

Florida International University

Knight Foundation School of Computing and Information Sciences

Project Documentation

Project Title: **Monitoring and mitigating misinformation that disrupts adaptation to climate driven health threats**

Team Members:

- **Capstone II:** Roberto Lopez-Trigo, Colleene Mccurdy, Brian Rodriguez, Gabriel Prieto, Javier Oliva, Kyle Ghani

- **Capstone I:** Nathan Chiche, Kyle Zimmer, Jeffrey Quispe
Khizar Nasir

Product Owner(s): Sam Malloy

Mentor(s): Sam Malloy, Mowafak Allaham, Ayse Lokmanoglu

Instructor: Masoud Sadjadi

Abstract

This document presents the foundation of our work over the course of the Summer 2022 semester. The primary goal of this program is to create a system that involves a general dashboard to help monitor misinformation in the climate space. The information used to populate the dashboard comes from various sources such as FEMA, Newswhip, Yale Climate Survey Data and Media Bias Fact Check. The project is broken up into three spaces; the data pipeline that automates the influx of data that comes in daily, the Database to house and organize the data, and a dashboard that helps users infer from this data.

Table of Contents

INTRODUCTION

6

CURRENT SYSTEM

6

USER STORIES

IMPLEMENTED USER STORIES

7

PENDING USER STORIES

7

PROJECT PLAN

REQUIRED SOFTWARE & HARDWARE

8

SPRINT PLANS

9

Sprint 1

9

Sprint 2

11

Sprint 3

14

Sprint 4

17

SYSTEM DESIGN

—WIREFRAME NOTES ARCHIVE—

19

—DATASET VISUAL NOTES *CONCEPTS—

27

—DIAGRAMS (ARCHITECTURAL PATTERNS, SYSTEM DEPENDENCIES, CLASS DIAGRAM, SEQUENCE DIAGRAM)—

29

—CODE BREAKDOWN—

33

PROJECT INFORMATION

50

SPRINT REVIEW REPORTS-

50

Sprint 1

50

Sprint 2

52

Sprint 3

54

Sprint 4

56

Sprint 5

59

INSTALLATION PROCESS

63

SHORTCOMINGS & WISHLIST DOCUMENT

REFERENCES

INTRODUCTION

We created a prototype of a tool that will help researchers identify patterns that occur in the climate social space. This was achieved through the use of a dashboard that pulled data from various sources into one highly interactive space. This gives power to researchers for studying the social attitude towards climate.

There are similar implementations applied to other topics such as Covid(Covaxxy). Our approach differs by offering an indepth look into the articles that cause misinformation campaigns on the publisher/article scale. Every project comes with its risks and ours is the risk of misrepresentation of data. Visuals and sources must be vetted thoroughly before being adopted, in order to be reliable.

This is the second iteration of the project and majority of features were added during this iteration. It was demanding of the team, but tested us in ways that we found enlightening. In the following document you will find a detailed outline of our current system, the user stories we focused on during this iteration, and pending use cases for the next iteration.

Current System

The presentation tier displays information specified by a criteria chosen by the user. Interactive features are implemented for organization and ease of use. Pare down your reports using filters. BERTopic was used to find patterns amongst specific topics. The overall system atmosphere provides a data model, view, and control.

The Application tier functions as a middleware between the backend and the frontend. The database is currently being served using Postgres. Throughout the document, you will find out our archive and the current plans that are set-in-place for the project.

USER STORIES

User Story 1 - Database Team --Completed

User Story 2 - Efficiency Team --Completed

User Story 3 - Feature Team --Completed

User Story 4 – UI/UX Team --Completed

User Story 5 – Machine Learning Team --Completed

User Story 6 - Scraper (Gabriel & Roberto)

--Need to scrape for 2017-2020

User Story 7 - Database Queries (Colleene, Brian, and Roberto)

--Need to finish queries and properly house non-newswhip data like the 'FEMA data'

User Story 8 - Anomaly detection (Brian, Roberto, Colleene, Gabriel, Javier)

-- Will be touched upon by PO in first meeting

User Story 9 - Date slider (Done) --Completed

User Story 10 – Theme changer (Khizar, Nathan, Jeffrey) --Completed

User Story 11 - General UI adjustments to overall theme. (Khizar, Nathan, Jeffrey) --Completed

User Story 12 - Survey data UI (In a week) --Completed

User Story 13 - Proportion to overall engagement (Roberto, KyleG) --Completed

User Story 14 - Informational text box (Javier, KyleZ, Roberto) --Completed

User Story 15 - Machine learning model (Brian and Colleene) --Completed

User Story 16 - Sql injection attack (Roberto, Kyle G, Colleene) --Completed

User Story 17 - Automate setup environment --Completed

User Story 18 - BertTopic (Brian and Colleene and Roberto)

--Needs to be redone after cleaning the data and adjusting BERT parameters

User Story 19 - MBFC Graph (Colleene, Brian, and Roberto) --Completed

PROJECT PLAN

From this part of the document, we'll demonstrate our progression through utilizing daily scrum user stories, software/hardware requirements for the project, and other related dependencies that were involved in the project.

Hardware and Software Req

Hardware:

- Any computer with an OS. There isn't any particular hardware that is needed for this project.

Software:

- Python - The main language that is used for this project.
- Visual Studio Code 2019 - Typically our main IDE that's used to implement most of our features. Any other IDE should be fine.
- Dash - Assists in managing the database and plotly app.
- Jupyter Notebook- Helps with manipulation of data and training the models
- Plotly - Allows a point & click interface. Assists in the creation of the project's dashboard.

Sprint Plans

Sprint #1 Planning Meeting

Project: Monitoring Climate Misinformation

Date: 5/18/22

Start Time: 6:00 pm

End Time: 8:00 pm

Attendees:

- Roberto Lopez-Trigo (RL) **(Capstone II Leader)**
- Colleene Mccurdy (CM)
- Gabriel Prieto (GP)
- Javier Oliva (JO)
- Kyle Ghani (KG)
- Brian Rodriguez (BR)
- Nathan Chiche (NC) **(Capstone I Leader)**
- Kyle Zimmer (KZ)
- Jeffrey Quispe (JQ)
- Khizar Nasir (KN)

After discussion, the velocity of the team was estimated to be 10+ hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story 1 - Database Team
- User Story 2 - Efficiency Team
- User Story 3 - Feature Team
- User Story 4 – UI/UX Team

The team members indicated their willingness to work on the following user stories.

- Roberto Lopez-Trigo
 - User Story 2 - Efficiency Team
 - User Story 3 - Feature Team
 - User Story 4 – UI/UX Team

- Gabriel Prieto
 - User Story 2 - Efficiency Team
 - User Story 3 - Feature Team
- Colleene McCurdy
 - User Story 1 - Database Team
- Kyle Ghani
 - User Story 1 - Database Team
 - User Story 2 - Efficiency Team
- Brian Rodriguez
 - User Story 1 - Database Team
 - User Story 3 - Feature Team
- Javier Oliva
 - User Story 2 - Efficiency Team
 - User Story 4 – UI/UX Team
- Kyle Zimmer
 - User Story 1 - Database Team
- Khizar Nasir
 - User Story 3 - Feature Team
 - User Story 4 – UI/UX Team
- Jeffrey Quispe
 - User Story 2 - Efficiency Team
- Nathan Chiche
 - User Story 3 - Feature Team

Sprint #2 Planning Meeting

Project: Monitoring Climate Misinformation

Date: 5/30/22

Start Time: 6:00 pm

End Time: 8:00 pm

Attendees:

- Roberto Lopez-Trigo (RL) **(Capstone II Leader)**
- Colleene Mccurdy (CM)
- Gabriel Prieto (GP)
- Javier Oliva (JO)
- Kyle Ghani (KG)
- Brian Rodriguez (BR)
- Nathan Chiche (NC) **(Capstone I Leader)**
- Kyle Zimmer (KZ)
- Jeffrey Quispe (JQ)
- Khizar Nasir (KN)

After discussion, the velocity of the team was estimated to be 10+ hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story 1 - Database Team
 - Implement relational, searching/indexing, and security best practices.
- User Story 2 - Efficiency Team
 - Implement memory and caching best practices for callbacks.
- User Story 3 - Feature Team
 - Implement more visuals and interactive components.
- User Story 4 – UI/UX Team
 - Implement improvements for usability and design.

The team members indicated their willingness to work on the following user stories.

- Roberto Lopez-Trigo
 - User Story 2 - Efficiency Team
 - User Story 3 - Feature Team
 - User Story 4 – UI/UX Team
- Gabriel Prieto
 - User Story 2 - Efficiency Team
 - User Story 3 - Feature Team
- Colleene McCurdy
 - User Story 1 - Database Team
- Kyle Ghani
 - User Story 1 - Database Team
 - User Story 2 - Efficiency Team
- Brian Rodriguez
 - User Story 1 - Database Team
 - User Story 3 - Feature Team
- Javier Oliva
 - User Story 2 - Efficiency Team
 - User Story 4 – UI/UX Team
- Kyle Zimmer
 - User Story 1 - Database Team
- Khizar Nasir
 - User Story 3 - Feature Team
 - User Story 4 – UI/UX Team
- Jeffrey Quispe
 - User Story 2 - Efficiency Team

- Nathan Chiche
 - User Story 3 - Feature Team

Sprint #3 Planning Meeting

Project: Monitoring Climate Misinformation

Date: 6/27/22

Start Time: 9:00 pm

End Time: 10:30 pm

Attendees:

- Roberto Lopez-Trigo (RL) **(Capstone II Leader)**
- Colleene Mccurdy (CM)
- Gabriel Prieto (GP)
- Javier Oliva (JO)
- Kyle Ghani (KG)
- Brian Rodriguez (BR)
- Nathan Chiche (NC) **(Capstone I Leader)**
- Kyle Zimmer (KZ)
- Jeffrey Quispe (JQ)
- Khizar Nasir (KN)

After discussion, the velocity of the team was estimated to be 10+ hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- User Story 1 - Database Team
 - Administer hosting
 - Work with features, UI and efficiency team to optimize data usage from application
- User Story 2 - Efficiency Team
 - Optimize data transmissions and load times
- User Story 3 - Feature Team
 - Create visuals for analysis from tested hypotheses
 - Creating UI made by the below team to be used for user interaction with the survey data
- User Story 4 – UI/UX Team

Create and update UI for new data sets based with our demographic in mind

- User Story 5 – Machine Learning Team

Scrape data from news articles to be used for the creation of a model that can predict the engagement score and other features of incoming articles.

Test hypotheses

The team members indicated their willingness to work on the following user stories.

- Roberto Lopez-Trigo
 - User Story 2 - Efficiency Team
 - User Story 3 - Feature Team
 - User Story 4 – UI/UX Team
 - User Story 5 – Machine Learning Team
- Gabriel Prieto
 - User Story 2 - Efficiency Team
 - User Story 3 - Feature Team
 - User Story 5 – Machine Learning Team
- Colleene McCurdy
 - User Story 1 - Database Team
 - User Story 5 – Machine Learning Team
- Kyle Ghani
 - User Story 1 - Database Team
 - User Story 2 - Efficiency Team
- Brian Rodriguez
 - User Story 1 - Database Team
 - User Story 3 - Feature Team

- User Story 5 – Machine Learning Team
- Javier Oliva
 - User Story 2 - Efficiency Team
 - User Story 4 – UI/UX Team
 - User Story 5 – Machine Learning Team
- Kyle Zimmer
 - User Story 1 - Database Team
 - User Story 5 – Machine Learning Team
- Khizar Nasir
 - User Story 3 - Feature Team
 - User Story 4 – UI/UX Team
- Jeffrey Quispe
 - User Story 2 - Efficiency Team
- Nathan Chiche
 - User Story 3 - Feature Team

Sprint #4 Planning Meeting

Project: Monitoring Climate Misinformation

Date: 7/11/22

Start Time: 8:00 pm

End Time: 10:00 pm

Attendees:

- Roberto Lopez-Trigo (RL) **(Capstone II Leader)**
- Colleene Mccurdy (CM)
- Gabriel Prieto (GP)
- Javier Oliva (JO)
- Kyle Ghani (KG)
- Brian Rodriguez (BR)
- Nathan Chiche (NC) **(Capstone I Leader)**
- Kyle Zimmer (KZ)
- Jeffrey Quispe (JQ)
- Khizar Nasir (KN)

After discussion, the velocity of the team was estimated to be 10+ hours.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

User Story 1 - Scraper (Gabriel & Roberto & Jeffrey & Brian)

User Story 2 - BertTopic (Brian and Colleene and Roberto)

User Story 3 - Informational text box (Javier,Jeffrey,KyleZ)

Add text to place holders

User Story 4 - Anomaly detection (Brian, Roberto, Colleene, Gabriel, Javier)

Get mean for each month and standard deviation find outliers that are two standard deviations greater than the mean.

