Southern New Hampshire University

Ilyass Hmamou

ilyass.hmamou@snhu.edu

CS-499: Computer Science Capstone

3-2 Milestone Two: Enhancement One: Software Design and Engineering

Southern New Hampshire University

1- Artifact description:

For the improvement in this category of software design and engineering, the chosen artifact is "Salvare Search For Rescue App." It was developed during the period between September 2023 and October 2023 as a component of the final project for CS-340: Client/Server development course.

The application enables users to engage with data from an existing animal shelter database through two distinct methods: an interactive datatable equipped with built-in filters, and an interactive map within a user-friendly interface. Constructed with the utilization of the Python language, pyMongo driver, Dash framework, and MongoDB, the app provides a comprehensive user experience.

2- Reason of choosing this artifact:

The rationale for choosing this particular artifact lies in its capacity to demonstrate proficiency in software design and engineering. The application was developed in adherence to object-oriented programming principles, which involve organizing objects into distinct classes, implementing CRUD operations specific to each object. Additionally, it serves as a testament to skills associated with full-stack development, encompassing back-end, front-end, and database implementation.

This artifact offers promising prospects for improvement, enabling me to demonstrate my capability in analyzing existing projects, identifying weaknesses, and implementing effective solutions. For this particular enhancement, I decided to enhance the user interface (as evident in the attached screens below) and incorporate a server-side filtering option, empowering users to filter by animal type.

3- Enhancement technical details:

The enhancement of this artifact required two significant actions. The initial step involved replicating the project locally, a critical measure that affords complete control over the development environment. This not only facilitates the current enhancement but also lays the groundwork for future improvements, particularly in the context of the upcoming database enhancement. This action item showcases proficiency in the DevOps domain, encompassing the establishment of a Python virtual environment, configuring MongoDB, importing requisite libraries, and integrating the database.

The second step pertains to updates in the code, encompassing both front-end and back-end modifications. Front-end updates focused on enhancing the interface for improved user-friendliness and introducing additional functionalities, such as buttons for filtering by animal type—a feature absent in the original build. On the back-end, a server-side filtering mechanism was implemented to enable users to filter data by animal type directly from the backend, as opposed to relying on client-side filtering within the browser UI. The rationale behind this update is to enhance the application's performance, recognizing that app servers typically possess more computational power than client machines, where client-side filtering typically occurs.

Below more details about the implementation of each of the two steps:

<u>1- Replicated the project locally:</u>

- I set up a Python virtual environment for Jupyter on my local machine (following steps documented here:https://sesync-ci.github.io/faq/python-virtual-env.html)
- Install Mongo db locally from the Mongodb official website link: https://www.mongodb.com/docs/manual/installation/
- Set up local user for mongodb
- Imported the database using the mongoimport command

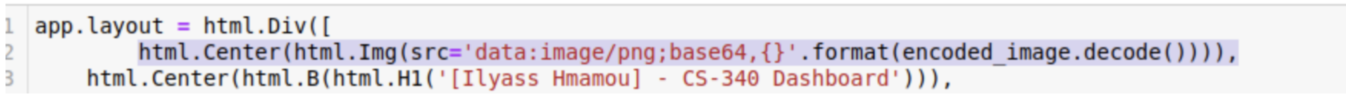
<u>2- code updates:</u>

2.1 Front-end updates:

- In the dashboard layout section, I updated the html div responsible on adding the Grazioso Salvare Logo image to the site, to resize the image to not take entire half of the screen and move it to the side.

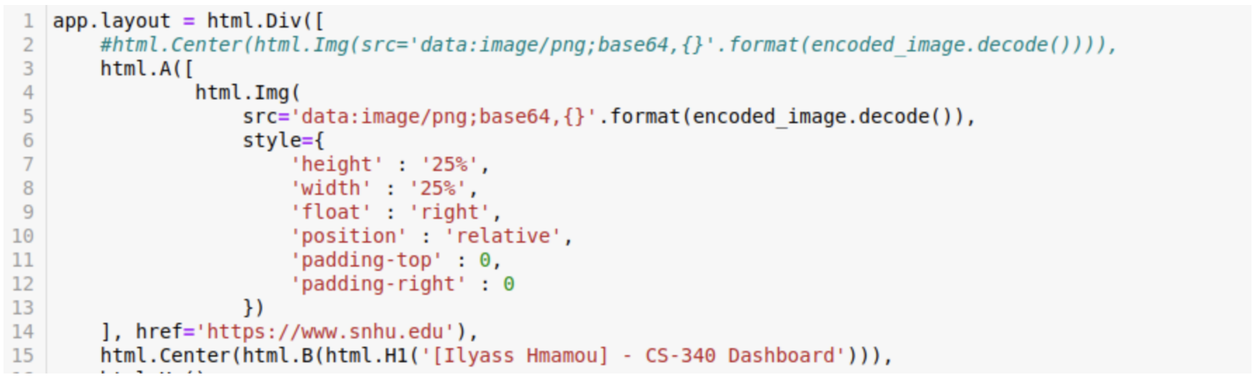
Below screenshot of the initial code Vs the new code

