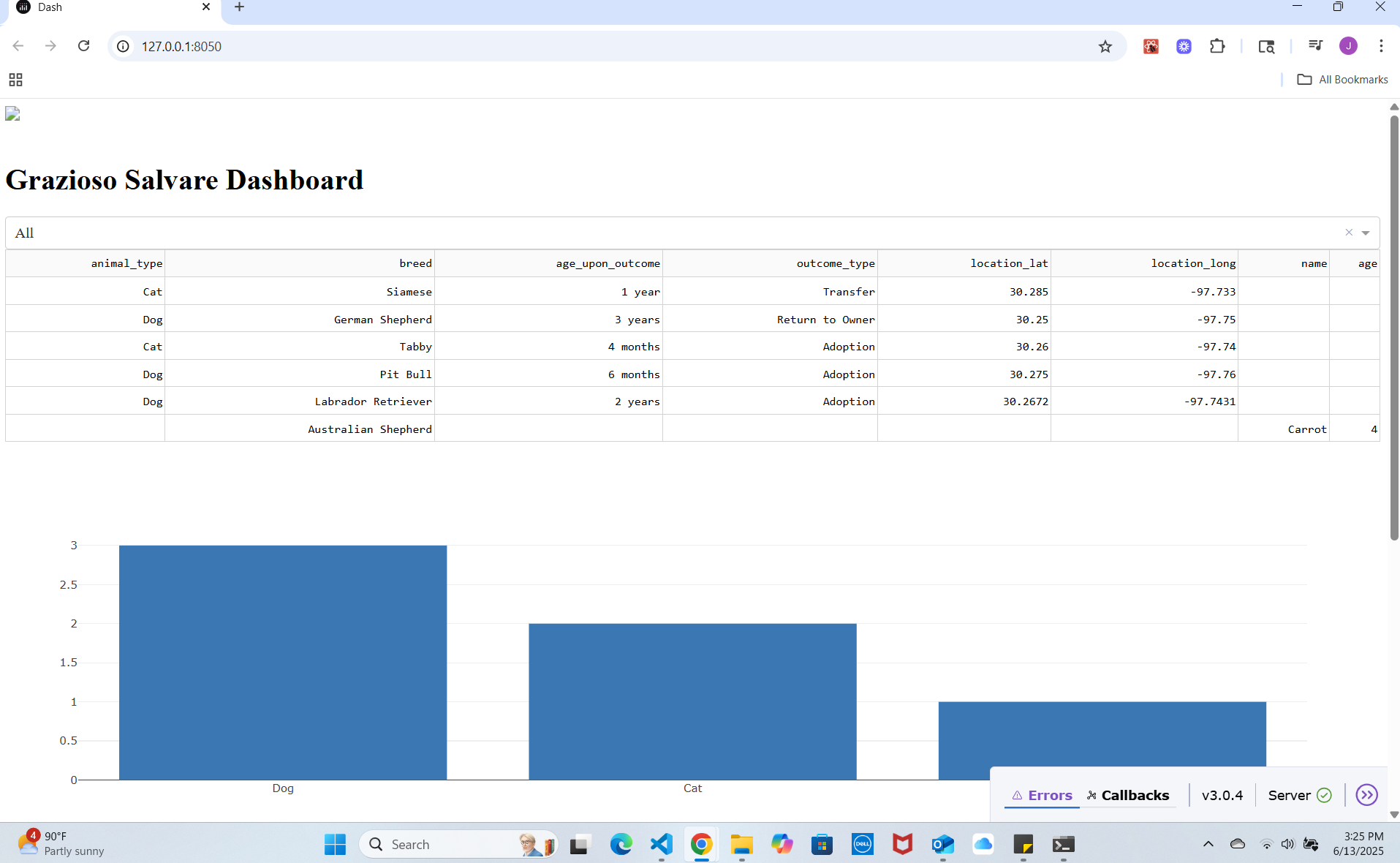
Original dashboard only displayed static table data with no input fields or CRUD features.