User Story 5 - General UI adjustments to overall theme. (Javier, Roberto,Jeffrey,KyleZ)

Create Figma for Survey data

Find ways to correlate data.

Theme changer ()

Change Graph box colors and dropdown colors

Navbar button should be more visible

User Story 6 - MBFC Graph (Colleene, Brian, and Roberto)???

Filter by MBFC Graph engagement sum

pd.groupby ('MBFC').sum() Something like that

User Story 7 - Proportion to overall engagement (Roberto & Brian)

Line Graph

User Story 8 - Sql injection attack (Roberto, Kyle G, Colleene)

Attack database to expose security threats

User Story 9 - Automate setup environment ()

Requirements txt file

SYSTEM DESIGN

Wireframe Notes Archive

In this section, these are just an archive of attempted wireframe designs that were utilized in forming our plans of a construction of a wireframe. Many of these designs have been scrapped or have been derived to the basis of our current wireframe. From the pages below are a series of images that demonstrate and explain the concept of each wireframe.

These designs have been adapted to the current application designs and actual implementation will be shown.

To Capstone students who work on future iterations of the project, please try to grasp an idea from one of these and attempt to formulate what the final version would look like based on the designs and current product.

DESIGNS:

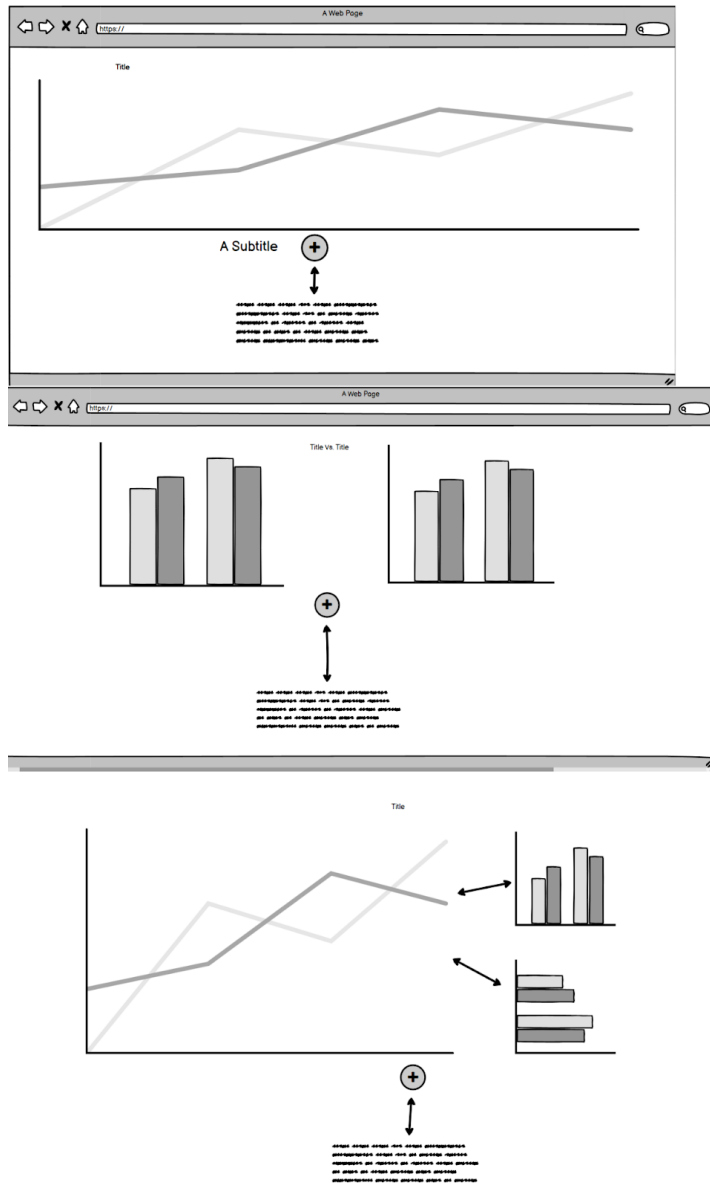
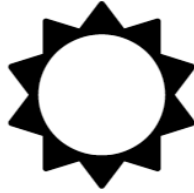


Image 1: An old concept of a wireframe. Supposedly, the circle would hide/expand a description.

Scrapped in favor of other wireframes.

Climate TruthWatch

We look to analyze the flow of misinformation on climate change through the spread of information on different social media sites



The Online Climate Discussion Map ?

Social Media Engagement across the U.S.



% of Climate Change Support

Social Media Engagement and Climate Change Support: Corelations ?

Climate Denial (% of pop) ▼

Percentage of low credible sources shared ▼

By Social Media

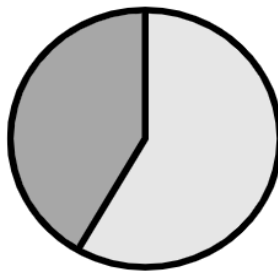
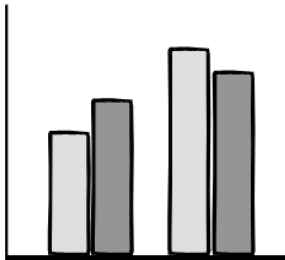


Image 2: Another concept of a wireframe. Scrapped in favor of other wireframes.

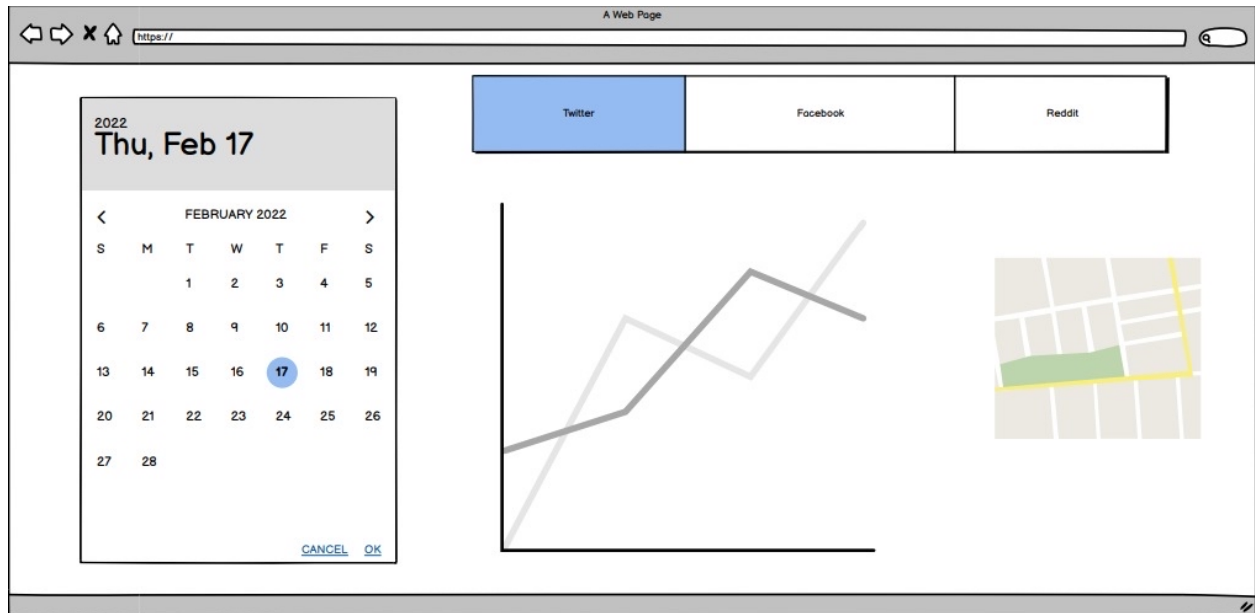


Image 3: Yet another concept of a Wireframe. Scrapped because of the impractical GPS and calendar. We would later derive a piece of this concept to include a drop down menu, a map of the U.S. population affected, and initializing centered text for the tool bars.



Image 4: Last concept of our wireframe. Truth section isn't feasible. Claim text is good, it can be used to summarize frequencies with each URL. This concept allowed us to focus on URL sources and credibility scores regarding information fact checking.



Image 5: Home page design on Figma. It contains information about the data sources and data processing. It also contains the design of the navbar.

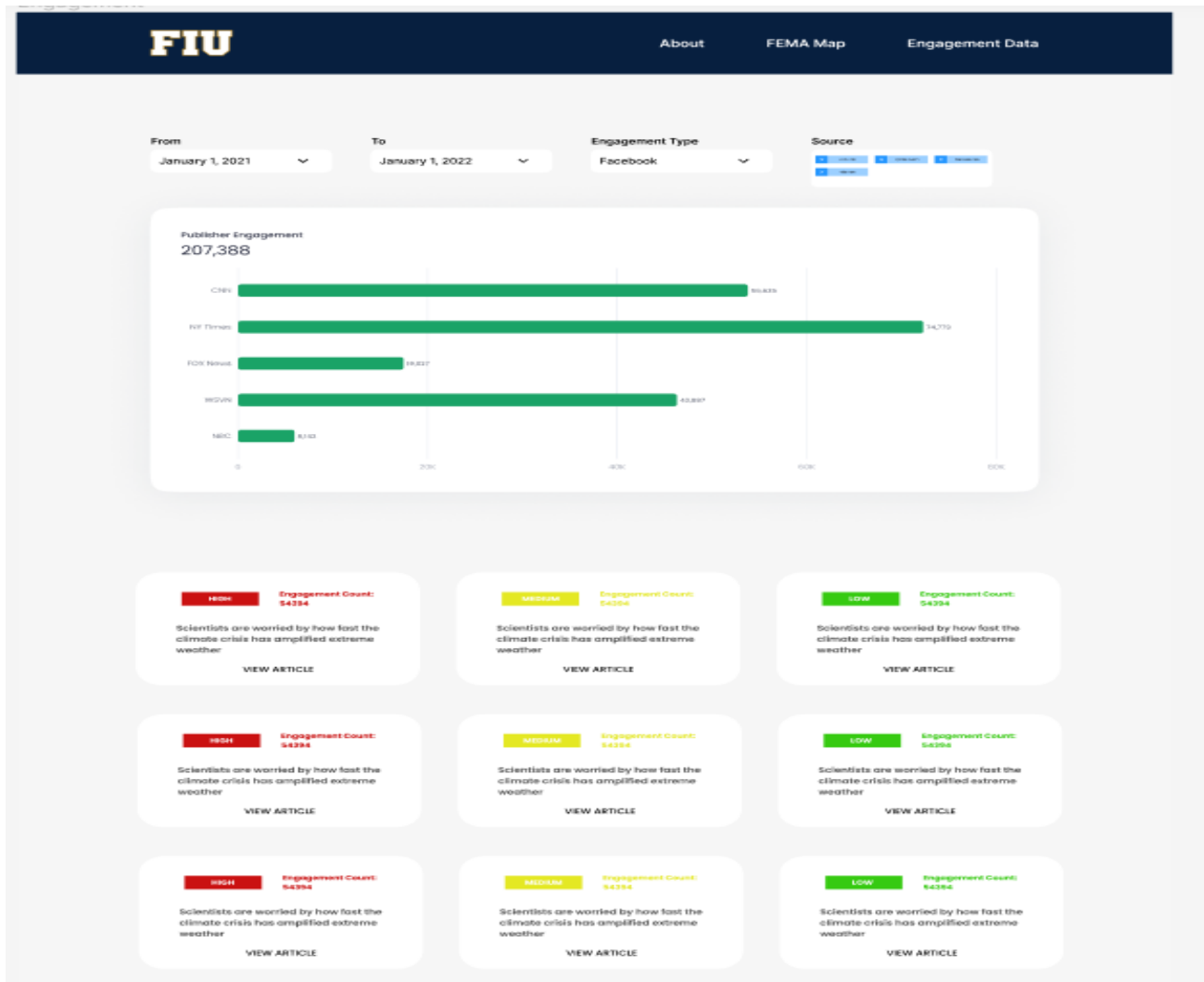


Image 6: Card and histogram designs on Figma. This was the initial design on Figma and inspiration for the creation of these features.



