Brandon Stultz

December 10, 2021

CS-340: Client Server Development

Southern New Hampshire University

**READ ME**

1. CRUD

The following presents potential users with the logic required to interact with the application. Text

Description automatically generated

The above image showcases two important. The first is establishing connection to our data base. We initiate this connection by importing ‘MongoClient’ from ‘pymongo’ and importing ‘ObjectId’ from ‘bson.objectId’. The first of the imports enable us to connect to our database while the second import allows for conversions to BSON data structures which we will need to interpret the data in python. This is then followed by our class declaration which will wrap all subsequent methods. Our first method establishes connection to our mongoDB database. This method takes in two key arguments ‘username’ and ‘password’. These arguments are later passed in the next line where we declare and set our ‘client’ using the local host route to make the connection. Passing the method defined ‘username’ and ‘password into the connection to secure authentication. After making the connection to the client we need to determine which database is to be used. This is seen the next line where we initialize and set ‘database’ to read the previous set clients stored data base of ‘AAC’.

We have now successfully connected our CRUD module to our database now we need to generate the logic to interact with our application. The second method present in the above image is CREATE. This method is passed data which is than passed to a mongo create query. Populating the database collection with the provided data.

Text

Description automatically generated

Now that we have a connection to our client and a method to create new entries, we need to be able to read entries already present in the collection. This is what the above image shows. The first of these methods is used to read the first entry found in a search, while the second method is used to find all entries thatr match a search. They both take in data as an argumnet ot the method. This data will take the form of a mongoDB entry item. When passed into either method will return either the first result found given the search peramiters , or return all mathcing results.

Text

Description automatically generated

Having three of the four necessary functionalities we move on to our update method. This method takes in two data streams. The first of the two data streams provide the client with a searchable entry inside the database. The second entry species how to update this entry.

Text

Description automatically generated

the last functionality we need to add is the DELETE method. This method is passed a mongoDB entry, provided by the user, which returns a mongoDB call to remove said entry.This entire file acts as the glue between our application and our backend database. We will move onto the application logic next to showcase this interaction.

Text

Description automatically generated with medium confidence

The above logic showcases the filter options that are presented to the user. In it we see the use of a ‘html.Div’, this ensures our elements are separated from other aspects of the application. We then generate ‘RadioItems’. This is a component that acts as a selection bubble the user can use to make their filter choice. We assign filter division in the application an id called ‘filter-id’. This key value will be used later to apply logic to make the filtering work correct. We will see this later in our description of the code base.

Moving on from there we see an array list of elements stored in the variable ‘options’. This variable is used to inform our ‘RadioItems’ component what ‘label’ should be shown to the user to describe what they are selecting, and the ‘value’ tag is a logical mapping that represents that selectable bubble. We will see its use in the next portion.

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Scatter chart

Description automatically generated

Text

Description automatically generated with medium confidence

The four images above represent a continues block of code that acts as a method that will filter our data table given the selection the user makes in the ‘RadioItems’ section we presented earlier. First we need to discuss the ‘@app.callback’ portion that starts off this segment of logic. This portion of the code is the logic that informs the application where data is to be gathered from and where new information is to be placed after we perform our method on it. This is recognizable from the ‘Input’ & ‘Output’ tags we see. You can also notice these tags take two arguments. ‘Output’ consisting of ‘datatable-id’, ‘data’ & ‘datatable-id’, ‘columns’, while ‘Input’ takes ‘filter-id’ & ‘value’. These value calls inform our application where to draw data user generated information from and what to do with it.

Previously we looked at the filter section of our code. In that section we learned of our id tags that are assigned to that section as well as the ‘RadioItems’ value tags. Their usefulness can be seen in the ‘Input’ tag present in the call back. The first of the two arguments passed to our input call is the ‘filter-id’, this logic informs our application what section of our page to listen to for user activity. The second argument, ‘value’ specifies what the user selected in that given section. Earlier we learned that the ‘value’ tag associated with the filter section was used to identify specific options the user can select, such as water or wilderness rescue. The users input in this section triggers the subsequent code present below the call back. Logically we have four ‘if’ statement. One for each ‘RadioItem’ presented to the user. When the user makes a filter selection using the selectable bubbles. Its ‘value’ is used to determine which logical ‘if ‘statement is used. Depending on the selection a mongoDB search query is run through our CRUD operation we discussed earlier. Returning all the elements in this search as our new datable. The exception to his comes with the reset option which simply returns the data table to listing all entries in the data table collection.

Table

Description automatically generated

Table

Description automatically generated

Above we see that outcome of this filter method. The selectable bubbles represent the filter division. With each bubble corresponding to the particular ‘options’ we examined earlier. I showcase the use of two such options ‘water’ and ‘Wilderness’ rescue. We can see that when the user selects one of these options the databale is changed corresponding to the logic present beneath the ‘callback’.

Graphical user interface, text, application, email

Description automatically generated

This segment of code represents that logic required to generate a pie chart with the given ‘dataTable’. The logic of this function takes in the current ‘derived\_viewport\_Data’. This data corresponds to the data visible to the use in the table. This means that if there are 10 entries present only 10 entries are passed into the method. This allows us to dynamically change that data set we are operating on as the user changes their filtering options, which change that data visible to the user. To enable the use of the pie chart, assign a variable to store this viewport data, we call this ‘’dff’ in our example. We than call a special library called ‘plotly.express’, which we shorten to ‘px’. This can have our data Frame and specific column names passed as arguments that generate a pie chart of our specific choosing. Doing this produces the object presented below.

Chart, pie chart

Description automatically generated