Rescue Animal Dashboard

Jeffrey Pritchett

About the project.

Global Rain tasked us, as a leading software company, to develop software for Grazioso, a company that identifies preferred dog candidates for search and rescue training. This application displays a table of animal data retrieved from MongoDB, including features like adding, sorting, filtering, editing, deleting, and retrieving information.

It will allow users to filter animals by rescue type (Water Rescue, Mountain or Wilderness Rescue, Disaster or Individual Tracking) using radio buttons. The application displays a pie chart showing the distribution of animal breeds based on available filtered data. Another function is how it displays the geolocation of selected animals on a map, including a tooltip and popup for detailed information.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A white background with many small text

Description automatically generated with medium confidence

A graph of a bar with a pie chart

Description automatically generated with medium confidence

**Vital instruments for the creation of Grazios Slavare application:**

**Dash**

* **Purpose**: Provides the framework for building the web application.
* **Why**: Dash integrates seamlessly with Plotly for visualizations and is designed specifically for building interactive, analytical web applications using Python.

**MongoDB**

* **Purpose**: Serves as the database for storing animal data.
* **Why**: MongoDB is a NoSQL database that provides flexibility in handling semi-structured data. It allows for easy scaling and querying using JSON-like documents, which integrates well with Python and Dash for data retrieval and manipulation.

**JupyterDash**

* **Purpose**: Allows running Dash applications within Jupyter notebooks.
* **Why**: Facilitates quick development and testing of Dash apps within the familiar Jupyter notebook environment.

**Additional Libraries**

* **Plotly**: For creating interactive charts and visualizations.
* **Pandas**: For data manipulation and conversion between data formats.
* **Dash-Leaflet**: For rendering maps and geolocation features.

Helpful links

(*Replace with urls used in creations of application)*

**Why was MongoDB our choice for the development of your application?**

MongoDB was our choice for its ability to control semi-structured data, making it an ideal choice for storing information with an array of attributes. Important qualities include:

* **Flexible Schema:** MongoDB's schema-less design allows for storing records with varying structures without predefined schemas.
* **Scalability:** Easily scales horizontally by sharding data across multiple servers.
* **Rich Query Language:** Provides powerful querying capabilities with support for ad-hoc queries, indexing, and aggregation pipelines.
* **Integration with Python:** Seamless integration with Python through libraries like PyMongo, making it easy to connect and perform CRUD operations.

**Why use Dash framework?**

Dash was used due to its suitability for building data-driven applications. Its advantages include:

* **Python-Based**: Allows developers to write applications entirely in Python.
* **Interactive UI**: Provides components for creating interactive UIs and linking them with backend logic.
* **Plotly Integration**: Directly integrates with Plotly for creating sophisticated visualizations.
* **Customizability**: Offers extensive options for customizing the layout and appearance of the application.

**Steps for a successful application**

**Step 1. Installation**

Install dependencies

* For a successful application you will need to have the compatible parts required to get your application off the ground.

**Step 2. Setup**

MongoDB

* This application requires that you have an instance of MongoDB running.
  + Open command terminal
  + Run the command: mongosh
  + Follow up with the command: use aac
  + Populate the database with data on animals (will be able to provided sample data for testing upon request)

**Step 3. Modify MongoDB Connection**

Configure

* Update/create the *username* and *password* in Animal Shelter database with your (or employees) credentials and permissions needed to access and manipulate the database.

**Step 4. Start**

Running the application

* Launch JupyterNotebook
* Locate JupyterNotebook file containing dashboard code.
* Execute the run function on the provided file to start Dash server.
* Access the dashboard through the local URL

**Averting Crisis**

In any application there are always challenges and with that there is always a solution.

Challenge: JubyterNotebook is unable to locate CRUD file.

* The import function was correct.
* The format was identical to CRUD file
* Why was it not locating the file.

Solution: CRUD file must be located in the same location as jupyter file.

* Even though the code looks correct, if the files are not in the same location/folder, the program will not execute.

Challenge #2: Too much indent or not enough.

* Indent errors wore showing after every run.
* Hint: indents always correlate to previous code, not the following code.

Solution #2: If @app.callback() is not indented, neither is the *def.*

Challenge #3: Stumped, confused, or caveman confused about ball of fire in sky?

* Getting stuck on a problem is normal.
* Being overwhelmed by information is human.
* Trying not to anger ball of sky fire is neanderthal.

Solution #3: Communicate, collaborate, and make cave drawings with tribe.

* Take breaks to renew your mind.
* Communicate the issue with your fellow developers.
* Start working on inventing a spoken language.