Image 7: Final implementation design of most features on Figma. Histogram, BERTopic, Cards

Dataset Design Notes Archive

The pages below are concepts of dataset visuals that are used to display claims from a dataset. From here, you can see the overall evolution of our displays we've derived.

DESIGNS:

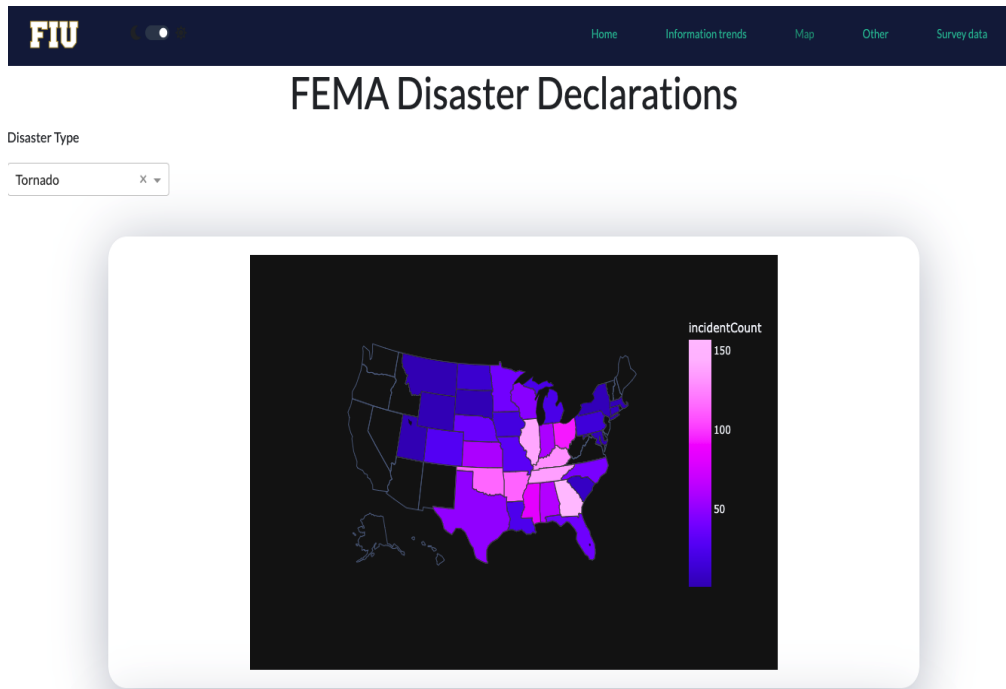


Image 6: The disaster type menu and timeline served as inspiration in construction of our drop down filter.

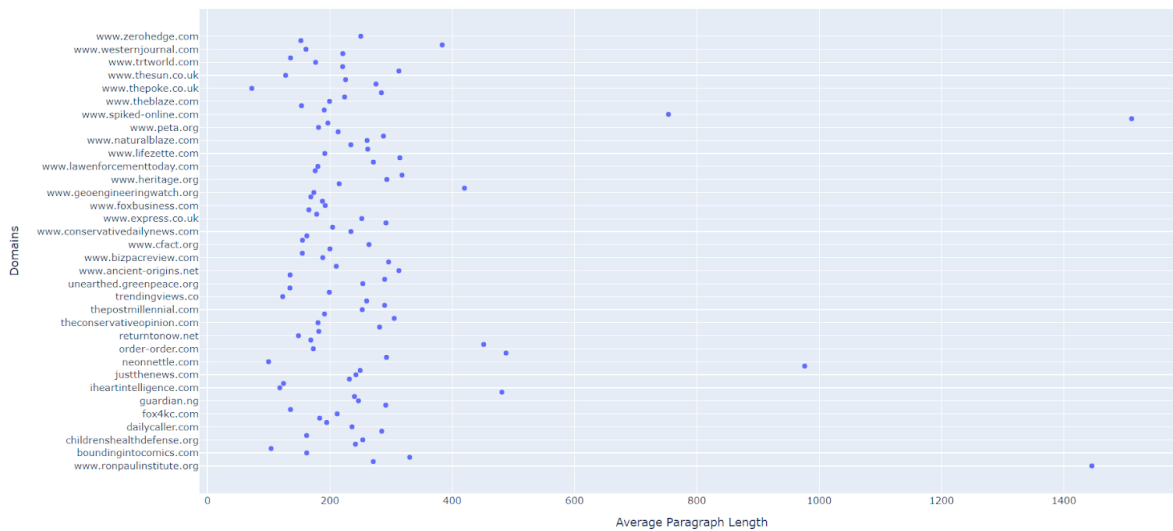
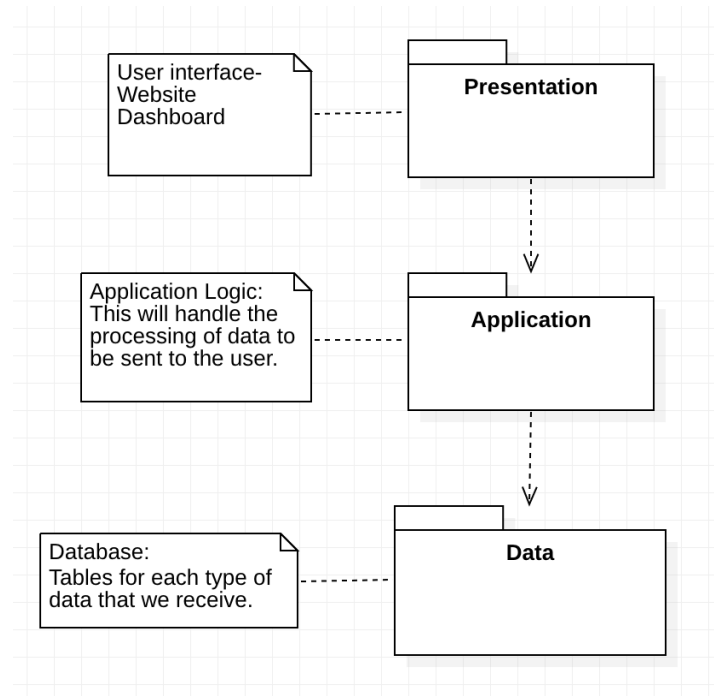


Image 7: The last concept used before implementing our dataset design.

Diagrams

This section will cover some diagrams related to the construction of our code. Down here serve some illustrations of which consists an architectural pattern, a diagram of system dependencies, a class Diagram, and sequence diagram

Image 7: Architectural Pattern
Primary Architecture:



Secondary Architectures:

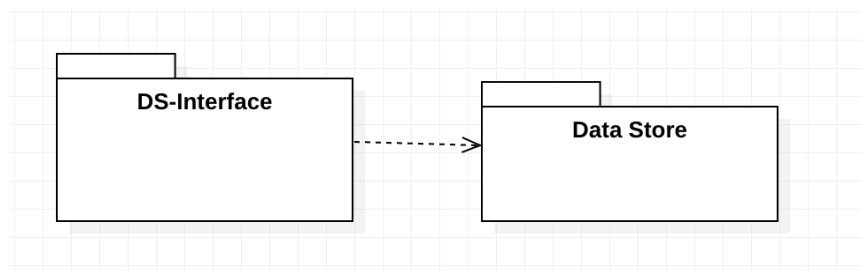
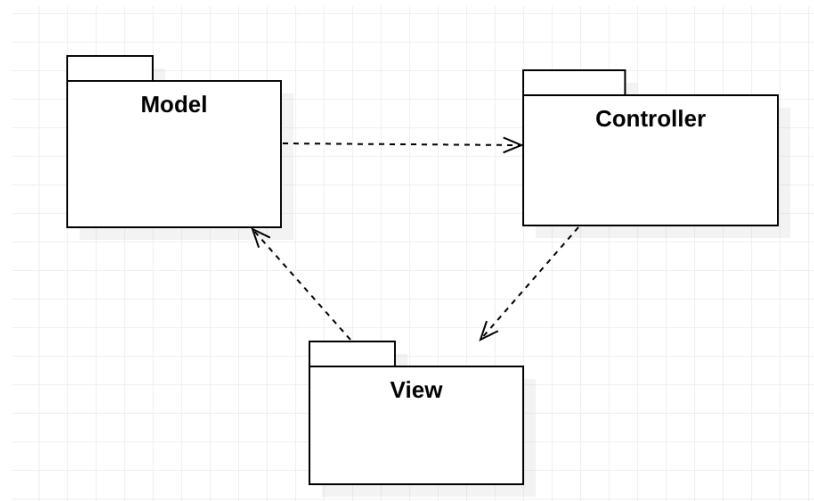