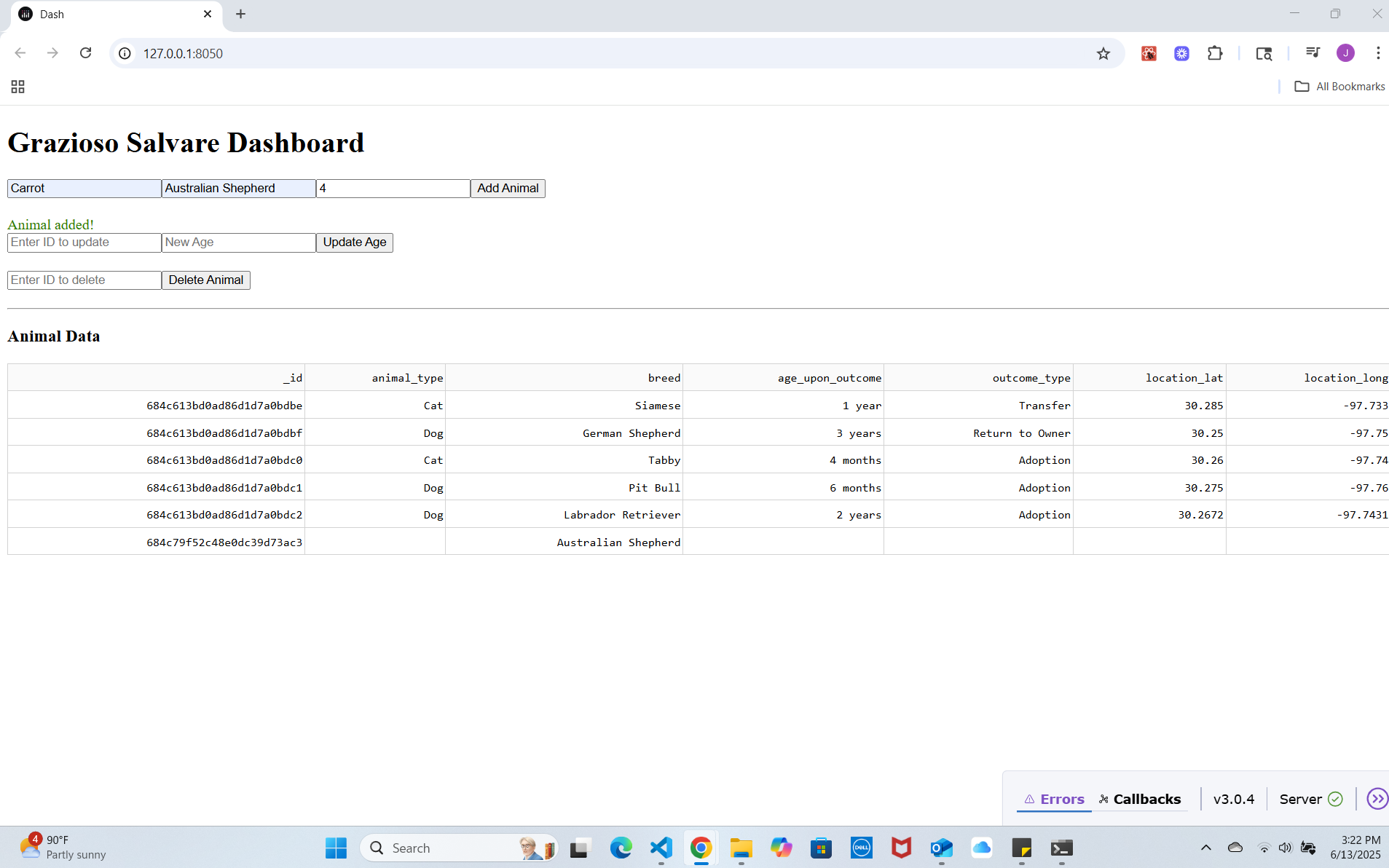


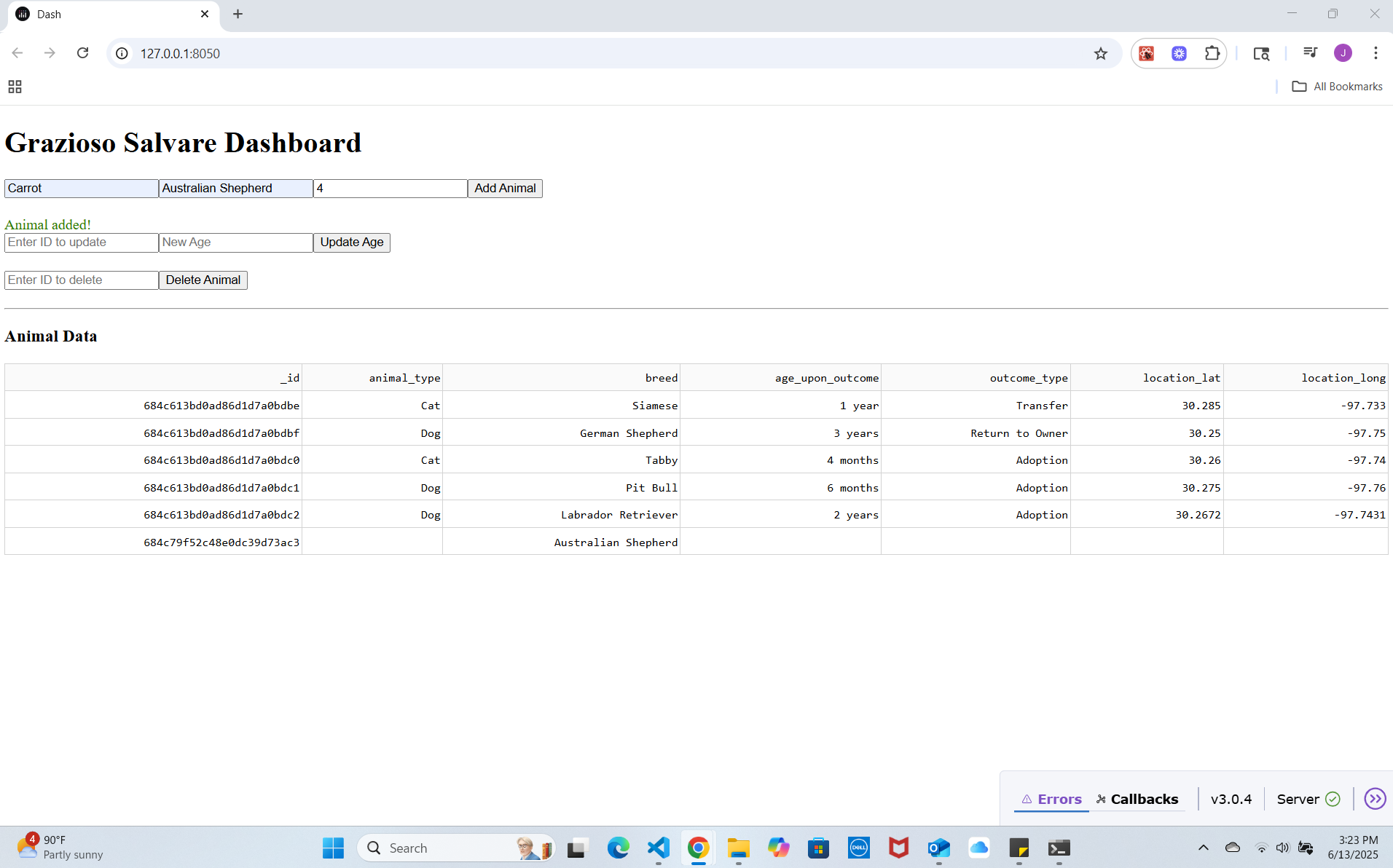
**After UI:** Enhanced dashboard interface now includes input fields and buttons to add, update, and delete animal records.