Initial:



```
app.layout = html.Div([
        html.Center(html.Img(src='data:image/png;base64,{}'.format(encoded_image.decode()))),
    html.Center(html.B(html.H1('[Ilyass Hmamou] - CS-340 Dashboard'))),
```

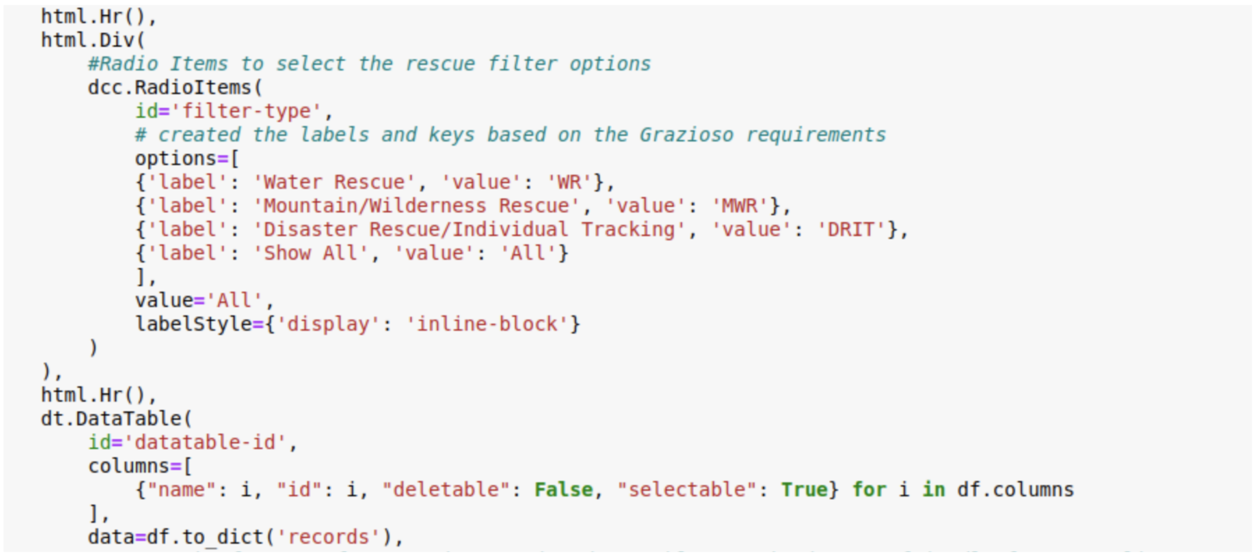
New:



```
app.layout = html.Div([
    #html.Center(html.Img(src='data:image/png;base64,{}'.format(encoded_image.decode()))),
    html.A([
            html.Img(
                src='data:image/png;base64,{}'.format(encoded_image.decode()),
                style={
                    'height' : '25%',
                    'width' : '25%',
                    'float' : 'right',
                    'position' : 'relative',
                    'padding-top' : 0,
                    'padding-right' : 0
                })
    ], href='https://www.snhu.edu'),
    html.Center(html.B(html.H1('[Ilyass Hmamou] - CS-340 Dashboard'))),
```

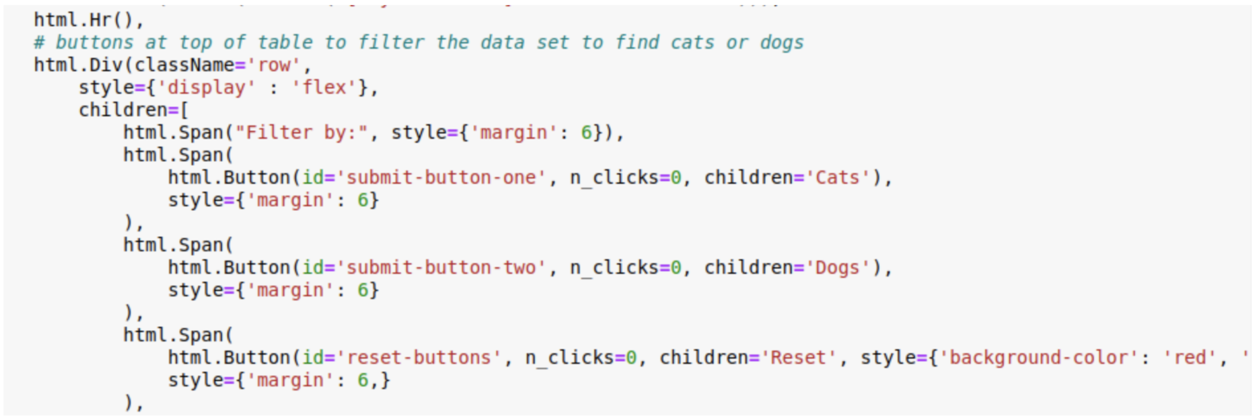
- In the same section (dashboard layout) updated the part responsible on creating the radio buttons for the Rescue Type Filter to be a dropdown menu, and added buttons to search by animal_type (as in the App screenshots in the **Before and After** section)