Image 8: Minimal Class Diagram

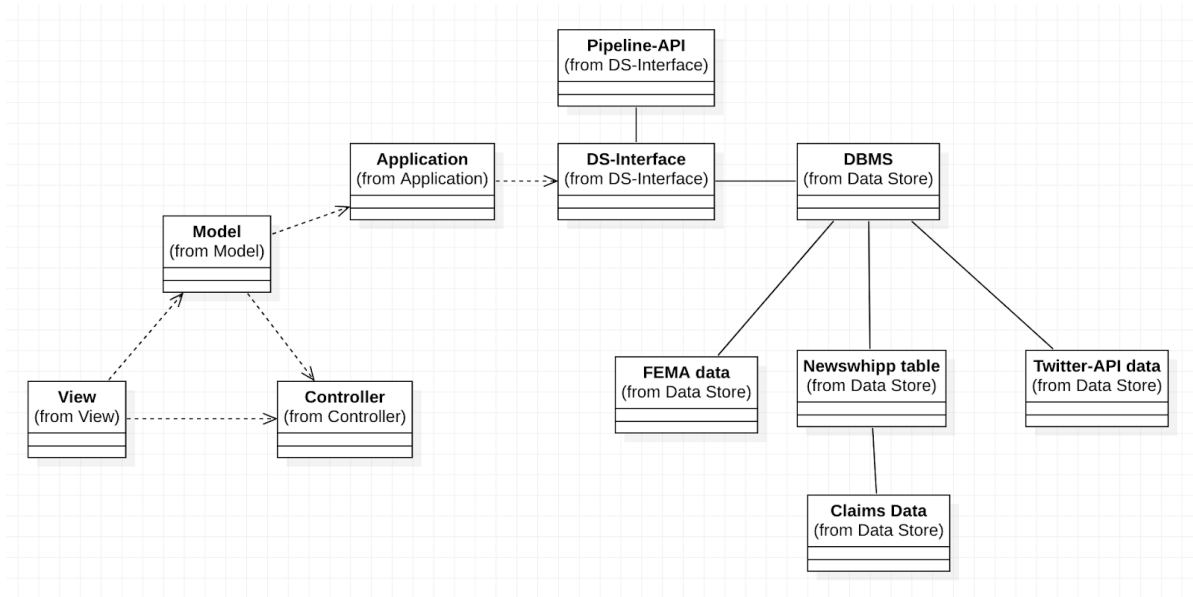
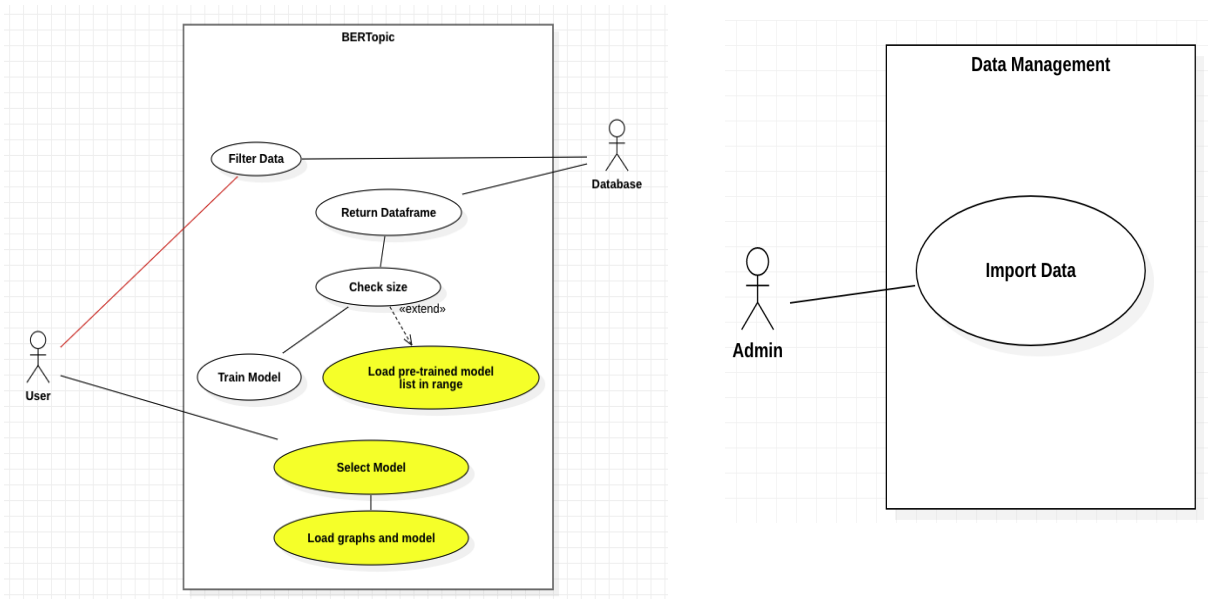
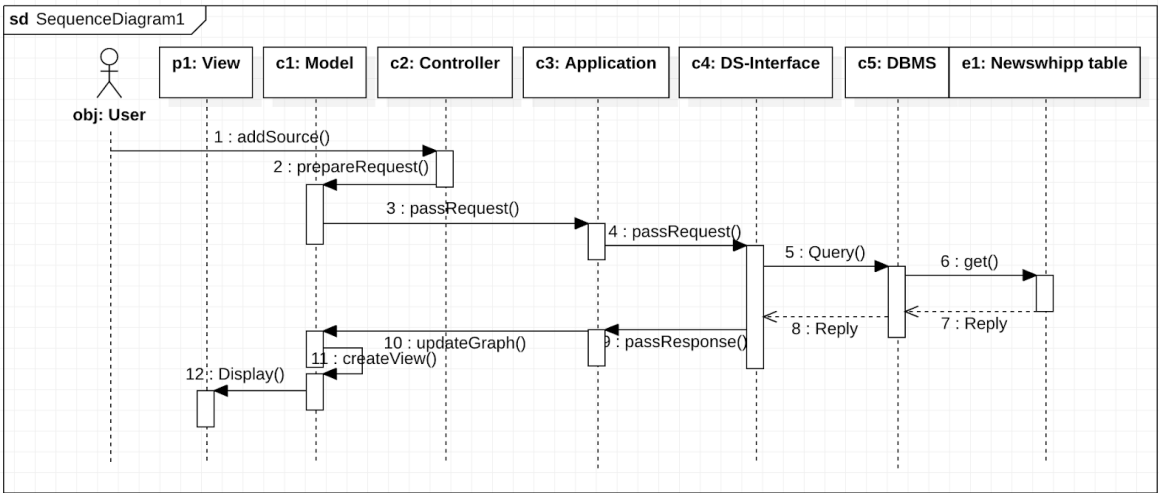


Image 9: Sequence and Use Case Diagrams





Code Breakdown

In this section we will break down the code in each section of the project.

Website and app.py:

The website has 5 main pages.

- Information Trends
- Other
- Home
- Survey Data
- FEMA map

These five pages are spawned by the app.py script. That script loops through the scripts within the pages directory and creates a page in the navbar for each script.

```
app.layout = dbc.Container(  
    [navbar, dl.plugins.page_container],  
    fluid=True,  
    style={  
        'margin': '0',  
    }  
)
```

App will create the pages and register their paths to the base url. In development this is localhost:8050/<script/pagename>

```
navbar = dbc.Navbar(  
    [  
        html.A(  
            # Use row and col to control vertical alignment of logo / brand  
            dbc.Row(  
                [  
                    dbc.Col(html.Img(src='assets/FIU-Logo.png', height="30px")),  
                ],  
                align="center",  
                className="g-0",  
            ),  
            href="https://plotly.com",  
            style={  
                "textDecoration": "none",  
                "margin-left": "30px"  
            },  
        ),  
        #Darkmode switch  
        html.Div(  
            ThemeSwitchAI0(aio_id="theme", themes=[url_theme1, url_theme2]),  
            className="darkMode"  
        ),  
        dbc.Container(  
            [  
                dbc.NavItem(dbc.NavLink(page["name"], href=page["path"], className='navLinks'))  
                for page in dash.page_registry.values()  
                if page["module"] != "pages.not_found_404"  
            ],  
            style={  
                'display': 'flex',  
                'justify-content': 'right'  
            },  
        ),  
    ],  
    color="#071F3F",  
    dark=True,  
    # class_names='navbar navbar-expand-lg navbar-light bg-light'  
)
```

Navbar ui is created here. It follows a similar architecture to react layout.

Dcc and Dbc are libraries that carry many components ready to use in Dash. They can be modified by looking into their documentation or through css styling in the style attribute of the function.

Information Trends:

This is the largest script. It's about 1000 lines long. There are two versions of the script. One that uses the postgres database and one that uses static data in the Data folder.

The postgres version constructs queries based on user input. The static data version loads in the data using pandas read_csv function to load in the data at run time, this data is then manipulated by pandas using user input in the callback functions. Most of the scripts follow a similar structure.

1. Import libraries

```
from operator import index
from turtle import width

from matplotlib.pyplot import legend
import pandas as pd
import plotly.io as pio
import dash
import numpy as np
from dash import dcc, html, callback
import plotly.graph_objects as go
import dash_bootstrap_components as dbc
import plotly.express as px
from dash.dependencies import Input, Output, State
import datetime
import os
from dotenv import load_dotenv
from dash import dash_table
from datetime import date
from sqlalchemy import create_engine
from bertopic import BERTopic
from dash.exceptions import PreventUpdate
```

2. Load the env variables

```
load_dotenv()

#load in environment variables
Newswhip_Data_Path= os.getenv('Newswhip_Data_Path')
user = os.getenv('user')
password = os.getenv('password')
port = os.getenv('port')
host = os.getenv('host')
database = os.getenv('database').lower()
```

3. Load in the data

```
#register route for website
dash.register_page(__name__, path="/informtaion_trends")

# Load in the data
data = pd.read_csv(Newswhip_Data_Path,encoding='Latin1')
df = pd.DataFrame(data)

#The only columns currently necessary within the data frame.
NEEDED_COLUMNS=[
    'fb_data.total_engagement_count',
    'tw_data.tw_count',
    'pi_data.pi_count',
    'li_data.li_count',
    'source.publisher',
    'publication_timestamp',
    'MBFC_category',
    'date_time',
    'headline',
    'link',
    'excerpt']
```

4. Defined functions for later use.(usually dataframe manipulation functions given input)

```
# This function will cut the headlines down to four words as the headlines are very long.
def setupListForTick(dataFrame):
    temp = dataFrame['headline'].str.split(n=4).tolist()
    newList=[]
    for x in temp:
        newList.append(' '.join(x[0:4]))
    return newList

def setupListForTickDoc(dataFrame):
    temp = dataFrame['document'].str.split(n=4).tolist()
    newList=[]
    for x in temp:
        newList.append(' '.join(x[0:4]))
    return newList

#Organize list of values into a list of options with a matching label and value
def createListOfSets(optionsList):
    newList=[]
    for x in optionsList:
        newList.append({'label':x,'value':x})
    return newList

def createListOfSetsPercent(optionsList):
    newList=[]
    for x in optionsList:
        newList.append({'label':str(x)+'%', 'value':x})
    return newList

def createSourceOptions():
    sourceDF = MBFC_reference['source.publisher'].copy()
    sourceDF = sourceDF.sort_values()
    print(len(sourceDF))
    return createListOfSets(sourceDF.to_list())
```

5. Layout for the documents (HTML)

```
#HTML for the page that renders underneath the navbar
layout = html.Div([
    html.Div([
        html.H1("Publisher Engagement", style={'text-align': 'right'}),
        dbc.Button(
            "i",
            id="collapse-button",
            className="m-4",
            n_clicks=0,
            style={
                'width': '25px',
                'height': '25px',
                'border-radius': '100%',
                'margin': '20px 0 0 5px',
                'padding': '0',
                'font-family': 'Cambria, Cochin, Georgia, Times, Times New Roman, serif'
            }
        ),
    ], style={
        'display': 'flex',
        'justifyContent': 'center'
    }),
    dbc.Collapse(
        html.Div([
            dbc.Card([
                html.H4("Date Range", className="card-title"),
                html.P(
                    "Select start date and end date to filter",
                    className="card-text",
                ),
            ],
            className="mb-3",
            style={
                "width": "30%"
            }
        )
    )
])
```

6. Callback functions to change layout based on user input (update graphs)

```
@callback(
    [Output('cluster-engagement', 'figure'),
     Output('mbfc_engagement_fb', 'figure'),
     Output('mbfc_engagement_tw', 'figure')],
    [Input('switch-bert', 'value'),
     Input('select-cluster', 'value')]
)
def select_cluster_graphs(chosen_model, word):
    if word is None:
        raise PreventUpdate
    if chosen_model is None:
        raise PreventUpdate

    with open(f'models/Batch{chosen_model}/bert-topics', 'rb') as f:
        topics = np.load(f)

    data = pd.read_csv(f'Portions/Batch{chosen_model}.csv')

    cluster_DF = pd.DataFrame({
        'topic': topics,
        'document': data.headline,
        'document_id': data.uuid,
        'fb_engagement': data['fb_data.total_engagement_count'],
        'tw_engagement': data['tw_data.tw_count'],
        'mbfc': data.MBFC_category
    })
```

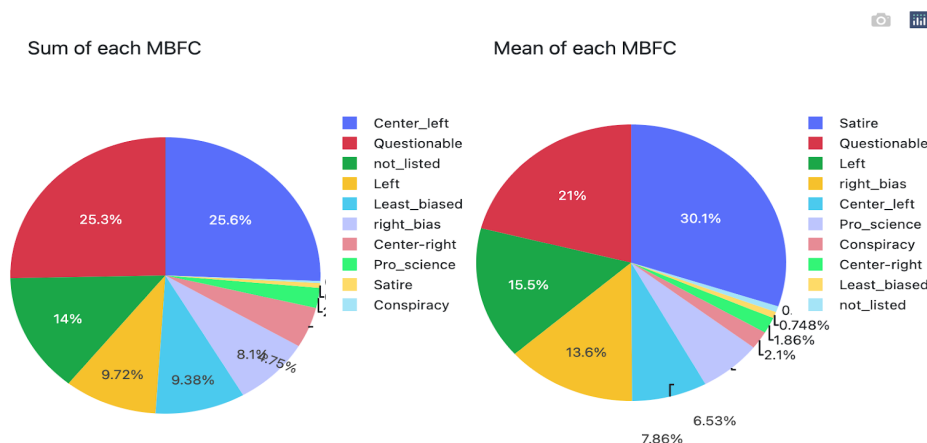
Each component has a title that corresponds to its function. It is tabbed in a parent child relationship. Callbacks target components through their id attribute. Classname is used by both the assets/navbar.css and bootstrap.

Other:

Although this page is titled Other it is very important to the project. As the information trends page was getting larger and more complex. We started working on experimental features in the Other page for ease of development. These features include a breakdown of the engagement rates users of facebook have with biases. The top graph was built using aggregated data compiled in the Data-Links folder.



This graph shows the level of engagement users had with climate news over time. This is done by finding the proportion of engagement with top climate news to engagement with top overall news in each month.



