David Anguiano

Professor Sanford

Project 2 Read Me

8/11/2022

**READ ME**

This application was created for Grazioso Salvare. This Dashboard will be used to identify dogs that are good candidates for search-and-rescue training.

**REQUIRED FUNCTIONALITY**

* Dashboard Branding
  + Grazioso Salvare Logo with URL redirecting to [www.snhu.edu](http://www.snhu.edu)
  + Unique Identifier containing my name (text or image)
* Dashboard Widgets
  + Interactive Filters
    - Water Rescue
    - Mountain or Wilderness Rescue
    - Disaster Rescue or Individual Tracking
    - Reset
  + Data Table that dynamically responds to filters
  + Geolocation Chart and Second Chart

**CHALLENGES**

During implementation, I faced numerous challenges mostly involving component parameters and data type mismatches. Some component parameters such as the children parameter require a list.

Ex. children=[html.Div()]

Another issue that I ran into was the overlapping of parameters, if a parameter overlaps the component doesn’t know how to interpret the inputs and stops working. At least in my experience.

I overcame all of these challenges by addressing each component one by one and focusing on each component until successfully completing required functionality.

**RESOURCES**

* MongoDB
  + Stores data in JSON-like documents
  + Flexible
  + Scalable
  + https://www.mongodb.com/docs/manual/crud/
* Dash Framework
  + Enables Layout Components
  + Enables Callback Input Interactions
  + Enables Graphs
  + https://dash.plotly.com/

**ANIMALSHELTER.PY**

This is the main class that allows us to Create, Update, Remove, and Delete items in our application. This class allows us to reference methods to communicate with the database. MongoDB stores data in JSON-like documents which makes it very flexible and scalable. For this application, the returnAll method will be our focal point for functionality.

Below I will list all the items highlighted in the screenshot in order.

* These are the libraries necessary for this class to work.
  + Pymongo imports MongoClient which allows us to call the database.
  + Bson.objectId allows us to work with ObjectIds from Bson to Json.
* This method Initializes the MongoClient using self, username, password, db.
* This variable stores and authenticates the connection to the Database using a username and password parameters
* This variable uses the database parameter to store the database we’re working with.
* The return all method uses self, search, and limit as parameters
  + The Self parameter is used to define the current item using these methods.
* If the limit is empty, it just queries based on the search parameter and there is no limit set on the query.
* Else it takes the limit parameter and limits the information pulled from the database. This value must be an integer, even if it’s 0.
* The ReturnAll method returns the searchResults variable which is our query results.

Graphical user interface, text, application

Description automatically generated

**PROJECTTWODASHBOARD.IPYNB**

Below I will be breaking down the contents of ProjectTwoDashboard.ipynb to digestible and easily consumable chunks.

**LIBRARIES & INITIAL SET UP**

These libraries are required for the dashboard to function properly.

* **LIBRARIES**
  + The Dash library allows us to work with different visual components in python.
    - Dash\_leaflet enables us to create map components.
    - Dash\_core\_components enables the use of buttons, Dropdowns, and other fields.
    - Dash\_html\_components enables the use of html components for layout management.
    - Dash\_table enables the use of tables.
    - Dash.dependencies enable input, output, and state changes during callbacks.
  + Plotly.express enables the use of charts.
  + OS enable us to interact with the operating system to reference files.
  + Numpy is used to work with arrays.
  + Pandas enables us to set and manipulate Dataframe types.
  + MongoClient enables database communications.
  + Bson.json\_util imports dumps which allows BSON to be converted to JSON.
  + Base64 enables us to encrypt information in Base64
  + AnimalShelter imports our CRUD class AnimalShelter.py
* **VARIABLES**
  + Shelter variable uses our AnimalShelter class to access the database. Here we input username, password, and the desired database to gain access.
  + Df variable uses pandas to set the default dataFrame the user sees on load.
    - We use our shelter variable and call the returnAll Method. The selectedValue variable sets our search parameter and the {} sets the limit.
    - All {}’s in this method should be 0’s otherwise an error will be thrown for data type mismatch.

Graphical user interface, text, application

Description automatically generated

**HEADER COMPONENTS**

* This variable stores the file name of the Grazioso Logo.
* This variable uses the open method to access our image and read it. This is wrapped with the base64 method to encode the image.
* Html.A method allows us to create link fields for a clickable image.
* Here we set the image type and expected input returning our image.
* Html.A has an href parameter which we can set up with a link.
* Html.H1 sets the header.
* Html.Label sets our unique identifier.
* Style parameter allows us to designate a row letting us place components side by side.