Initial code:

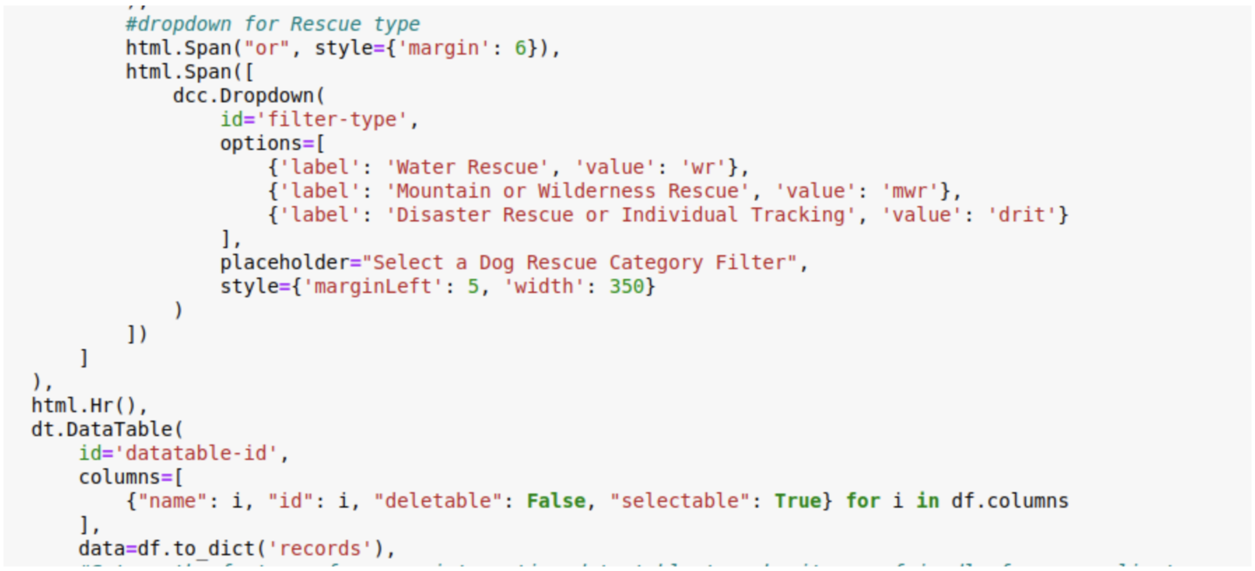


```
html.Hr(),
html.Div(
    #Radio Items to select the rescue filter options
    dcc.RadioItems(
        id='filter-type',
        # created the labels and keys based on the Grazioso requirements
        options=[
        {'label': 'Water Rescue', 'value': 'WR'},
        {'label': 'Mountain/Wilderness Rescue', 'value': 'MWR'},
        {'label': 'Disaster Rescue/Individual Tracking', 'value': 'DRIT'},
        {'label': 'Show All', 'value': 'All'}
        ],
        value='All',
        labelStyle={'display': 'inline-block'}
    )
),
html.Hr(),
dt.DataTable(
    id='datatable-id',
    columns=[
        {"name": i, "id": i, "deletable": False, "selectable": True} for i in df.columns
    ],
    data=df.to_dict('records'),
```

New:



```
html.Hr(),
# buttons at top of table to filter the data set to find cats or dogs
html.Div(className='row',
    style={'display' : 'flex'},
    children=[
        html.Span("Filter by:", style={'margin': 6}),
        html.Span(
            html.Button(id='submit-button-one', n_clicks=0, children='Cats'),
            style={'margin': 6}
        ),
        html.Span(
            html.Button(id='submit-button-two', n_clicks=0, children='Dogs'),
            style={'margin': 6}
        ),
        html.Span(
            html.Button(id='reset-buttons', n_clicks=0, children='Reset', style={'background-color': 'red', '
            style={'margin': 6,}
        ),
```