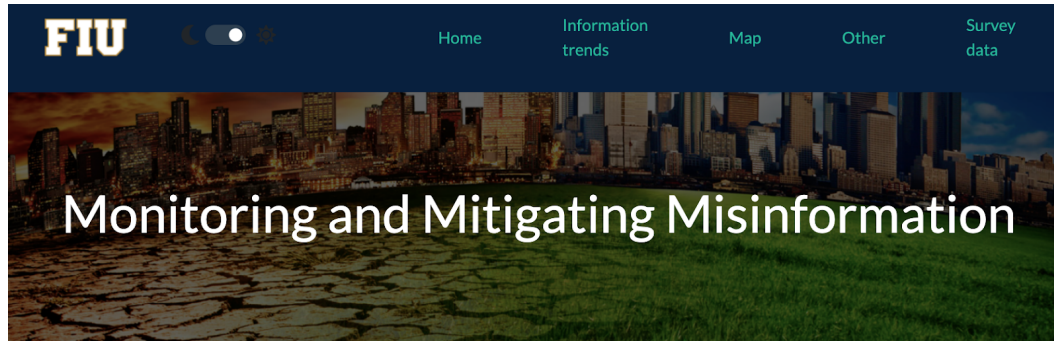
These graphs show the breakdown of user engagement with different biases. Using both a composition of the sum of engagement for that time period and the mean engagement. This is very revealing to the magnitude of the issue we are trying to solve.

The last portion allows users to look at the top performing articles within that group.

All of these graphs are created using plotly and basic dash components.

Home:

The home page features no callback functions. This page serves as only an informational exposition to the project. The information could be redone and explained more concisely but we will leave that to the next capstone. The structure and layout are there with a verbose description.



What is Monitoring and Mitigating Misinformation

The monitoring and mitigating information project aims to spread awareness about climate misinformation and the public perception of climate related issues. User's are able to interact with the engagement types of different sources and compare them visually with either a histogram or a bubble chart. There is also a chart comparing the engagement level of different articles from a publisher and links to the categorized articles below. Various climate disaster types can also be found on a United States map with the number of incidents separated by states. Estimates of the public's opinion on certain climate questions can also be seen on the survey data page

Our Data Sources

The survey data of climate opinions was conducted by Monitoring and Mitigating the Yale Program on Climate Change Communication. Misinformation scrapes the most and their most recent climate opinion map can be frequently visited news sources for

Our Process

Our Data Sources

sit amet, consectetur adipiscing elit. Ut enim ad elit do eiusmod tempor incididunt ut et dolore magna aliqua. Ut enim ad elit

Images like the one used in the background need to be loaded into an assets folder in order to be accessible to the application.

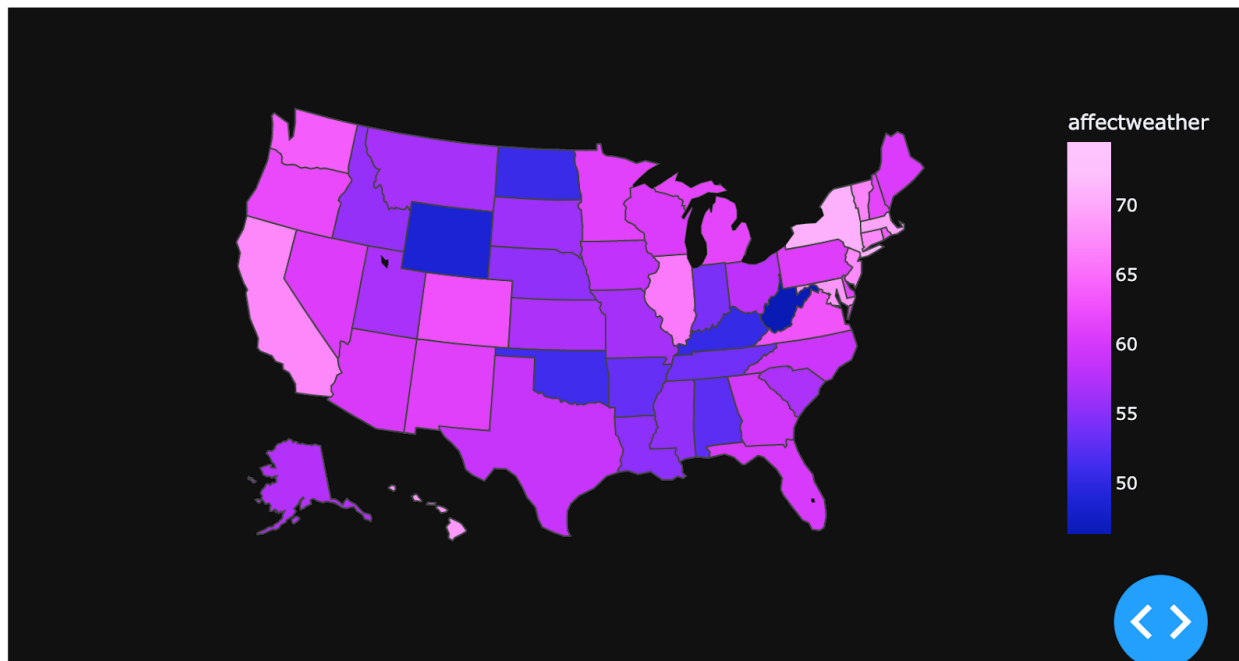
Survey Data:

This part of the application was built using Yale's survey data. It is meant for future correlation to news article data and fema data. As it is now it is nothing more than a version of the original yale climate survey map project. This part is not an original idea and needs to be properly credited to them if included. The data to be used by the project was given to us by one of the researchers at Yale in hopes to use it for machine learning or the like. We did not get far enough in this venture and kept the data in this form for ease of use and reference in future iterations.

Questions asked

Estimated percentage who somewhat/strongly agree that global warming is affecting the weather in the United States

2018 2019 2020



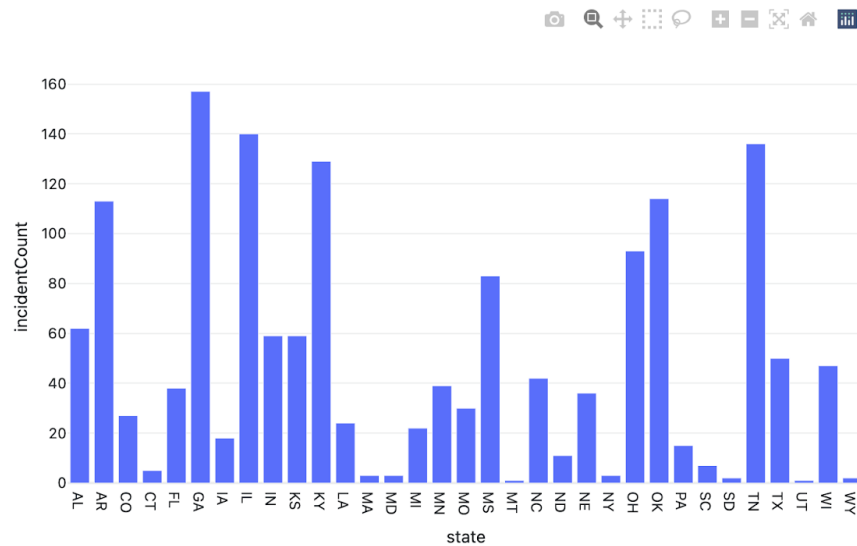
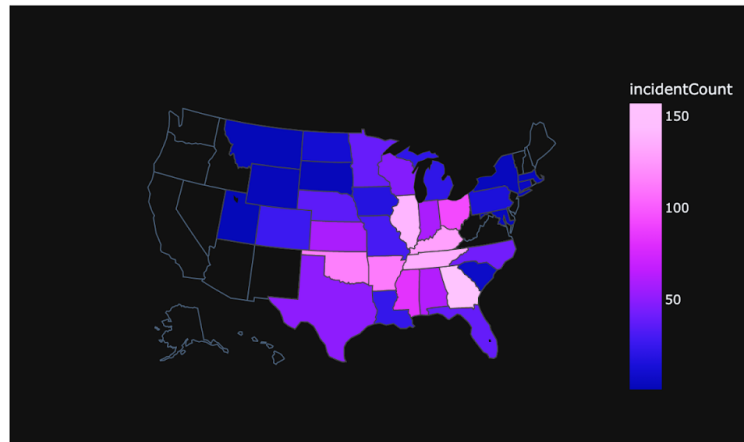
The graph is able to change dynamically using the question as input. The slider is also able to change dynamically using the question as input. Some questions were not asked in previous years.

FEMA Map:

This page shows the amount of disasters declared by FEMA in each state. The year is available in the data set and focusing on specific years or states should be relatively easy to do. As of right now the graph only shows a bar graph of total occurrences and an intensity map.

Disaster Type

Tornado



Scraper:

The scraper is used to get the article text from the internet. A very basic scraper will just get the html received from an http request to a specified url. A better scraper is able to parse that html for only the necessary information. Script tags and style tags don't have a lot of value for this work and neither do contact footers. Newspaper3k is a library recommended by the PO for scraping. BeautifulSoup(another scraping library) was also suggested but was quite slow and yielded information that was not very clean. The team built a scraper that would take the data frame and use multithreading to speed up the process. The script needs to be run on all 6,000,000 plus articles. It was executed on the data set for 2021-2022 but needs to be done to the datasets for 2017-2020. That info then needs to be stored in a database.

```
def scrape(list_of_urls,startP,endP,fileName):
    if endP == 'last':
        list_of_urls=list_of_urls[startP:]
    else:
        list_of_urls=list_of_urls[startP:endP]

    hourCount=1
    count=0
    rows = []
    # print(len(list_of_urls))
    for link in list_of_urls:
        count+=1
        try:
            a = Article(url="%s" % (link), language='en')
            a.download()
            a.parse()

            author = a.authors
            text = a.text
            title = a.title

            if (time.time()-start)>hourCount*UNIX_HOUR:
                print(f'{fileName}: {count} of {len(list_of_urls)} completed')
                hourCount+=1

            # print(f'\n\nNewspaper***\ntitle:{title}\n\ntext:{text}')
            text=text.replace('\n','')

            row = {'url':link,
                  'author':author,
                  'textNW':text,
                  'title': title}

            rows.append(row)
        except Exception as e:
            # print(e)
            row = {'url':link,
                  'author':'N/A',
                  'textNW':'N/A',
                  'title': 'N/A'}

            rows.append(row)

    df_v1 = pd.DataFrame(rows)
```

As it is currently it can collect the data for the remaining years in approximately 210 hours.

The scrape function basically tasks in a list of urls, loops through them and collects the relevant text.

The script will update you every 5 minutes on the progress. Of each thread. Each thread is given a portion of the dataframe to work with.

A Parallel scraper is included but it is not close to being complete and should not be run without knowledge and use of proper multiprocessing techniques.

The multithread one works fine though.

Often exceptions will occur; the script catches those and creates an "N/A" row when this occurs. These can be targeted

with another scraper for completeness sake.

BERTopic:

```
import pandas as pd
import numpy as np
from bertopic import BERTopic
import matplotlib.pyplot as plt

#Please change the Batch to your batch number
#READ THIS!!!
path = 'Portions/Batch6.csv'
data = pd.read_csv(path, encoding='Latin1', index_col=0)

topic_model = BERTopic()

print(len(data.headline.to_list()))
print(len(data.textNW.to_list()))

14001
14001

dataset=[]

for x,y in zip(data.headline.to_list(),data.textNW.to_list()):
    dataset.append(str(x)+str(y))

topics, probs = topic_model.fit_transform(dataset)
```

Located in TemplateBert/Train-model.ipynb. This script trains models. It takes the data from the headlines and the text from the articles to generate a model.

The model takes a long time to train. About 1 hour per 50,000 items. The one here took 3 hours. The model is then saved in rb format. Along with the topics.

The script is straight forward but the results were also not great. I believe it is a worthy endeavor, but the text from the articles needs to be cleaned up.

Words that have nan at the end when they shouldn't is an artifact of javascript. These words need to be fixed algorithmically.

Stop words also need to be taken out like "the" and "and". This will improve the model. Plus the amount of topics returned is too much. It should be limited to at most 100 per two month span.

