Max Gilhespy CS-340 Project Two

**Animal Center Program**

**Overview**

The Animal Center Program is a web-based dashboard developed using Dash and Plotly to visualize and interact with data from an animal shelter. It allows users to explore information about animals in the shelter, their outcomes, and geographical distribution.

**Features**

* **Data Table:** View detailed information about animals, including breed, name, outcome type, and more.
* **Geolocation Chart:** Explore the geographical distribution of animals in the shelter using an interactive map.
* **Pie Chart:** Visualize the distribution of outcomes by rescue type.
* **Filtering:** Filter data by outcome type to focus on specific categories of interest.

**Prerequisites**

Before running the program, ensure you have the necessary Python modules installed. You can install them using the following command:

bashCopy code

pip install dash dash-leaflet dash-table pandas plotly

**Usage**

1. **Clone the Repository:**

bashCopy code

git clone https://github.com/your-username/animal-center-program.git cd animal-center-program

1. **Run the Program:**

bashCopy code

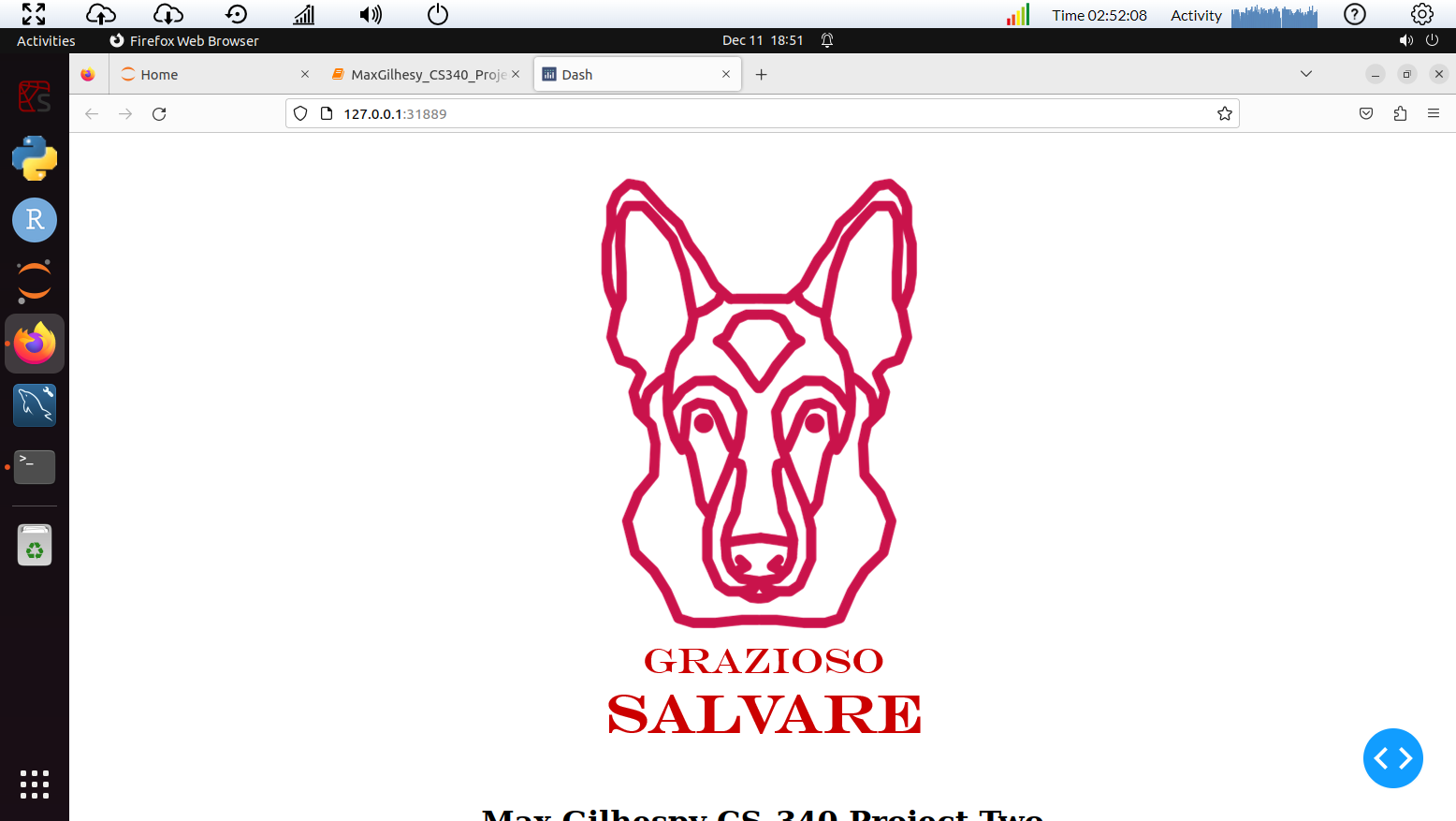
python AnimalCenterClass.py

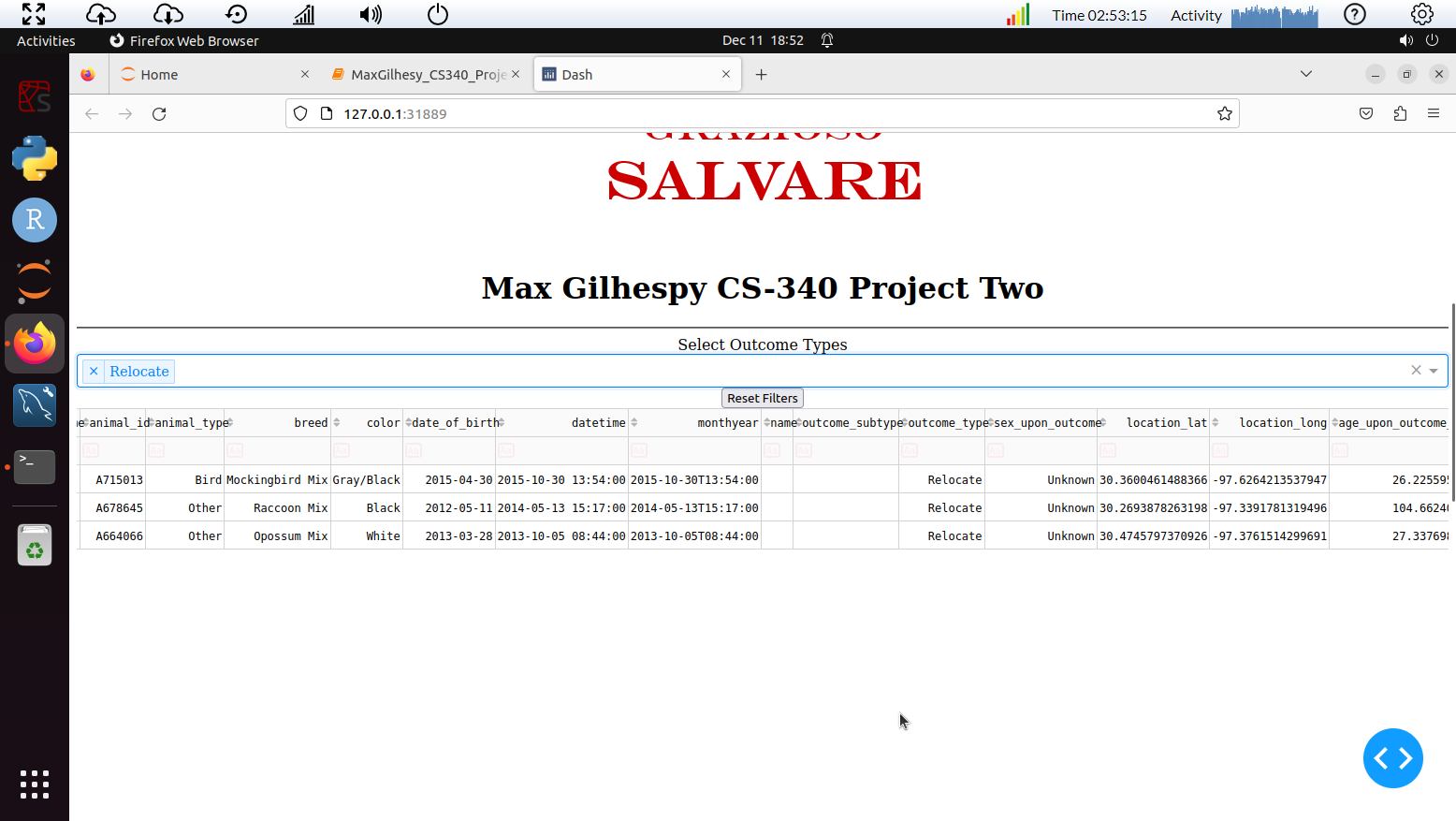
1. **Access the Dashboard:** Open your web browser and navigate to <http://127.0.0.1:8050/> to access the Animal Center Dashboard.