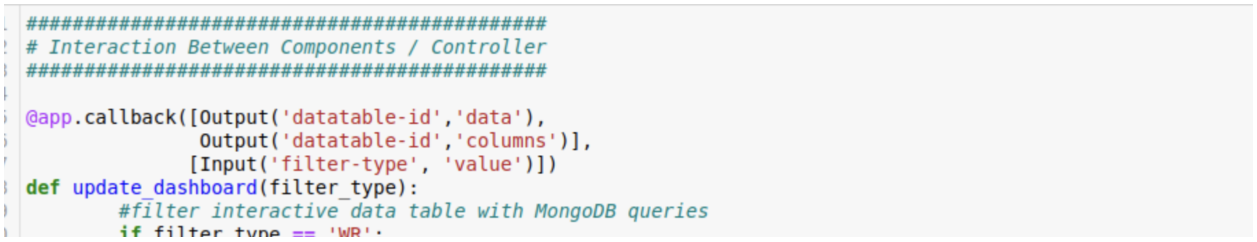


```
        #dropdown for Rescue type
        html.Span("or", style={'margin': 6}),
        html.Span([
            dcc.Dropdown(
                id='filter-type',
                options=[
                    {'label': 'Water Rescue', 'value': 'wr'},
                    {'label': 'Mountain or Wilderness Rescue', 'value': 'mwr'},
                    {'label': 'Disaster Rescue or Individual Tracking', 'value': 'drit'}
                ],
                placeholder="Select a Dog Rescue Category Filter",
                style={'marginLeft': 5, 'width': 350}
            )
        ])
    ]
),
html.Hr(),
dt.DataTable(
    id='datatable-id',
    columns=[
        {"name": i, "id": i, "deletable": False, "selectable": True} for i in df.columns
    ],
    data=df.to_dict('records'),
```

2.2 Back-end updates:

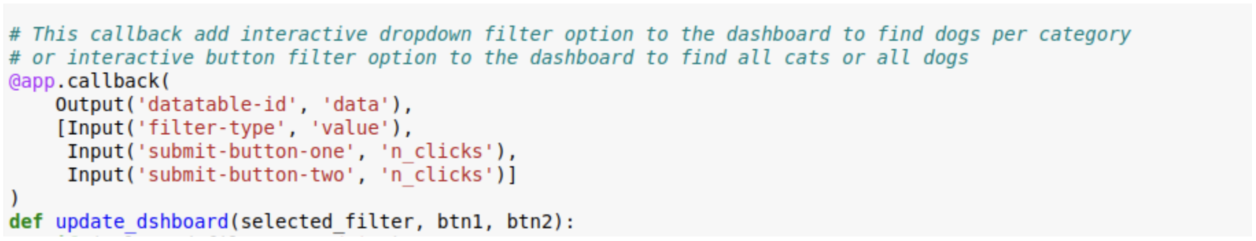
- Updated the app callback action for the dashboard update method to get triggered by the created dropdown and buttons instead of the radio button that the app was initially using.

Initial:



```
################################################
# Interaction Between Components / Controller
################################################

@app.callback([Output('datatable-id','data'),
               Output('datatable-id','columns')],
              [Input('filter-type', 'value')])
def update_dashboard(filter_type):
        #filter interactive data table with MongoDB queries
        if filter_type == 'WR':
```