Located in Website_DASH/notebooks/convert_to_figure.ipynb

```
chosen_model=6

topic_model=BERTopic.load(f'../models/Batch{chosen_model}/bert-model')

with open(f'../models/Batch{chosen_model}/bert-topics', 'rb') as f:
    topics = np.load(f)

data=pd.read_csv(f'../Portions/Batch{chosen_model}.csv')

headlines= data.headline.to_list()
date_times= data.date_time.to_list()

fig1=topic_model.visualize_topics()
fig1.show()

fig1JSON=fig1.to_json()
fig1file= open(f'../models/Batch{chosen_model}/visualize_topics.json','w')
fig1file.write(fig1JSON)
fig1file.close()
```

This script creates the figures seen in the website and saves them as json. The creation of these figures can take hours and loading them takes seconds, so it is best to save pretrained model figures for use.

WARNING: There are a lot of issues running BERTopic on a windows computer. Even loading them is not possible. There is a workaround that we will go into detail on in a later section.

```
def filterData(arrayOfColumns,columnToSortBy,divider):
    #array of columns are the columns you want in your new data frame
    #column to sort by example sort by engagement count or by alphabetical
    #percent of data you want
    copyOfDf= df[arrayOfColumns].copy()

    sortedData=copyOfDf.sort_values(by=columnToSortBy,ascending=False)

    return sortedData.head(int(len(sortedData)*(divider/100)))
def amountOfPublishers(dataFrame):
    sourcesOnly= dataFrame['source.publisher'].copy()
    sortedData= sourcesOnly.sort_values()
    noDupsData= sortedData.drop_duplicates()
    return noDupsData

def createListOfSets(optionsList):
    newList=[]
    for x in optionsList:
        newList.append({'label':x,'value':x})
    return newList
```

Image 11: Filter Data Functionality

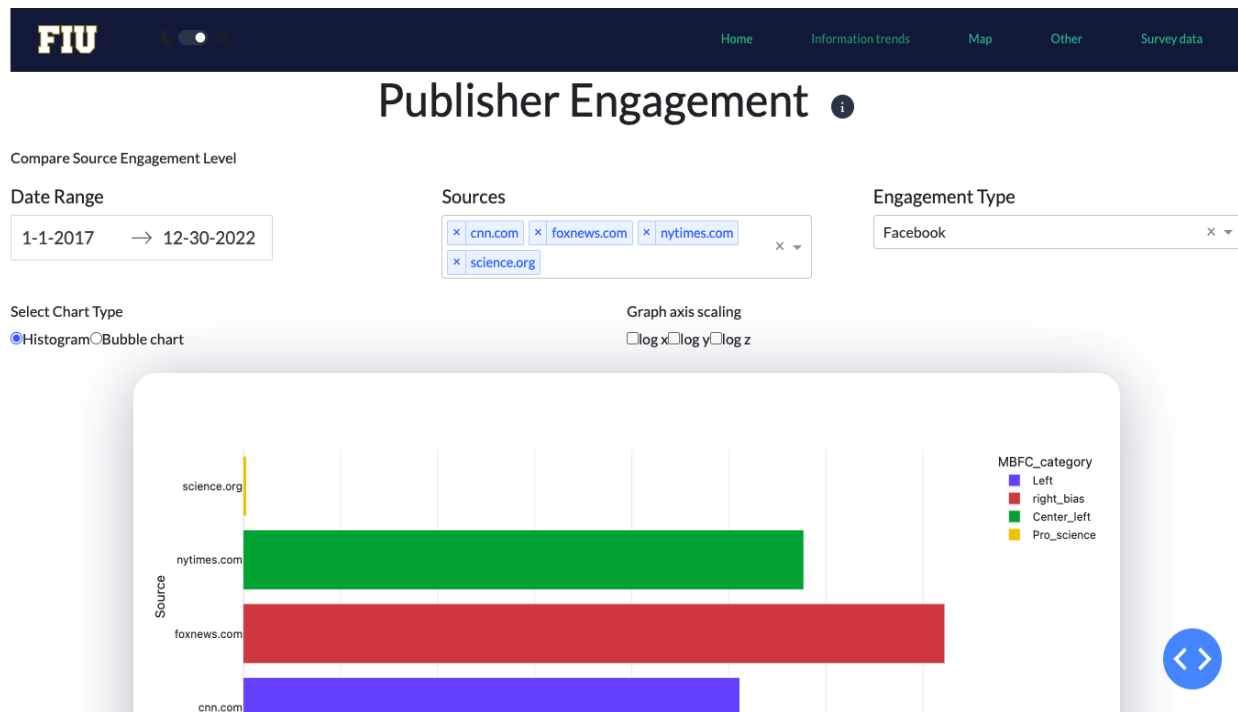


Image 12: Filtering dataframe interactive menu

```
def update_graph(source_selected,date_selected):

    #format container
    container = "Source(s): {}".format(source_selected)
    #load in dataframe
    GraphDF = setup_source_engagement(
        updatedDataFrame,
        datevalues[date_selected[0]],
        datevalues[date_selected[1]],
        source_selected)

    #create graph
    fig = px.histogram(GraphDF,
        x='fb_data.total_engagement_count',
        y='source.publisher',
        color='MBFC_category')

    #clean up axis title
    fig.update_xaxes(title_text='Engagement Count')
    fig.update_yaxes(title_text='Source')

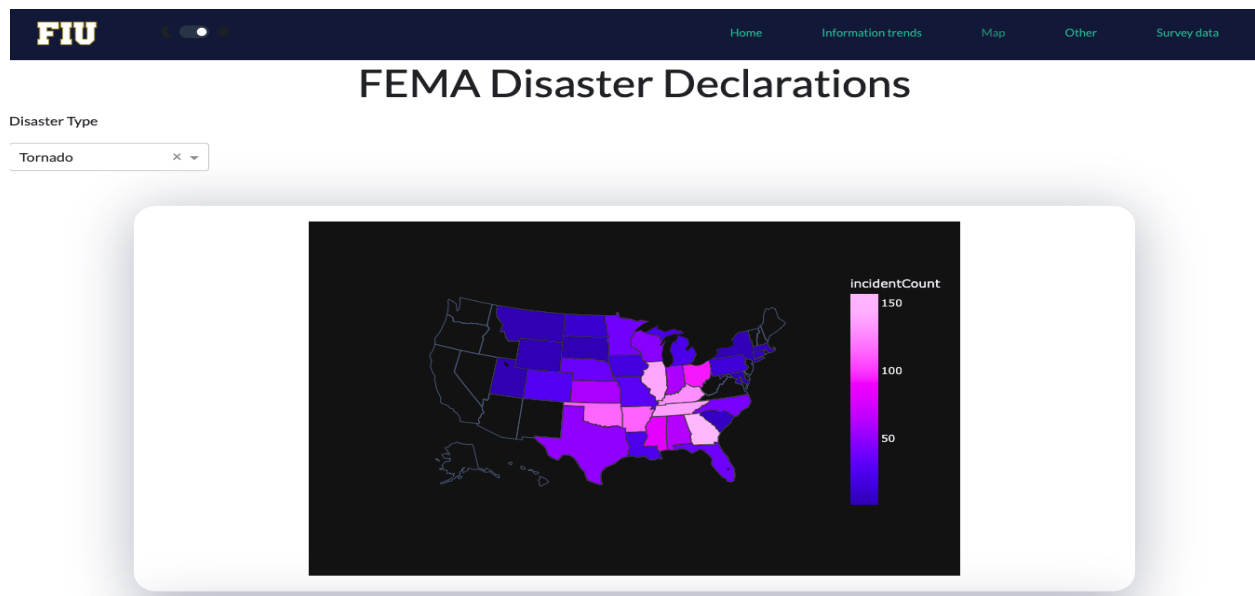
    return container, fig

#Do data frame manipulations here that do not need to be dynamic.
# Store them in variable so they are ready
# at load and do not need to be recalculated
updatedDataFrame=df[NEEDED_COLUMNS].copy() #we only need certain collumns
MBFC_reference= publisher_bias(updatedDataFrame) #makes a reference table
updatedDataFrame['date_time']=updatedDataFrame['date_time'].astype('datetime64[ns]')

#Completely sets up dataframe for source vs. engagement
def setup_source_engagement(dataFrame,time_start,time_end,source_selected):
    #filter by date
    tempFrame=dataFrame[(dataFrame['date_time'] > time_start) & (dataFrame['date_time'] < time_end)]
    #group by source and sum
    groupedData= tempFrame.groupby('source.publisher',as_index=False)['fb_data.total_engagement_count'].sum()
    #filter grouped data by chosen source
    groupedData = groupedData[groupedData['source.publisher'].isin(source_selected)]
    #merge table with MBFC_category
    groupedData= groupedData.merge(MBFC_reference,on='source.publisher',how='left')

    return groupedData
```

Image 14: Filtering dataframe interactive menu (Code)



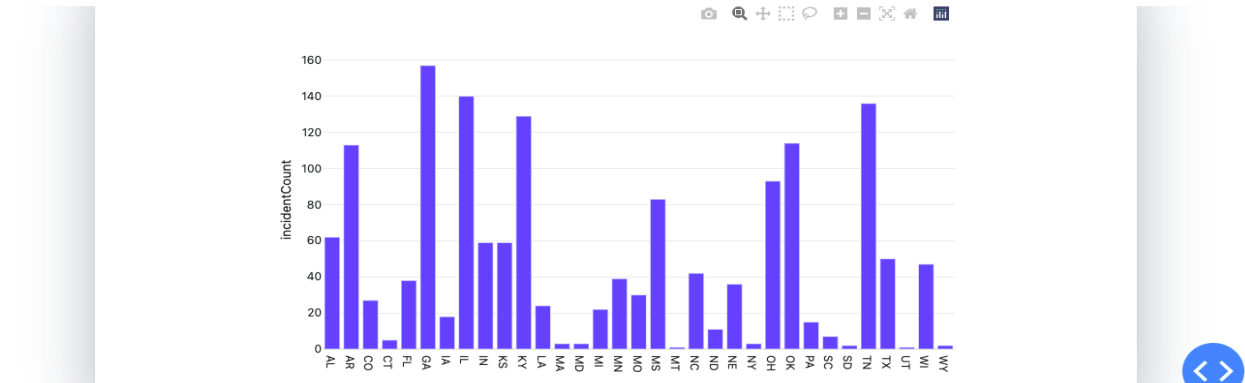


Image 15: Map & Graph

```

app.layout = html.Div([
    html.H1("FEMA Disaster Declarations", style={'text-align': 'center'}),
    html.P("Disaster Type", style={'margin-top': '5px', 'color': 'black'}),
    dcc.Dropdown(id="select_type",
        options=[{"label": "Tornado", "value": "Tornado"},
                  {"label": "Hurricane", "value": "Hurricane"},
                  {"label": "Flood", "value": "Flood"},
                  {"label": "Fire", "value": "Fire"},
                  {"label": "Severe Storms", "value": "Severe Storm(s)"},
                  {"label": "Drought", "value": "Drought"},
                  {"label": "Coastal Storm", "value": "Coastal Storm"},
                  {"label": "Snow", "value": "Snow"},
                  {"label": "Freezing", "value": "Freezing"},
                  {"label": "Severe Ice Storm", "value": "Severe Ice Storm"},
        ],
        multi=False,
        value="Tornado",
        style={'width': "40%"}),
    html.Div(id="output_container", children=[]),
    html.Br(),
    dcc.Graph(id="disaster_map", figure={})
])

def update_graph(option_selected):
    container = "Disaster Type: {}".format(option_selected)

    dff = disasters_data.copy()
    dff = dff[dff['incidentType'] == option_selected]

    fig = px.choropleth(
        data_frame=dff,
        locations='state',
        locationmode='USA-states',
        scope="usa",
        color='state',
        hover_data=['state', 'incidentCount'],
        color_continuous_scale=px.colors.sequential.YlGnBu,
        labels={'incidentType': 'Incident Type'},
        template='plotly_dark'
    )

    return container, fig