1. **Add Animal** – Before vs After:  
   Original version had no way to add new animals through the interface.

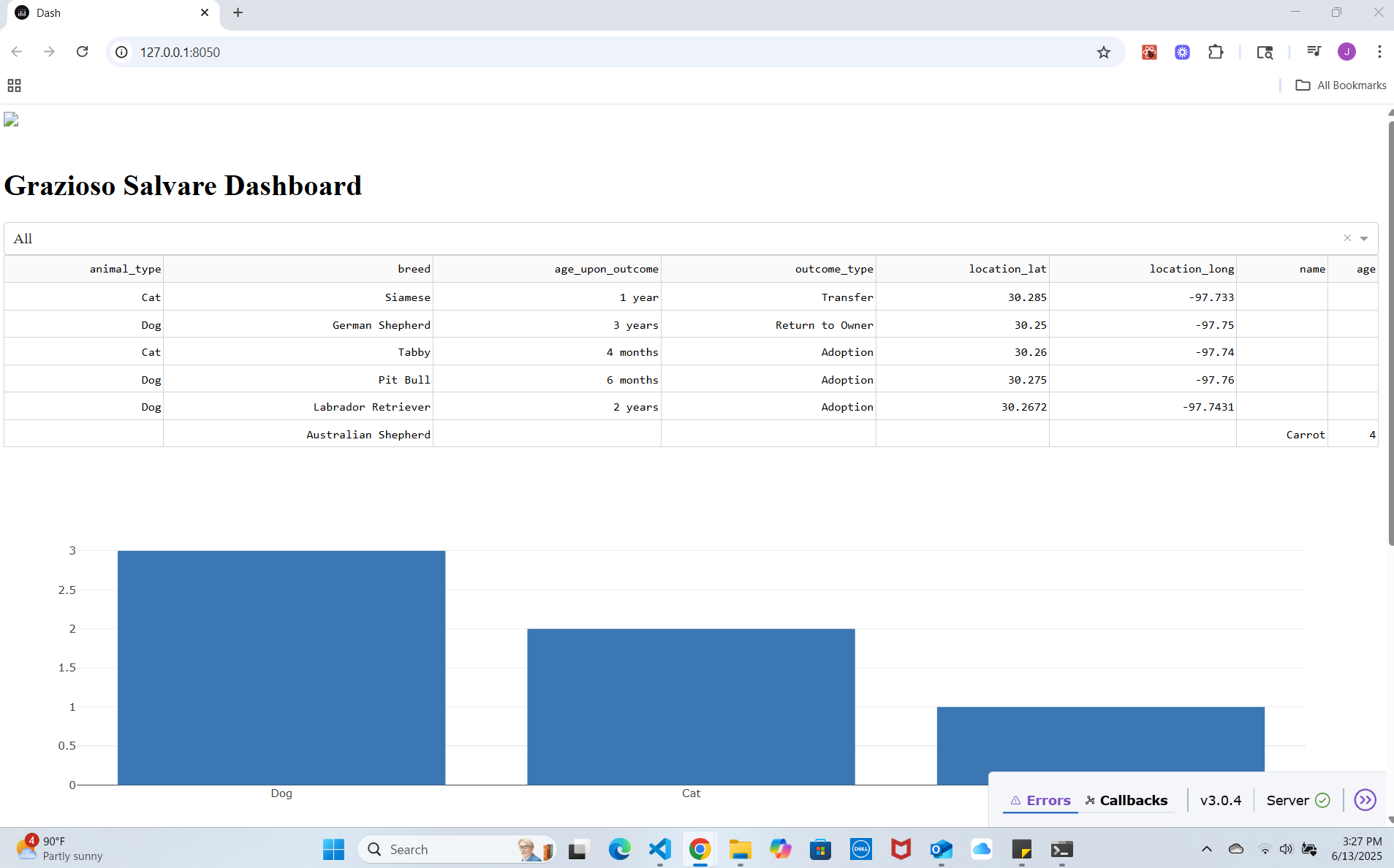


**After add form**: New "Add Animal" form allows users to enter name, breed, and age before submission.  