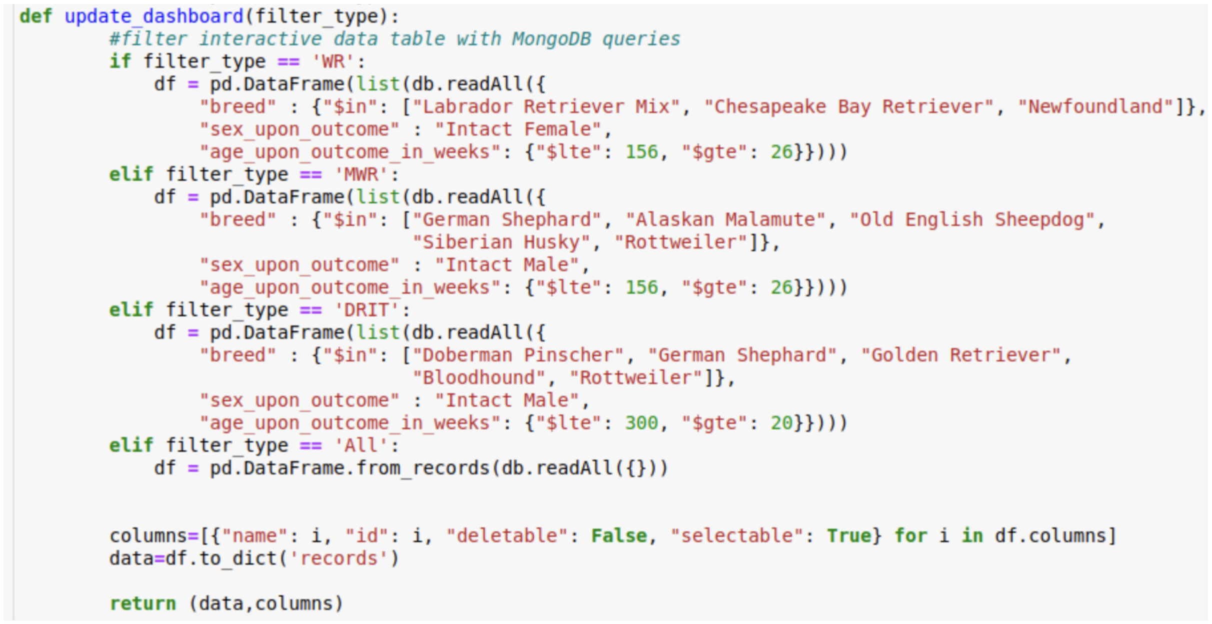
New:



```
# This callback add interactive dropdown filter option to the dashboard to find dogs per category
# or interactive button filter option to the dashboard to find all cats or all dogs
@app.callback(
    Output('datatable-id', 'data'),
    [Input('filter-type', 'value'),
     Input('submit-button-one', 'n_clicks'),
     Input('submit-button-two', 'n_clicks')]
)
def update_dshboard(selected_filter, btn1, btn2):
```

- Updated the method update_dashboard responsible of getting the data from database by calling the CRUD methods implemented in the AnimalShelter class.

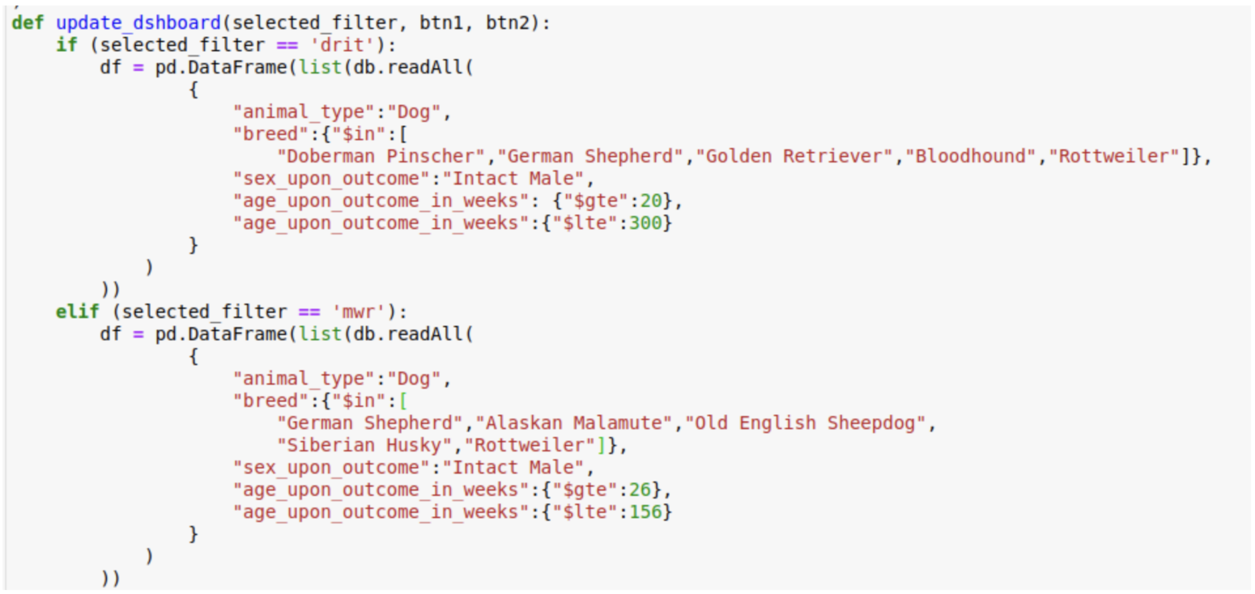
Initial:



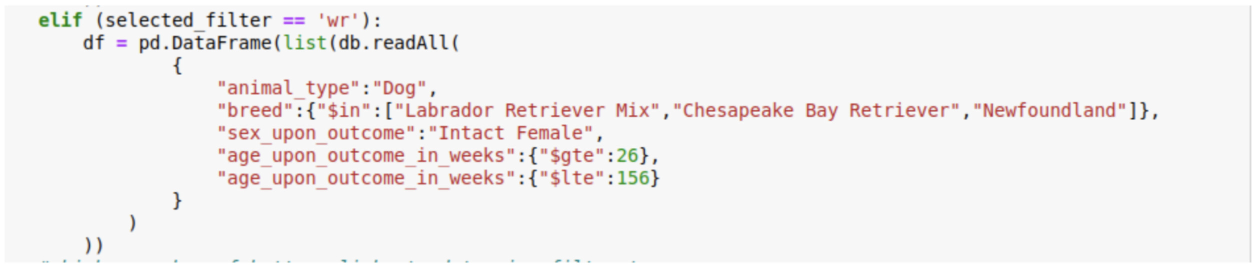
```
def update_dashboard(filter_type):
        #filter interactive data table with MongoDB queries
        if filter_type == 'WR':
            df = pd.DataFrame(list(db.readAll({
                "breed" : {"$in": ["Labrador Retriever Mix", "Chesapeake Bay Retriever", "Newfoundland"]},
                "sex_upon_outcome" : "Intact Female",
                "age_upon_outcome_in_weeks": {"$lte": 156, "$gte": 26}})))
        elif filter_type == 'MWR':
            df = pd.DataFrame(list(db.readAll({
                "breed" : {"$in": ["German Shephard", "Alaskan Malamute", "Old English Sheepdog",
                                   "Siberian Husky", "Rottweiler"]},
                "sex_upon_outcome" : "Intact Male",
                "age_upon_outcome_in_weeks": {"$lte": 156, "$gte": 26}})))
        elif filter_type == 'DRIT':
            df = pd.DataFrame(list(db.readAll({
                "breed" : {"$in": ["Doberman Pinscher", "German Shephard", "Golden Retriever",
                                   "Bloodhound", "Rottweiler"]},
                "sex_upon_outcome" : "Intact Male",
                "age_upon_outcome_in_weeks": {"$lte": 300, "$gte": 20}})))
        elif filter_type == 'All':
            df = pd.DataFrame.from_records(db.readAll({}))


        columns=[{"name": i, "id": i, "deletable": False, "selectable": True} for i in df.columns]
        data=df.to_dict('records')

        return (data,columns)
```

New:

Code for the dropdown filter:

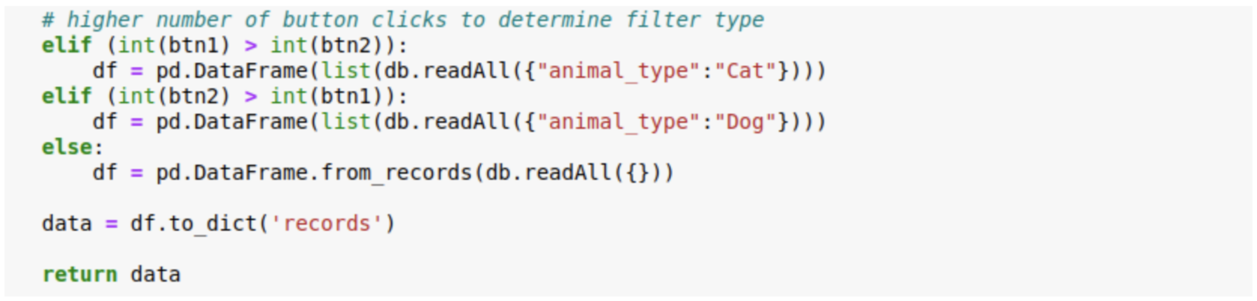


```python
def update_dshboard(selected_filter, btn1, btn2):
    if (selected_filter == 'drit'):
        df = pd.DataFrame(list(db.readAll(
                {
                    "animal_type":"Dog",
                    "breed":{"$in":[
                        "Doberman Pinscher","German Shepherd","Golden Retriever","Bloodhound","Rottweiler"]},
                    "sex_upon_outcome":"Intact Male",
                    "age_upon_outcome_in_weeks": {"$gte":20},
                    "age_upon_outcome_in_weeks":{"$lte":300}
                }
            )
        ))
    elif (selected_filter == 'mwr'):
        df = pd.DataFrame(list(db.readAll(
                {
                    "animal_type":"Dog",
                    "breed":{"$in":[
                        "German Shepherd","Alaskan Malamute","Old English Sheepdog",
                        "Siberian Husky","Rottweiler"]},
                    "sex_upon_outcome":"Intact Male",
                    "age_upon_outcome_in_weeks":{"$gte":26},
                    "age_upon_outcome_in_weeks":{"$lte":156}
                }
            )
        ))
```



```python
    elif (selected_filter == 'wr'):
        df = pd.DataFrame(list(db.readAll(
                {
                    "animal_type":"Dog",
                    "breed":{"$in":["Labrador Retriever Mix","Chesapeake Bay Retriever","Newfoundland"]},
                    "sex_upon_outcome":"Intact Female",
                    "age_upon_outcome_in_weeks":{"$gte":26},
                    "age_upon_outcome_in_weeks":{"$lte":156}
                }
            )
        ))
```