```

Image 16: Map Creation (Left code is html, right is graph)

```

app = dash.Dash(__name__)

app.layout = html.Div([
    html.H1("NewswhipPublishers", style={'text-align': 'center'}),
    html.P("Publisher", style= {'margin-top': '5px', 'color': 'black'}),
    dcc.Dropdown(id="select_publisher",
        options= dropDownPublishers,
        searchable=True,
        multi=False,
        value="cnn.com",
        style={'width': '40%'}),

    dcc.Dropdown(id="select_topic",
        options= dropDownTopics,
        searchable=True,
        multi=False,
        value="ALL",
        style={'width': '40%'}),

    dcc.Dropdown(id="select_percentage",
        options= dropDownPercentages,
        searchable=True,
        multi=False,
        value="10",
        style={'width': '40%'}),

    html.Div(id="output_container", children=[]),
    html.Br(),

    dcc.Graph(id="publisher_histog", figure={}),

])

@app.callback([
    Output(component_id='output_container', component_property='children'),
    Output(component_id='publisher_histog', component_property='figure')],
    [Input(component_id='select_publisher', component_property='value'),
    Input(component_id='select_topic', component_property='value'),
    Input(component_id='select_percentage', component_property='value')])

def update_graph(option_selected, topic_selected, percentage_selected):

    container = "Publisher Source: {}".format(option_selected)
    print(option_selected)
    print(topic_selected)
    print(percentage_selected)
    dff = prelimEngageDF

    ###Splice by source
    dff = dff[dff['source.publisher'] == option_selected]

    ###Splice by keyword
    if(topic_selected != 'ALL'):
        dff = dff[dff['keywords'].astype(str).str.contains(topic_selected)]

    print(dff['keywords'])
    dff = dff.sort_values(by='fb_data.total_engagement_count', ascending=False)
    dff = dff.head(int((len(dff)*(int(percentage_selected)/100))))

    fig = px.histogram(
        dff,
        y='headline',
        x='fb_data.total_engagement_count',
    )

    fig.update_xaxes(title_text='Engagement Count')
    fig.update_yaxes(title_text='Article')

    fig.update_traces(texttemplate="%{y}")

    return container, fig

```

Image 17: Histogram Script

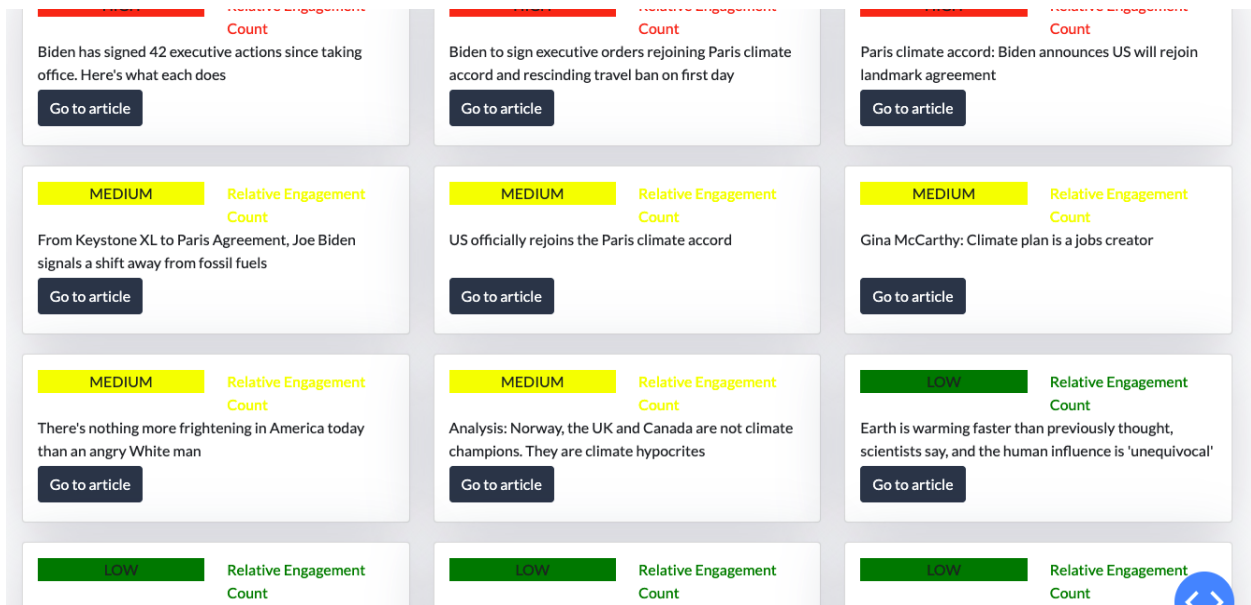


Image 18: Article cards

```

def card_main(title, excerpt, link, index, listLength ):
    color='green'
    tag='LOW'
    if index/listLength< 0.10:
        color='red'
        tag='HIGH'
    elif index/listLength< 0.30:
        color='yellow'
        tag='MEDIUM'

    return dbc.Card(
    [
        dbc.CardBody(
            [
                dbc.Row([
                    dbc.Col([
                        html.Div(
                            tag,
                            style={
                                'backgroundColor':color,
                                'textAlign':'center'
                            })
                    ]),
                    dbc.Col([
                        html.Div(
                            'Relative Engagement Count',
                            style={
                                'color':color
                            })
                    ])
                ]),
                html.P(
                    title,
                    className="card-text",
                    style={
                        'padding-bottom':'30px',
                    },
                ),
                dbc.Button("Go to article",href=link, color="primary",style={
                    'position':'absolute',
                    'bottom':'20px',
                    'margin-top':'20px'
                }),
            ],
        ),
    ],
    class_name='cardBody'
)

```

Image 19: Cards Script



Image 20: Line Graph

```
def CreateAggregates(filename):
    #Read Csv and index by date_time
    df = pd.read_csv(filename,encoding='Latin1')
    df = df[NEEDED_COLUMNS].copy()
    df.index = pd.to_datetime(df['date_time'])

    #group data by month and sum up articles that occurred each month using .count()
    countDF= df.groupby(pd.Grouper(freq='M')).count()
    countDF= countDF[['date_time']].copy()
    countDF.rename(columns={'date_time':'articlecount'},inplace=True)

    #group data by month and sum up engagement on a separate DataFrame
    df= df.groupby(pd.Grouper(freq='M')).sum()
    #Merge the two data frames
    df=df.join(countDF)

    return df
```

Image 21: Line Graph Script



Image 22: BERTopic graphics

PROJECT INFORMATION

Sprint Review Reports

Sprint 1 Review Meeting Minutes

Attendees:

- Roberto Lopez-Trigo (RL) **(Capstone II Leader)**
- Colleene Mccurdy (CM)
- Gabriel Prieto (GP)
- Javier Oliva (JO)
- Kyle Ghani (KG)
- Brian Rodriguez (BR)
- Nathan Chiche (NC) **(Capstone I Leader)**
- Kyle Zimmer (KZ)
- Jeffrey Quispe (JQ)
- Khizar Nasir (KN)

Start time: 7pm

End time: 8pm

Notes: Individual Responsibilities have been crafted for each team member within each team. Each of these will be distributed during the planning meeting. The concepts and UI were all green lit. Each team this week was mostly in charge of understanding and researching their role and how they could add to the project.

Sprint User Stories:

- User Story 1 - Database Team
 - Colleene Mccurdy (CM)
 - Brian Rodriguez (BR)
 - Kyle Zimmer (KZ)
 - Kyle Ghani (KG)

You guys are in charge of creating a database that the application can query from. The implementation is up to you, but it needs to be secure. Research best practice for creating a secure database

- User Story 2 - Efficiency Team
 - Gabriel Prieto (GP)
 - Javier Oliva (JO)
 - Kyle Ghani (KG)
 - Jeffrey Quispe (JQ)

Look into how to improve current code and project features to make it more efficient. You will also work on features, but with an emphasis on efficiency, speed and reliability. Research best practice for creating efficient dashboards

- User Story 3 - Feature Team
 - Gabriel Prieto (GP)
 - Nathan Chiche (NC)
 - Roberto Lopez-Trigo (RL)
 - Brian Rodriguez (BR)
 - Khizar Nasir (KN)

Implement User Interface components that have been accepted by Sam. Add features and functions that are discussed in meetings/ delegated to you. Research best practice for creating analytical tools

- User Story 4 – UI/UX Team
 - Roberto Lopez-Trigo (RL)
 - Javier Oliva (JO)
 - Khizar Nasir (KN)

Develop the User interface and the user experience using Figma. Research best practice for creating analytical tools. Look into how to visualize the data. And how to best interact with it.

Sprint 2 Review Meeting Minutes

Attendees:

- Roberto Lopez-Trigo (RL) **(Capstone II Leader)**
- Colleene Mccurdy (CM)
- Gabriel Prieto (GP)
- Javier Oliva (JO)
- Kyle Ghani (KG)
- Brian Rodriguez (BR)
- Nathan Chiche (NC) **(Capstone I Leader)**
- Kyle Zimmer (KZ)
- Jeffrey Quispe (JQ)
- Khizar Nasir (KN)

Start time: 8pm

End time: 9:30pm

Notes: Individual Responsibilities have been crafted for each team member within each team. Each of these will be distributed during the planning meeting. The concepts and UI were all green lit. Each team this week was mostly in charge of understanding and researching their role and how they could add to the project.

Sprint User Stories:

- User Story 1 - Database Team
 - Colleene Mccurdy (CM)
 - Brian Rodriguez (BR)
 - Kyle Zimmer (KZ)
 - Kyle Ghani (KG)

Task: Database created and deployable waiting for ssh ability into FIU's remote server to deploy it and test with the application.

- User Story 2 - Efficiency Team
 - Gabriel Prieto (GP)

- Javier Oliva (JO)
- Kyle Ghani (KG)
- Jeffrey Quispe (JQ)

Task: Current code has been looked into and improved currently working on scraping articles and setting up queries for when the database gets deployed at FIU.

- User Story 3 - Feature Team
 - Gabriel Prieto (GP)
 - Nathan Chiche (NC)
 - Roberto Lopez-Trigo (RL)
 - Brian Rodriguez (BR)
 - Khizar Nasir (KN)

Task: Implemented new charts such as the bubble chart and the 3d bubble chart. Needs to add more filtering options for the user, Such as sort MBFC, fake and political.

- User Story 4 – UI/UX Team
 - Roberto Lopez-Trigo (RL)
 - Javier Oliva (JO)
 - Khizar Nasir (KN)

Task: Implement UI/UX decisions made namely the homepage and source vs engagement histogram.

- User Story 5 – Machine Learning Team
 - All members

Task: This new team is responsible for creating models from the data we have accumulated to help mitigate misinformation and gain inference from the database.

Sprint 3 Review Meeting Minutes

Attendees:

- Roberto Lopez-Trigo (RL) **(Capstone II Leader)**
- Colleene Mccurdy (CM)
- Gabriel Prieto (GP)
- Javier Oliva (JO)
- Kyle Ghani (KG)
- Brian Rodriguez (BR)
- Nathan Chiche (NC) **(Capstone I Leader)**
- Kyle Zimmer (KZ)
- Jeffrey Quispe (JQ)
- Khizar Nasir (KN)

Start time: 8pm

End time: 9:30pm

Notes: Individual Responsibilities have been crafted for each team member within each team. Each of these will be distributed during the planning meeting. The concepts and UI were all green lit. Each team this week was mostly in charge of understanding and researching their role and how they could add to the project.

Sprint User Stories:

- User Story 1 - Database Team
 - Colleene Mccurdy (CM)
 - Brian Rodriguez (BR)
 - Kyle Zimmer (KZ)
 - Kyle Ghani (KG)

Task: Made the database more efficient by removing found duplicates and further cleaning the data.

- User Story 2 - Efficiency Team
 - Gabriel Prieto (GP)
 - Javier Oliva (JO)
 - Kyle Ghani (KG)
 - Jeffrey Quispe (JQ)

Task: Checked and fixed up the newly added code to make sure it is running as efficiently as possible.

- User Story 3 - Feature Team
 - Gabriel Prieto (GP)
 - Nathan Chiche (NC)
 - Roberto Lopez-Trigo (RL)
 - Brian Rodriguez (BR)
 - Khizar Nasir (KN)

Task: Using the newly acquired Yale dataset, worked and discussed the best ways of implementing it to our product.