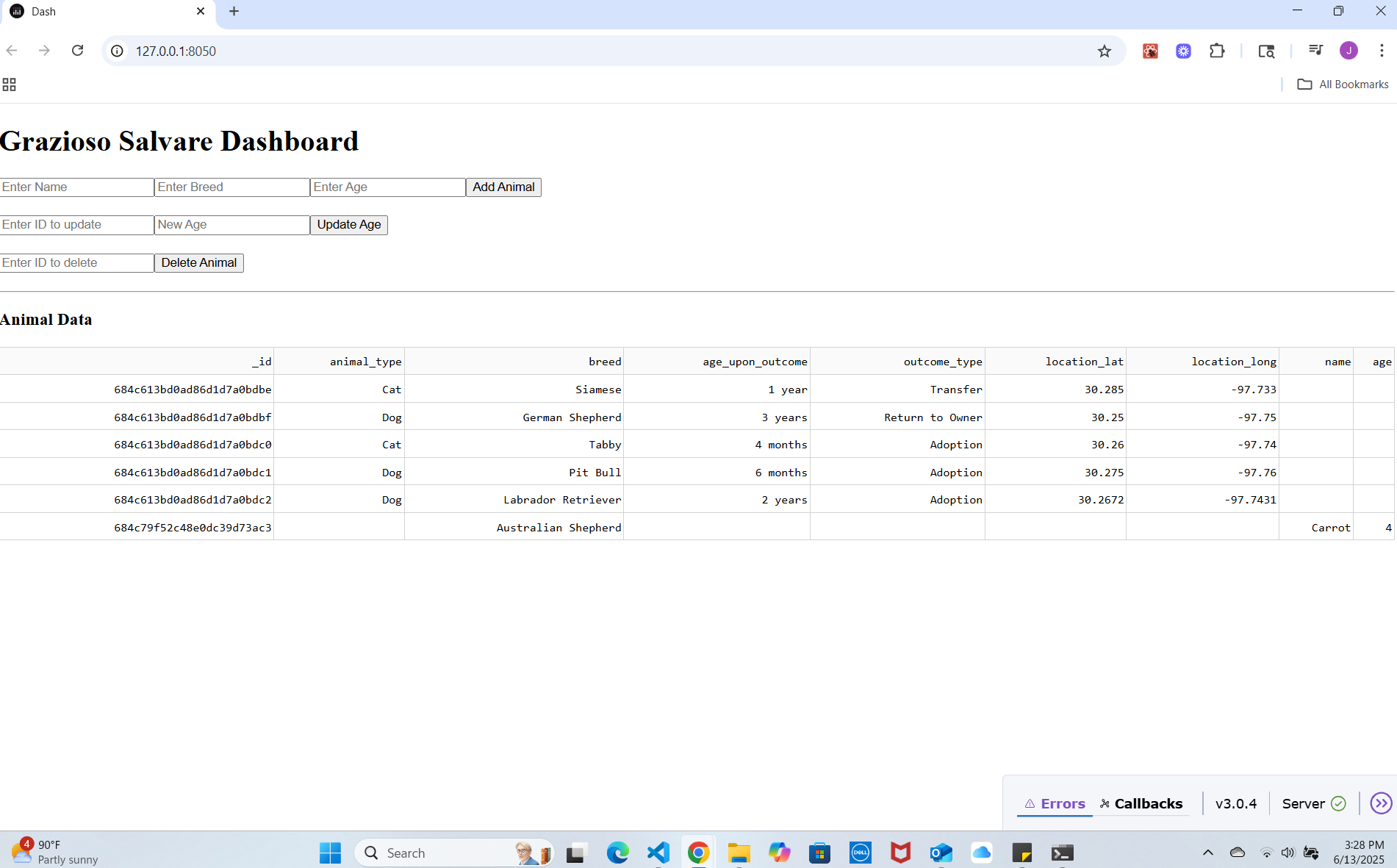
**After add success:** Successful addition confirmed with a green message and table updated with the new record.  


1. **Update/Delete** - Before vs After:

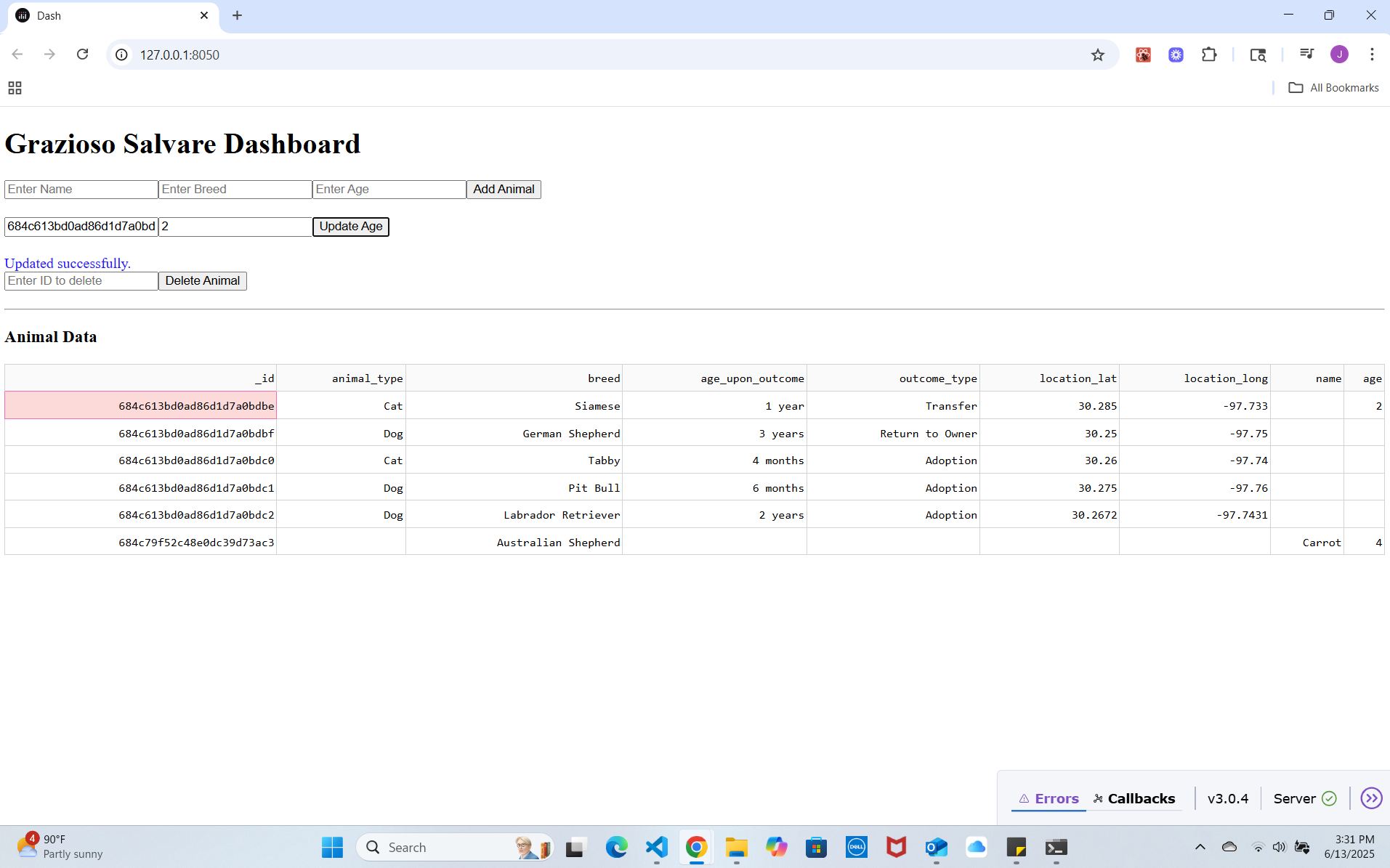
Before update: No editing or deletion options were available in the original dashboard.