Code for the buttons



```python
    # higher number of button clicks to determine filter type
    elif (int(btn1) > int(btn2)):
        df = pd.DataFrame(list(db.readAll({"animal_type":"Cat"})))
    elif (int(btn2) > int(btn1)):
        df = pd.DataFrame(list(db.readAll({"animal_type":"Dog"})))
    else:
        df = pd.DataFrame.from_records(db.readAll({}))

    data = df.to_dict('records')

    return data
```

Southern New Hampshire University

4- Course objective achieved:

This Artifact and the enhancement made to it, showcases the outcome related to demonstrating an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals.

Upon completing the implementation, it is apparent that this goal was successfully realized. This accomplishment was made possible by integrating class design techniques and leveraging expertise in both back-end and front-end languages to develop the application and implement significant updates for the end users.

The skills utilized to attain this result encompass the following:

1. Establishing a local environment to replicate the application locally, demonstrating proficiency in the DevOps domain, a crucial aspect for initiating any programming project.
2. Performing code updates across the back-end, front-end, and database, showcasing coding skills indicative of a full-stack developer.
3. Introducing the AnimalShelter class in a distinct file, separated from the utility file "projectTwoDash," illustrating a comprehension of Object-Oriented Programming concepts.
4. Evaluating the existing application, identifying weaknesses, and implementing effective solutions, highlighting analytical skills and innovative techniques as a programmer.

5- Reflection:

Revising existing projects and code poses a constant challenge; developers often find creating and implementing new code more preferable than attempting updates on existing ones.

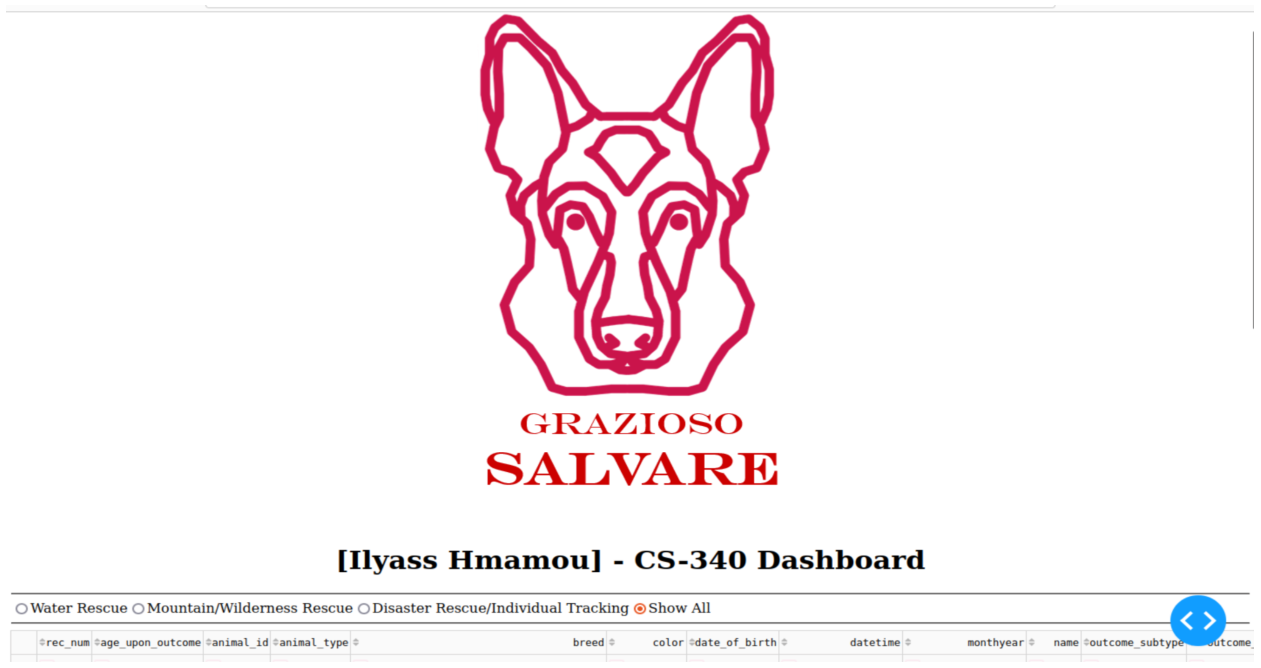
The preference for new code arises from the opportunity to initiate a fresh design and develop new code for projects, free from legacy issues or connections.