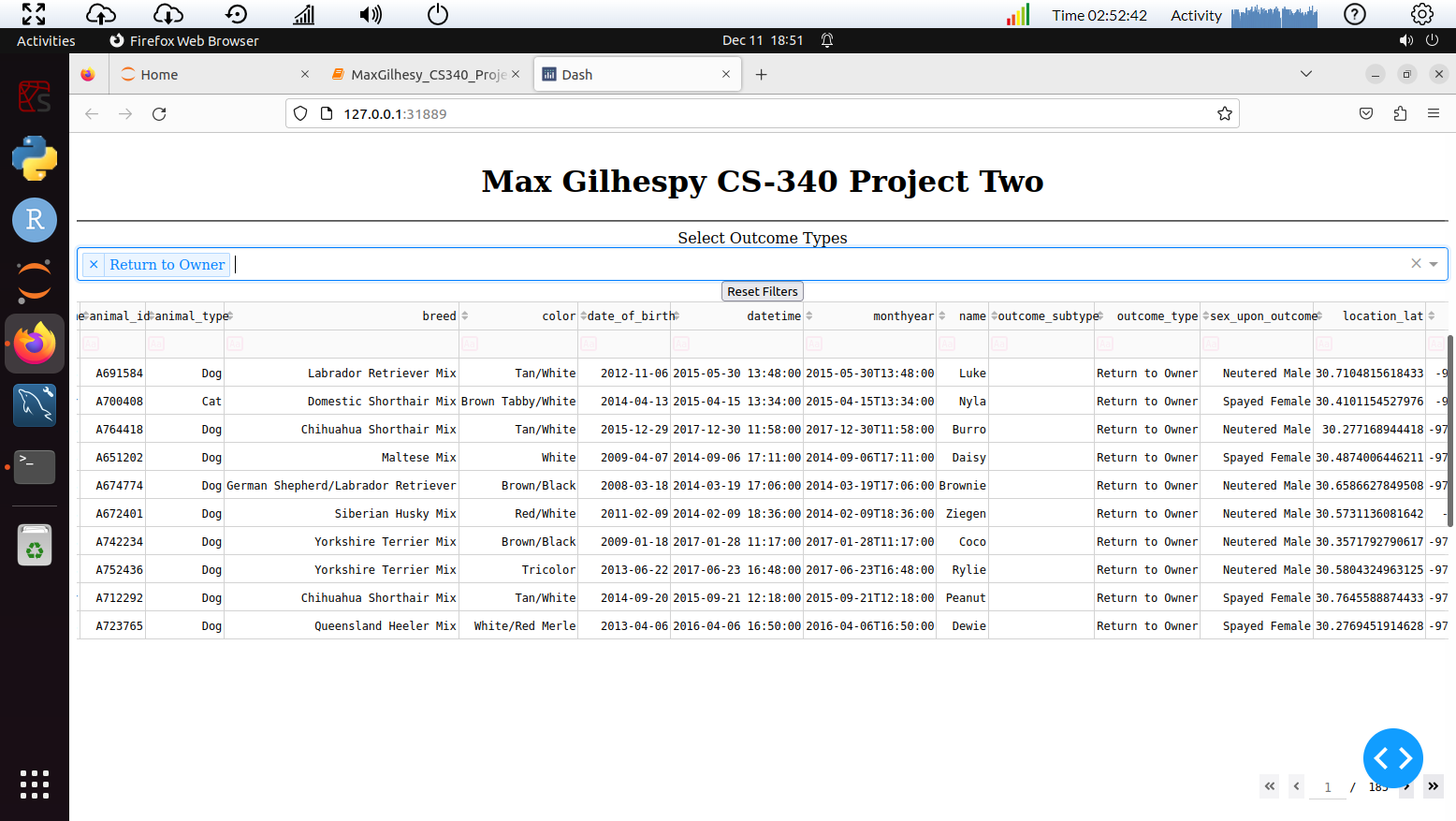
**Configuration**

* Update the **AnimalCenterClass.py** file with your MongoDB credentials and adjust any other necessary settings.

**Screenshots**

****

****

****

**A screenshot of a computer screen

Description automatically generated**

**Contributing**

Feel free to contribute to the development of the Animal Center Program. Follow the [contribution guidelines](https://chat.openai.com/c/CONTRIBUTING.md) for more information.

**License**

This project is licensed under the MIT License - see the [LICENSE](https://chat.openai.com/c/LICENSE) file for details.

**Acknowledgments**

* Grazioso Salvare for inspiration.

Required Functionality

The required functionality of the Animal Center Program includes the following features:

1. **Data Table:**
   * Display detailed information about animals in the shelter.
   * Include columns for attributes such as breed, name, outcome type, and more.
   * Support sorting, filtering, and pagination to enhance data exploration.
2. **Geolocation Chart:**
   * Present a geospatial visualization of the animals' geographical distribution on an interactive map.
   * Utilize markers to represent each animal's location, with tooltips providing additional details.
3. **Pie Chart:**
   * Visualize the distribution of outcomes based on different rescue types.
   * Provide a clear representation of the percentage of each outcome type.
4. **Filtering:**
   * Allow users to filter data based on the outcome type.
   * Enable users to focus on specific categories of interest, enhancing the ability to analyze and understand shelter data.
5. **Reset Filters Button:**
   * Include a "Reset Filters" button to revert to the default state, showing all animals without any applied filters.
6. **User Interface:**
   * Incorporate a visually appealing and intuitive design.
   * Include a logo (Grazioso Salvare’s, in this case) for branding purposes.
7. **Read and Display Data:**
   * Read data from a MongoDB database using the **AnimalCenterClass** module.
   * Display the retrieved data in the data table and update visualizations accordingly.
8. **Interactive Dashboards:**
   * Develop an interactive and responsive web-based dashboard using Dash, incorporating the data table and visualizations.
9. **Deployment:**
   * Allow the program to be run in a Jupyter environment, providing flexibility for users.
10. **Documentation:**
    * Include a README file providing clear instructions on how to run the program, install dependencies, and configure settings.
    * Document any prerequisites, configuration steps, and usage guidelines.

These functionalities collectively create a comprehensive Animal Center Program that enables users to explore and analyze shelter data effectively. The program should be user-friendly, visually appealing, and capable of providing valuable insights into the outcomes and distribution of animals in the shelter.