**After update form:** Update form where the user can enter an animal's ID and a new age.

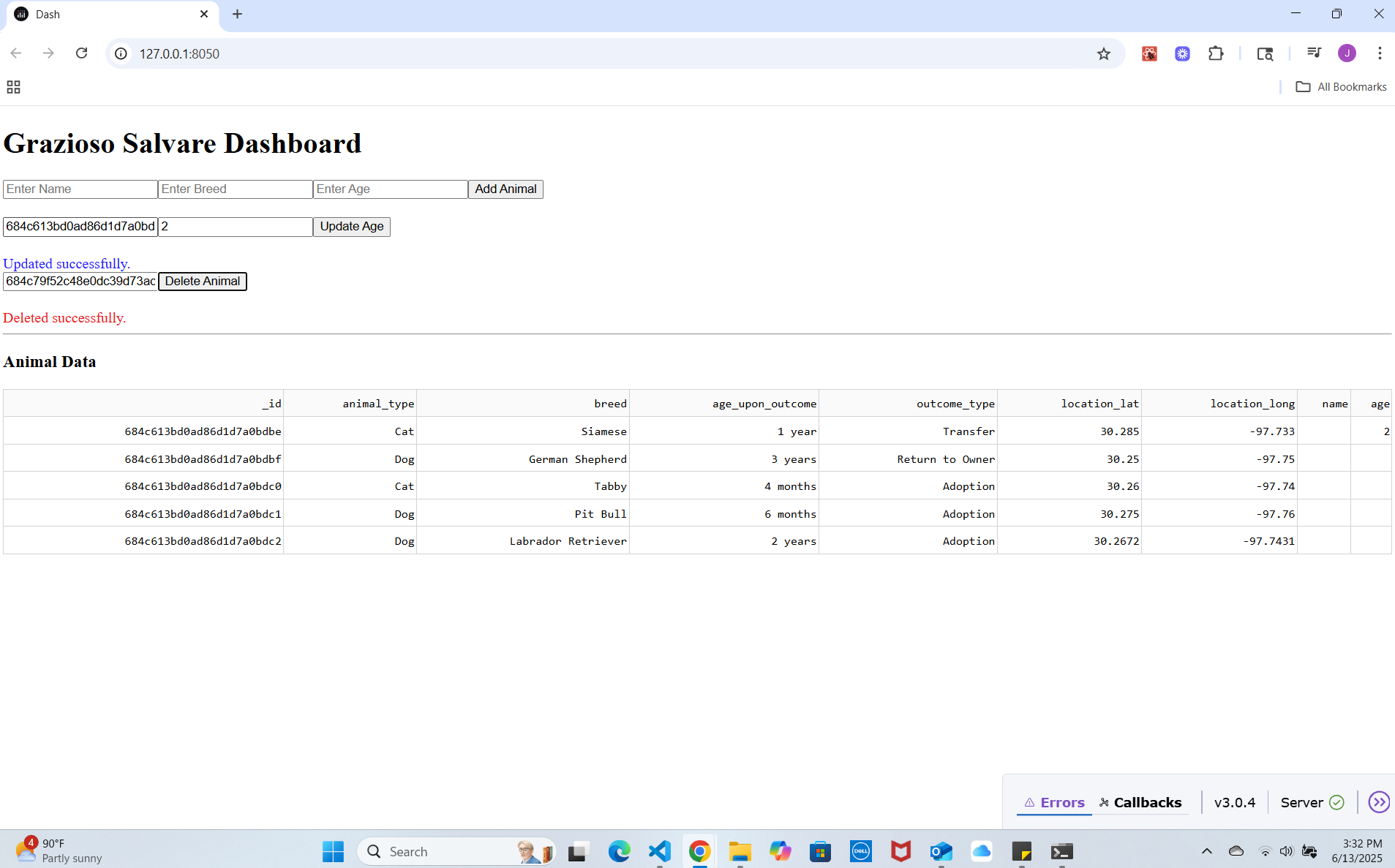


**After update success:** Confirmation message appears after a successful update, and the table reflects the change.



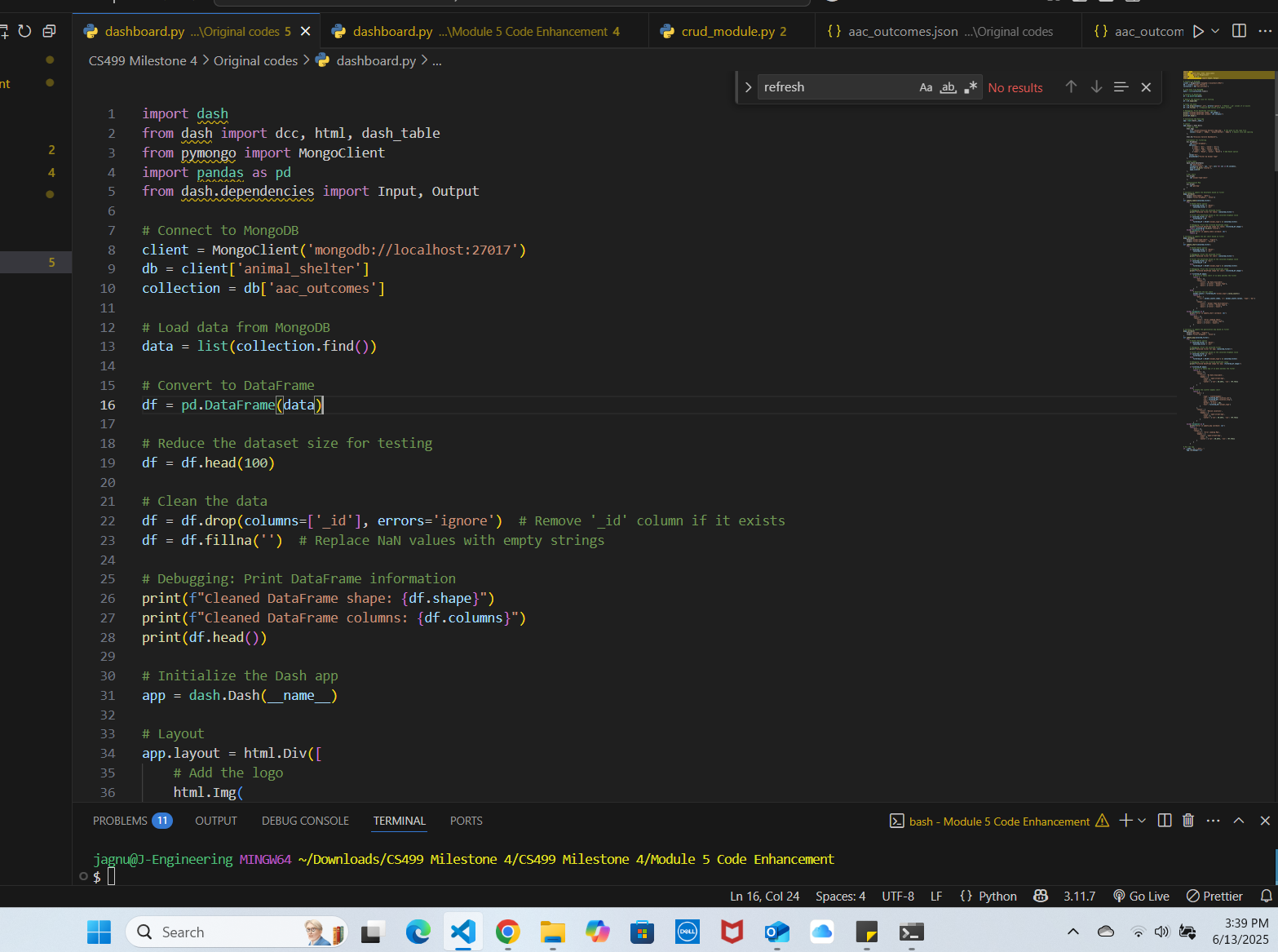
**After deleting form**: Delete form where the user can enter an animal’s ID to remove it.