During this process, I revisited my previous course documents to review the project and its relevant material. This review was crucial for gaining a deeper understanding, facilitating the implementation of changes and enhancements to the application.

The primary lesson derived from this experience underscores the importance of maintaining thorough documentation for a project and incorporating clear code comments. Effective documentation and meaningful comments prove invaluable when modifications are necessary. Code lacking in meaningful comments becomes notably challenging to comprehend, let alone enhance.

6- Before and after:

Before:



[Ilyass Hmamou] - CS-340 Dashboard

( ) Water Rescue ( ) Mountain/Wilderness Rescue ( ) Disaster Rescue/Individual Tracking [✓ Show All]

rec_num	age_upon_outcome	animal_id	animal_type	breed	color	date_of_birth	datetime	monthyear	name	outcome_subtype	outcome_

Southern New Hampshire University

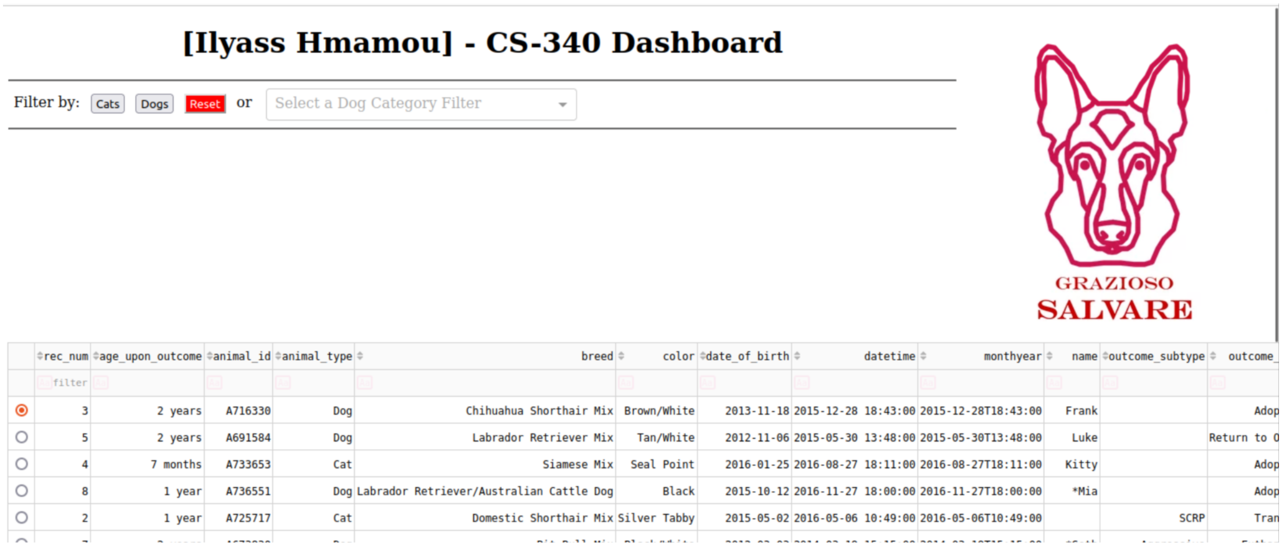
After:

Improvements:

- New buttons for to filter by animal type

- resized image to not occupy half of the screen

- enhanced UI filter for rescue type, by replacing radio buttons with a dropdown



[Ilyass Hmamou] - CS-340 Dashboard

Filter by: Cats Dogs Reset or Select a Dog Category Filter

	rec_num	age_upon_outcome	animal_id	animal_type	breed	color	date_of_birth	datetime	monthyear	name	outcome_subtype	outcome_
	filter											
●	3	2 years	A716330	Dog	Chihuahua Shorthair Mix	Brown/White	2013-11-18	2015-12-28 18:43:00	2015-12-28T18:43:00	Frank		Adop
○	5	2 years	A691584	Dog	Labrador Retriever Mix	Tan/White	2012-11-06	2015-05-30 13:48:00	2015-05-30T13:48:00	Luke		Return to O
○	4	7 months	A733653	Cat	Siamese Mix	Seal Point	2016-01-25	2016-08-27 18:11:00	2016-08-27T18:11:00	Kitty		Adop
○	8	1 year	A736551	Dog	Labrador Retriever/Australian Cattle Dog	Black	2015-10-12	2016-11-27 18:00:00	2016-11-27T18:00:00	*Mia		Adop
○	2	1 year	A725717	Cat	Domestic Shorthair Mix	Silver Tabby	2015-05-02	2016-05-06 10:49:00	2016-05-06T10:49:00		SCRP	Tran