README

A picture containing text

Description automatically generatedFunctionality:  
 The functionality of this app is to create an overlay through dash, that will read data from a MongoDB database and present the information in a neat and organized manner. Our client Grazioso Salvare wanted to add their logo to the project. The project includes both a pie chart and a geolocational chart that updates based on the data presented on the data table. The data table has filtering options that will change the data in the data table to represent the filter type. As a result, the data in the pie chart and geolocation chart change as well. The following picture displays what the interface looks like:

As you can see, the application has the Grazioso Salvare logo, followed by a data table displaying all the information in the database. Then both the pie chart and the geolocation map are displayed right after the data table. There are four buttons on the top of the data table that serve as filtering options. The four options are Water rescue, Mountain/Wilderness rescue, Disaster/Individual tracking, and reset. Clicking on these buttons will change the information in the data table as well as the pie chart and map.

The following pictures display the different filtering options:

Water Rescue:

A picture containing graphical user interface

Description automatically generated

Mountain/Wilderness Rescue:

Graphical user interface, application

Description automatically generated with medium confidence

Disaster/Individual Tracking:

Graphical user interface, application

Description automatically generated

Reset:

Graphical user interface

Description automatically generated

As you can see from the various screenshots, all the data changes when clicking on the filter options. This allows for the user to be able to go through the data and find the information that pertains to the filtering options faster.

Tools:

The tools used for this project are MongoDB, Python, Dash, Plotly, and Leaflet. MongoDB was the bulk of the back-end operations. MongoDB allows the storage of documents into a database and gives us various operations to retrieve many documents to view and use. With the MongoDB database in the background, we used Python to create a class that will allow us to use CRUD methodology to connect python with the MongoDB database. The python class allows me to create, read, update, and delete files from the database all within my python application. The Python language is very similar to the MongoDB language, so it makes the connection between the two easier to understand. Once the python application is made to use the database, we used Dash to create the application layout. We import our Python class we created to read the MongoDB database into our main application with dash. Now we can create the data table with the information we get from the MongoDB database. Plotly and Leaflet come into play because they are what we use to create the pie chart and geolocation map respectively. The Dash framework allows us to connect everything and display it into a neat and organized layout like what you see above. Python and MongoDB is what we use to gather the data and manages the database respectively. With everything together, we were able to create the application pictured above.

Completing the Project:

The project was a collection of steps taken in order to get the final application. We first began with understanding and using the MongoDB database. Once we understood the language and functionality of MongoDB, we created authentication within the database, with an admin account and a regular user account. The difference is that the regular user is only allowed to read from the database where the admin is allowed to do everything within the database. Once authentication was created, we headed over to Python to create a class that will do our CRUD functions on the database. The class will initiate the database under the regular user and allow us to use all the functions create, read, update , and delete. Once our CRUD functions are created in the class, we create a separate Python project that will use the Dash framework to put the data into a nice web-based application. We begin the application by importing all the necessary tools and especially our Python class that has all the CRUD functions. With everything imported we then start creating the data table that will display all the data from the database in an organized way. Then we work on getting a filter option on the table to allow us to filter between different options. Following that is the addition of the Leaflet geolocation map. The geolocation map will display the location of the first entry of our data table and will update if we change the data table entries. With the Leaflet map in order, we then got a pie chart up and running with Plotly. Plotly functioned the same way as the Leaflet map, displaying the data in the table, and updating based on the filter type. Once we have everything running together, we have an application like the one pictured above.

References:

*Introduction | Dash for Python Documentation | Plotly*. (n.d.). https://dash.plotly.com/introduction