- User Story 4 – UI/UX Team
 - Roberto Lopez-Trigo (RL)
 - Javier Oliva (JO)
 - Khizar Nasir (KN)

Task: Removed the bubble chart due to adjusted findings and working on continued improvement to the homepage to make it look more visually presentable.

- User Story 5 – Machine Learning Team
 - All members

Task: Made improvements to the scraper that is being used and using technologies like regex to further iterate on it to make it better.

Sprint 4 Review Meeting Minutes

Attendees:

- Roberto Lopez-Trigo (RL) **(Capstone II Leader)**
- Colleene Mccurdy (CM)
- Gabriel Prieto (GP)
- Javier Oliva (JO)
- Kyle Ghani (KG)
- Brian Rodriguez (BR)
- Nathan Chiche (NC) **(Capstone I Leader)**
- Kyle Zimmer (KZ)
- Jeffrey Quispe (JQ)
- Khizar Nasir (KN)

Start time: 8pm

End time: 9:30pm

Notes: Individual Responsibilities have been crafted for each team member within each team. Each of these will be distributed during the planning meeting. The concepts and UI were all green lit. Each team this week was mostly in charge of understanding and researching their role and how they could add to the project.

Sprint User Stories:

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

User stories have been changed to more accurately reflect the current scope of the sprint from planning meeting.

- User Story 1 - Scraper (Gabriel & Roberto)
- User Story 2 - Database Queries (Colleene, Brian, and Roberto)
- User Story 3 - Anomaly detection (Brian, Roberto, Colleene, Gabriel, Javier) IQR
- User Story 4 - Date slider (Done)
- User Story 5 – Theme changer (Khizar, Nathan, Jeffrey)
- User Story 6 - General UI adjustments to overall theme. (Khizar, Nathan, Jeffrey)
- User Story 7 - Survey data UI (In a week)
- User Story 8 - Proportion to overall engagement (Roberto, KyleG)
- User Story 9 - Informational text box (Javier, KyleZ, Roberto)
- User Story 10 - Machine learning model (Brian and Colleene)
- User Story 11 - Sql injection attack (Roberto, Kyle G, Colleene)
- User Story 12 - Automate setup environment
 - Requirements txt file

User Story 1 - Scraper

Description: Scraper tool needs to pull articles from the internet using our list of five million links for the past 5 years. We will start with a sample of 50. Once a sample of 50 starts working with. We will extract for 6 months.

- Members involved: Roberto and Gabriel
- **Progress: Scraper runs effectively with 1000 entries, but failed during long run due to unforeseen edge cases. Scraper has been re-written using multi threading and a focus on small batches of 100,000.**

User Story 2 - Database Queries

Description: Database was created. We need to create the queries that will replace the pandas functions which splice static versions of the data.

- Members involved: Roberto, Brian, Colleene
- **Progress: Query Logic is complete for the current features of the database.**

User Story 3 - Anomaly detection

Description: Detect articles that are trending a lot more than the norm. 1-2 standard deviations away from the mean within the month or week.

- Members involved: All members
- **Progress: None yet.**

User Story 4 - Dynamic Date slider for survey data

Description: Date range slider ui that allows the user to input the year in which they would like to view the survey data. This needs to dynamically render based on the survey question selected.

- Members involved: Jeffrey, Roberto
- **Progress: Completed**

User Story 5 – Theme changer (light mode and dark mode)

Description: Change between light mode and dark mode. Keep note of user preference.

- Members involved: Jeffrey, Roberto, Kyle Z
- **Progress: Completed and bugs have been squashed**

User Story 6 - General UI adjustments to overall theme.

Description: A more cohesive theme to the product is needed and must be programmed in a way that works with the theme changer.

- Members involved: Khizar, Javio, Jeffrey, and Nathan
- Progress: Complete

User Story 7 - Survey data UI

Description: Create The ui on figma for the survey data obtained from Yale.

- Members involved: Khizar, Javio and Nathan
- **Progress: Mid Fi is in progress.**

User Story 8 - Proportion to overall engagement

Description: Tally up the proportion of engagement articles get compared to the engagement that top performing articles get

- Members involved: Khizar, Javio and Nathan
- **Progress: Proportions have been collected. Need to be added to the database.**

User Story 9 - Informational text box

Description: Component that expands with information when the user clicks it.

Members involved: KyleZ and Javio

Progress: Component was created and added to the necessary places. Text for the project needs to be written out.

User Story 10 - Machine learning model

Description: Use the data to create a useful machine learning model

Members involved: Colleene, Brian

Progress: BertTopic model successfully created and will be added to the main project.

User Story 11 - Sql injection attack

Description: Scan the queries and create a function that prevents sql injection attacks through sanity checks of user input.

Members involved: Kyle G, Roberto

Progress: Sanitizing functions for specific queries have been identified and need to be written out.

Sprint 5 Review Meeting Minutes

Attendees:

- Roberto Lopez-Trigo (RL) **(Capstone II Leader)**
- Colleene Mccurdy (CM)
- Gabriel Prieto (GP)
- Javier Oliva (JO)
- Kyle Ghani (KG)
- Brian Rodriguez (BR)
- Nathan Chiche (NC) **(Capstone I Leader)**
- Kyle Zimmer (KZ)
- Jeffrey Quispe (JQ)
- Khizar Nasir (KN)

Start time: 8pm

End time: 10:30pm

Notes: Individual Responsibilities have been crafted for each team member within each team. Each of these will be distributed during the planning meeting. The concepts and UI were all green lit.

Sprint User Stories:

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

User stories have been changed to more accurately reflect the current scope of the sprint from planning meeting.

- User Story 1 - Scraper (Gabriel & Roberto & Jeffrey & Brian)
- User Story 2 - BertTopic (Brian and Colleene and Roberto)
- User Story 3 - Informational text box (Javier,Jeffrey,KyleZ)
- User Story 4 - Anomaly detection (Brian, Roberto, Colleene, Gabriel, Javier)
- User Story 5 - General UI adjustments to overall theme. (Javier, Roberto,Jeffrey,KyleZ)
- User Story 6 - MBFC Graph (Colleene, Brian, and Roberto)???
- User Story 7 - Proportion to overall engagement (Roberto & Brian)
- User Story 8 - Sql injection attack (Roberto, Kyle G, Colleene)
- User Story 9 - Automate setup environment ()

User Story 1 - Scraper

Description: Scraper tool needs to pull articles from the internet using our list of five million links for the past 5 years. We were able to pull the articles for the 2021-2022. 2017-2020 still needs to be pulled.

- **Progress: Scraper has been rewritten to use multithreading and optimized. The scraper is able to pull 100,000 articles every three hours. A version of this should be made to use multiple cores(parallel processing). We created a shell of it. If this is employed it may be able to pull (100000 articles times the number of cores per hour.)**

User Story 2 - BERTopic

Description: Machine learning model that allows us to see the clusters of topics that occur with the extracted text from the scraper.

- **Progress: Bert models have been made and need to be added to the ui to change dynamically. The ability to focus on a cluster needs to be employed.**

User Story 3 - Informational text box

- Description: Describe what each section is about and how to use each part of the program in select boxes.

- **Progress: We added text to most of the text boxes but there are one or two that we need to add after user feedback was given.**

User Story 4 - Anomaly Detection

- Description: Use IQR and z score to find the outliers within the data set.
- **Progress: Finished this examination of the data but it did not match up with what the PO wanted. This needs to be re-done with focus on seeing anomalies that occur within the data, such as derivations in mean engagements with sources, topics, or bias groups.**

User Story 5 – General UI adjustments to overall theme

Description: Clean up UI as spacing is inconsistent in some places. Finish UI for BERTopic

- **Progress: UI has improved a lot. The BERTopic section could still use a lot more work.**

User Story 6 - MBFC Graph

Description: Graph that looks at mean engagement and total engagement of each MBFC category.

- **Progress: Complete**

User Story 7 - Proportion to overall engagement

Description: Create a line graph that looks at the proportion Climate of engagement to overall engagement. Note: a version of this should be done for each MBFC group.

Progress: Complete

User Story 8 - SQL injection attack

Description: Tally up the proportion of engagement articles get compared to the engagement that top performing articles get

- **Progress: Proportions have been collected. Need to be added to the database.**

User Story 9 - Automate setup environment

Description: Found a python package that can generate a requirements file for easy installation next semester.

Progress: Complete

Among many more...

Not many user stories have been created for this sprint, we more or less conducted a final meeting laying out future plannings and wishlist with the Product Owner. In regards as to what exactly is expected, you will find this list in the shortcomings/wishlist presentation video OR near the end of the documentation.

Installation / Maintenance

Running website on mac

Step 1: Install Git. Install python. Install VScode.

Step 2: Within the project directory Website_DASH:

- Create a virtual environment using
Python -m venv myenv
- source the virtual environment
source myenv/bin/activate
- On windows it is: source myenv/Scripts/activate

Step 3: Use an IDE, preferably Visual Studio Code.

Install the required packages using:

```
pip install -r requirements.txt
```

This step takes a while.

Step 4: Run the project using:

```
python3 app.py
```

Running Website on windows

Same as above except the BERTopic section may not be runnable. If that's the case. Comment out the BERTopic sections in the project before running.

This issue popped up in the last week and no solution was found yet. There was a workaround being attempted using the hdbscan wheel in the project, but it was not solved. Worst case the project can be run on a linux virtual machine or the bert sections can be commented out until it is solved.

Shortcomings

The app was built from scratch only using some of the individual scripts from last semester. The app works well but needs a lot of work. Communication was there but I would like to stress to the next team lead that deadlines and descriptive user stories will help the team a lot. Sometimes describing a feature or concern in a meeting works well but when there is little in writing it becomes harder for the team to understand how to tackle the problem.

Start the documentation template and outline the project goal in a formal software engineering report as soon as the semester starts in order to have a clear goal for the team. Not having one from the start leads to a lot of confusion. More could have been completed for our team had we been more organized and clear in our project goals.

Wishlist

- Room for front-end development. As of yet, other objectives must be satisfied to approach this part of the project.
 - Having the data linked between visuals that are structured in tiers.
 - Tier one: source vs engagement
 - Tier two: articles from a source vs engagement
 - Tier three: Article vs claims made
 - Wire frame this to see how best to show this tier system.
- Create an automated pipeline that pulls data at regular time intervals from NewsWhip.
 - Create a portal that allows for admin upload of csv files from Ayse(PO)
 - Portal can be made using the django admin page that was created in the github repo.
- Including high level analysis functions to show correlation between data sets.
 - On Covaxxy, they correlate between political orientation and vaccination rates.
- Find a way to show how the data is pulled.
 - Look into component architecture for clean implementation of this
 - Possible implementation: modal box, question mark, diagram that shows how data is pulled, but very open to suggestions. Be creative here.
 - (Overall we need to show data transparency), EX: New york times, Covaxxy, Wall street journal graphics
- Organize the data into a bonafide database. Currently, the dashboard is loading using static data in the project folder.
 - Pull the data from the database that could be used in the dashboard.
 - Create tables within the database that maximize the efficiency of the dashboard.
 - Incorporate claims data into the dashboard (PO will let you know more about this)..
- Form a future ideation regarding the exploratory analysis for the project.

REFERENCES

Sources of Data:

<https://mediabiasfactcheck.com/>

<https://www.fema.gov/about/openfema/api>

Documentation and Software:

<https://plotly.com/python/histograms/>

<https://www.codespeedy.com/pandas-groupby-sort-in-python/>

<https://dash.plotly.com/dash-core-components/dropdown>

<https://balsamiq.com/>

Tutorials:

Django Plotly Dash: <https://www.youtube.com/watch?v=psvU4zwO3Ao&list=WL&index=121>

Github:

<https://www.youtube.com/watch?v=RGOj5yH7evk>

Django: <https://www.youtube.com/watch?v=VBqJ0-imSMU>

Inspiration:

<https://osome.iu.edu/tools/covaxxy>

<https://arxiv.org/abs/2112.01379>

<https://www.climatechangecommunication.org/resources/>

<https://covid19obs.fbk.eu/#>