**Tools Used:**

1. **Python:**
   * Python is a versatile and widely-used programming language known for its readability and ease of use. It offers a rich ecosystem of libraries and frameworks, making it well-suited for web development, data analysis, and visualization.
2. **Dash (Dash by Plotly):**
   * Dash is a productive web framework for building interactive, web-based data visualizations. It's built on top of Flask, Plotly, and React, combining the power of these technologies to simplify the development of interactive dashboards in Python. Dash provides a high-level abstraction for creating web applications without requiring extensive knowledge of web development.
3. **Dash Leaflet:**
   * Dash Leaflet is a Dash component library for building interactive maps within Dash applications. It integrates with Leaflet, a leading open-source JavaScript library for interactive maps. Dash Leaflet enables the inclusion of dynamic and responsive maps in the Dash web application.
4. **Pandas:**
   * Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrames that are essential for handling and processing structured data efficiently.
5. **Plotly and Plotly Express:**
   * Plotly is a graphing library for creating interactive, publication-quality graphs. Plotly Express is a high-level wrapper for Plotly that simplifies the process of creating various charts. These libraries are well-suited for generating dynamic and visually appealing visualizations within Dash applications.
6. **MongoDB:**
   * MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. It was chosen as the model component for the Animal Center Program due to its schema-less nature, allowing for easy storage and retrieval of diverse data types. The JSON-style documents align well with Python dictionaries, simplifying interactions between MongoDB and Python. MongoDB's scalability and flexibility make it suitable for handling large datasets and varied document structures, which is common in real-world applications such as animal shelters.

**Rationale:**

1. **Dash for Web Application Development:**
   * Dash was chosen for its simplicity and productivity in creating interactive web applications using Python. Its reactive components and declarative syntax allow for the seamless integration of data visualizations and user interactions.
2. **MongoDB for Data Storage:**
   * MongoDB's NoSQL, document-oriented database model is well-suited for handling diverse data types and dynamic schemas. In the context of an animal shelter, where data can vary across animals, MongoDB provides flexibility. Its integration with Python through libraries like PyMongo simplifies data retrieval and manipulation.
3. **Pandas for Data Manipulation:**
   * Pandas is widely used for data manipulation and analysis. It complements MongoDB by facilitating data processing and transformation before visualizing it in the Dash application.
4. **Plotly for Interactive Visualizations:**
   * Plotly's interactive visualizations enhance the user experience. Dash's compatibility with Plotly allows for the creation of interactive and dynamic charts that can be seamlessly integrated into the web application.

**Links:**

* [Dash Documentation](https://dash.plotly.com/)
* [Dash Leaflet Documentation](https://dash-leaflet.herokuapp.com/)
* [MongoDB](https://www.mongodb.com/)
* [Pandas Documentation](https://pandas.pydata.org/)
* [Plotly Documentation](https://plotly.com/)
* [Leaflet](https://leafletjs.com/)

These tools were chosen to create a robust, interactive, and user-friendly Animal Center Program that leverages the strengths of each component to achieve the desired functionality. The combination of Python, Dash, MongoDB, and related libraries provides a comprehensive solution for data storage, manipulation, and visualization.