Confirmation message after deletion; table is refreshed with the record removed.

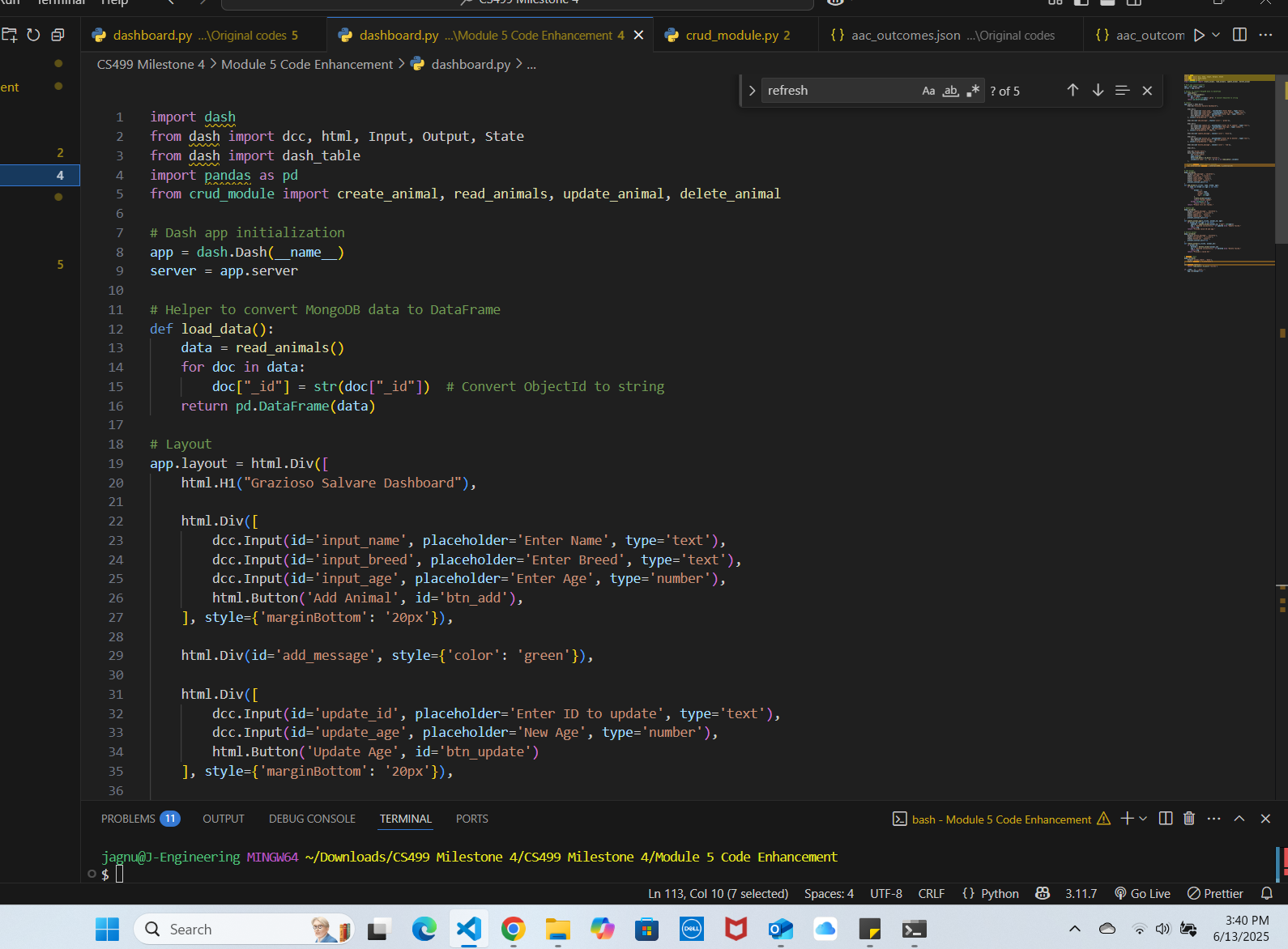


1. **Code Refactor - Before vs After**

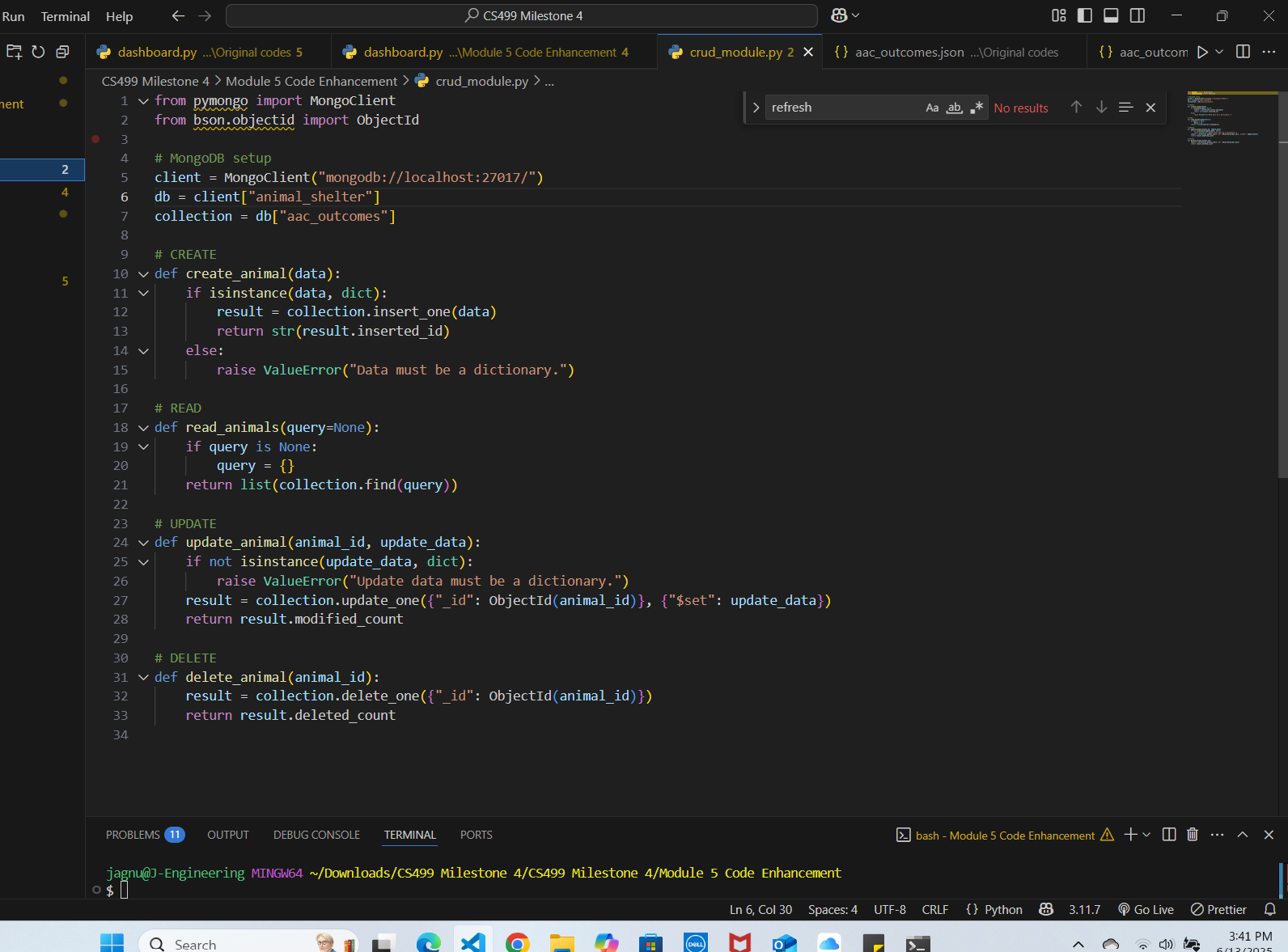
Before code:



**After code dashboard:**



**After code crud\_module:**



### **Justification for Inclusion**

I chose this artifact because it best demonstrates my growth in backend development, data-driven web applications, and secure coding. The process of transforming a basic read-only dashboard into a full-stack CRUD system required me to:

* Understand data flow between client and server
* Safely handle user inputs
* Design a scalable architecture using modular code
* Incorporate defensive coding practices such as error handling and input sanitation

This artifact gave me the opportunity to strengthen my backend development skills and demonstrate my ability to deliver functional, secure, and scalable applications—skills that are critical in a real-world software engineering role.

**Course Outcomes Met**

The enhancements directly align with the following CS 499 course outcomes:

* **Design, develop, and deliver professional-quality technical solutions**: I built a fully functional, database-backed dashboard using professional tools and design patterns (modularity, separation of concerns).
* **Demonstrate innovative techniques and tools in computing practices**: I integrated Python Dash with MongoDB in a modular way, using callbacks and interactive elements to build a modern web UI.
* **Develop a security mindset**: I added input validation, error handling, and restricted unsafe queries, reducing vulnerabilities like data corruption and injection attacks.

### **Reflection & Improvement Process**

While enhancing this project, I learned that real-world development requires **both functionality and security**. Implementing CRUD operations in Dash required a clear understanding of how the frontend elements interact with backend logic through callbacks.

I encountered and overcame several challenges:

* Ensuring data updates reflect **live changes** in the table
* Preventing **invalid entries** (e.g., missing fields, wrong types)
* Managing ObjectId fields from MongoDB in a user-friendly way
* Designing the system in a way that it’s easy to **debug, extend, and test**

Refactoring the logic into crud\_module.py taught me the value of clean separation between application layers. It made my system much easier to maintain and allowed me to focus on one part of the code at a time.

Overall, this milestone represents a major leap in my ability to think like a **backend engineer**, designing secure, maintainable systems that interact directly with databases and users.