A picture containing text

Description automatically generated

**HEADER BANNER**

Left to Right

* Grazioso Salvare Logo
* Clickable link to [www.snhu.edu](http://www.snhu.edu)
* Unique Identifier and Header

Chart

Description automatically generated

**FILTER**

* Html.Label creates a label for the filters
* This is the id that will allow us to reference this dropdown component for callbacks.
* These are our Dropdown options they require labels and values by default the value is set to 0.

Text, timeline

Description automatically generated

**FILTER CALLBACK**

* Our output is set to our Data Table and will return the data parameter for this component
* The input that influences this call back is our rescue-filters dropdown and this method runs off our value
* Selected\_filter
  + The value is evaluated using if and else if statements that look for specific values
  + If the value matches the Data variable is set to the appropriate find query.
  + The query is inputted into our returnAll method and the dataframe is saved in the df variable.
  + The df variable is turned into a dictionary record type and is returned to the Data Table’s data parameter.

Text

Description automatically generated with medium confidence

**FILTERS**

Graphical user interface, application, Teams

Description automatically generated

**DATA TABLE**

* This is the id for this component used for callbacks
* Selected\_columns is set to empty, we will use this for a column selection.
* Page\_size lets us dictate how many records are displayed to the user at a time.

Graphical user interface, text, application, email

Description automatically generated

**COLUMN SELECTION**

* Ouput for the Data Table using the style\_data\_conditional parameter
* Input using the Data Table functionality from selected\_columns
* Uses for loop to change the color in selected columns
* An additional example of outputs to the Data Table data parameter using the filters call back
* An additional example of Inputs to the influencing parameters within the Data Table

A picture containing text

Description automatically generated

**DEFAULT DASHBOARD LOAD WITH COLUMN SELECTION**

Graphical user interface, application

Description automatically generated

**CHART & GEOMAP**

Below I will cover all the information related to chart and map implementation.

* Starting an html.Div like this starts us in the children parameter which enables us to place multiple items in one section
* **GEO MAP**
  + This id allows us to references this dl.Map component for future callbacks
  + This parameter allows us to dictate where the map will start off
  + This controls the zoom
  + We will place our markers within the Children parameter
* **BAR CHART**
  + This id allows us to reference this dcc.Graph component for future callbacks
* Setting a row style puts all these components side-by-side

Text

Description automatically generated

**GEO MAP**

* Our Output references our map component with the id of map-update and it populates it’s children parameter
* Our Input for this call back uses our rescue-filters dropdown and references it’s value.
* **Update\_Map**
  + This method works very similarly to our filter method which evaluates our value.
  + Sets the query.
  + Queries the Data and sets a Data Frame.
  + In the Markers variable we store the TileLayer which sets the map and the top 3 markers for this data set.
  + Within the Marker component we set the position
  + We set a toolTip
  + We set a popup with the Animal name as a header and the information underneath
  + Then we return the markers variable (our components)

Text

Description automatically generated

A picture containing text

Description automatically generated

**MAP MARKERS FOR WATER RESCUE DATA**

A picture containing chart

Description automatically generated

**BAR CHART**

* Our callback Output is set to our bar-chart component, and it will return the figure parameter
* Our input comes from our rescue-filters and we use the value
* **Update\_barchart**
  + This method works very similarly to our filter method which evaluates our value.
  + Sets the query.
  + Queries the Data and sets a Data Frame.
  + We use the value to control the title for our Chart using our Filter Labels
  + Call a px.bar component using our Data Frame all parameters for these are the actual column names
  + Set the Title
  + Return the barchart variable (our component)

Text

Description automatically generated

Text

Description automatically generated

**CHART COMPONENT UPDATES AND HOVER**

Graphical user interface, application

Description automatically generated

**FULL DASHBOARD UPDATE – Water Rescue**

Graphical user interface

Description automatically generated

**FULL DASHBOARD UPDATE – Mountain or Wilderness Rescue**

Graphical user interface

Description automatically generated with low confidence

**FULL DASHBOARD UPDATE – Disaster Rescue or Individual Tracking**

A picture containing table

Description automatically generated