**Steps Taken to Complete the Project:**

1. **Understanding Requirements:**
   * Begin by thoroughly understanding the requirements and functionalities expected in the Animal Center Program. Identify key features such as data visualization, filtering, and mapping.
2. **Choosing Technologies:**
   * Select appropriate technologies based on project requirements. Choose Python as the primary programming language and identify tools such as Dash for web development, Pandas for data manipulation, Plotly for visualizations, and MongoDB for data storage.
3. **Data Collection and Model Implementation:**
   * Implement the AnimalShelter class using MongoDB as the data model. Configure the class to interact with the MongoDB database, handling CRUD (Create, Read, Update, Delete) operations. Retrieve data from the database and create a Pandas DataFrame for further analysis.
4. **Data Exploration and Cleaning:**
   * Explore the dataset to understand its structure and contents. Clean the data by handling missing values, removing unnecessary columns, and converting data types as needed.
5. **Dashboard Layout and Design:**
   * Design the dashboard layout using Dash. Include components for data tables, dropdowns, buttons, and interactive charts. Add branding elements such as images or logos.
6. **Data Visualization:**
   * Implement visualizations using Plotly and Dash components. Create charts like pie charts and maps to represent data trends and geospatial information. Ensure that the visualizations are responsive to user interactions.
7. **User Interaction and Filtering:**
   * Implement user interaction functionalities, such as dropdown filters and buttons, to enable users to filter and customize the displayed data. Integrate these components with callback functions to dynamically update the visualizations based on user inputs.
8. **Map Integration:**
   * Utilize Dash Leaflet to integrate an interactive map into the dashboard. Configure the map to display location-based information, and update it based on user selections in the data table.
9. **Testing and Debugging:**
   * Test the application thoroughly to identify and resolve any bugs or issues. Ensure that user interactions, data filtering, and visualizations work as intended.
10. **Documentation:**
    * Create comprehensive documentation, including a README file that explains the project, its functionalities, and how to run the application. Include information on technologies used, dependencies, and any specific setup instructions.
11. **Version Control:**
    * Use version control systems like Git to track changes in the codebase. This helps manage collaboration, track issues, and maintain a record of project development.
12. **Deployment:**
    * Deploy the application to a server or platform of choice. Ensure that the deployment environment meets the necessary requirements for running the Dash application.
13. **User Training (if applicable):**
    * Provide user training or documentation if the application is intended for use by individuals who may not be familiar with the system. Explain how to navigate the dashboard, use filters, and interpret visualizations.
14. **Feedback and Iteration:**
    * Collect feedback from users and stakeholders. Iterate on the application based on feedback received, making improvements and addressing any additional requirements.

By following these steps, the project can be completed systematically, resulting in a functional and user-friendly Animal Center Program.

**Challenges Encountered and Solutions:**

1. **Dropdown Options Error:**
   * **Challenge:** The dropdown options were not populating correctly, leading to errors.
   * **Solution:** Inspect the data and filter out null or empty values before creating dropdown options. Ensure that the data used to generate options is clean and contains valid entries.
2. **Callback Context Error (Triggered ID):**
   * **Challenge:** The use of **CallbackContext** and **triggered\_id** resulted in an error in older Dash versions.
   * **Solution:** Update the Dash library to a version that supports the **triggered\_id** attribute. If updating is not feasible, find alternative ways to determine the trigger context or consider using **dash.callback\_context.triggered\_id**.
3. **Percentage Calculation in Pie Chart:**
   * **Challenge:** Calculating percentages for the pie chart based on selected outcomes was not straightforward.
   * **Solution:** Calculate the percentages manually by dividing each outcome count by the total count. Ensure that the values are formatted correctly before updating the pie chart.
4. **Dropdown Not Displaying Options:**
   * **Challenge:** The dropdown was not displaying options, even after data cleaning.
   * **Solution:** Check the data types of the outcome types column. Ensure that the dropdown receives a list of dictionaries with 'label' and 'value' keys. Manually inspect the processed data and debug the dropdown creation step.
5. **Deployment and Hosting:**
   * **Challenge:** Deploying the Dash application and hosting it on a server presented challenges related to dependencies and environment setup.
   * **Solution:** Document the dependencies, versions, and any specific configurations needed for deployment. Choose a deployment platform compatible with Dash applications, such as Heroku, AWS, or others, and follow their deployment guidelines.
6. **User Interface and Design:**
   * **Challenge:** Designing an intuitive and visually appealing user interface.
   * **Solution:** Iterate on the dashboard layout, seeking feedback from potential users. Consider user experience principles, and keep the interface clean and easy to navigate. Utilize Dash components and styling options to enhance the visual appeal.
7. **Data Cleaning and Preprocessing:**
   * **Challenge:** Ensuring the data is clean and suitable for visualization.
   * **Solution:** Implement thorough data cleaning steps, including handling missing values and converting data types. Use Pandas methods to preprocess the data before feeding it into Dash components.
8. **Map Integration:**
   * **Challenge:** Integrating an interactive map (Dash Leaflet) into the dashboard.
   * **Solution:** Follow the Dash Leaflet documentation for proper integration. Ensure that the map components, such as markers and tooltips, are configured correctly based on the geospatial data.

By addressing these challenges systematically, the development process can overcome obstacles and result in a robust and functional Animal Center Program.