# CITIZENS' VOICE

Author: Michael Dubem

*"The people have the power. All we have to do is awaken the power in the people."*

- John Lennon

# Content

## 1. Introduction

The opening quote on the cover page by John Lennon embodies the entire purpose behind this project. Democracy which is the most popular form of government amongst the human race (for good reason) is least susceptible to perversion, only when those who wield the highest office in the land (the office of the citizen) begin to take their roles seriously. There is a saying in the Igbo culture (a tribe amongst many others in Nigeria) which states "Ala adi mma bu uru ndi nze", and translates to "A failing nation is to the benefit of the elites". Nigeria amongst many countries suffer so much from the perversion of her democracy, and the perseverance of many of these perversions can be traced to the dormancy (or in another context malleability) of her citizens. I dream of a day when her people will wake up and resist this perversion.

For those occupying the office of the citizen (like any other public office) to be successful, they need to be equipped with the necessary factors. This is what this idea aims to solve. This idea hinges on the notion that there are four major factors that the people need for a democracy to work in their favour, which are;

- Education: the people need to be educated enough to understand what binds a society and the philosophy behind it. They need to understand the importance of their unity, what it takes to achieve it and what power lies within. They need to be aware of their laws, constitutions, and have access to all information about their institutions. Arguably the most important of all, they need to know their history. "Those who cannot remember the past are condemned to repeat it." - George Santayana.

- Courage: In countries like Nigeria, so much is perpetrated on the people with the use of intimidation. It is one thing to be educated enough to understand the wrong in injustice, it is another thing to be brave enough to stand up in the face of injustice. "Nothing strengthens authority so much as silence." - Leonardo da Vinci

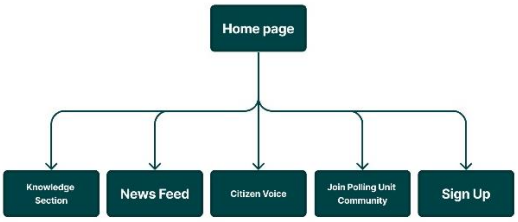
- Unity: is arguably the most important of these factors as this is where the entire power of the people lie, in their unity. A people who have united above all their differences are a formidable people. As much as the individuality of the human species can be seen as its greatest strength, it can also be seen as its greatest weakness. "When there is no enemy within, the enemy outside can do us no harm." – African Proverb

- Coordination/Direction: As important as it is for any movement or network to be decentralized, there still needs to be some level of central coordination/direction so as to have this power pointing in the right direction and with efficiency.

## Proposed Solution

The Citizens' Voice platform aims to serve as a one stop solution to the absence of these factors stated above. The Citizen's Voice Platform is intended to provide citizens access to all necessary information needed to be active citizens in their country, help them coordinate during necessary times (like elections and when the demand for justice is necessary), help them rate their public office holders regularly so as to keep them on their toes, and breach the gap between their voices and the ears of well-meaning public office holders. This platform comprises of six major sections (with the possibility of expansion during the detailed planning and development phase), which are; The Homepage, Knowledge Section, Politicians' Profile, Polling Unit Community, Election Observer, Citizens' Voice/Sentiment.

- Home Page: This section of the platform is meant to be a summary of the entire platform.




This section of the platform will also serve as a link to all other sections of the platform, for quick navigation.

*Figure 1. Sitemap of the Home Page.*

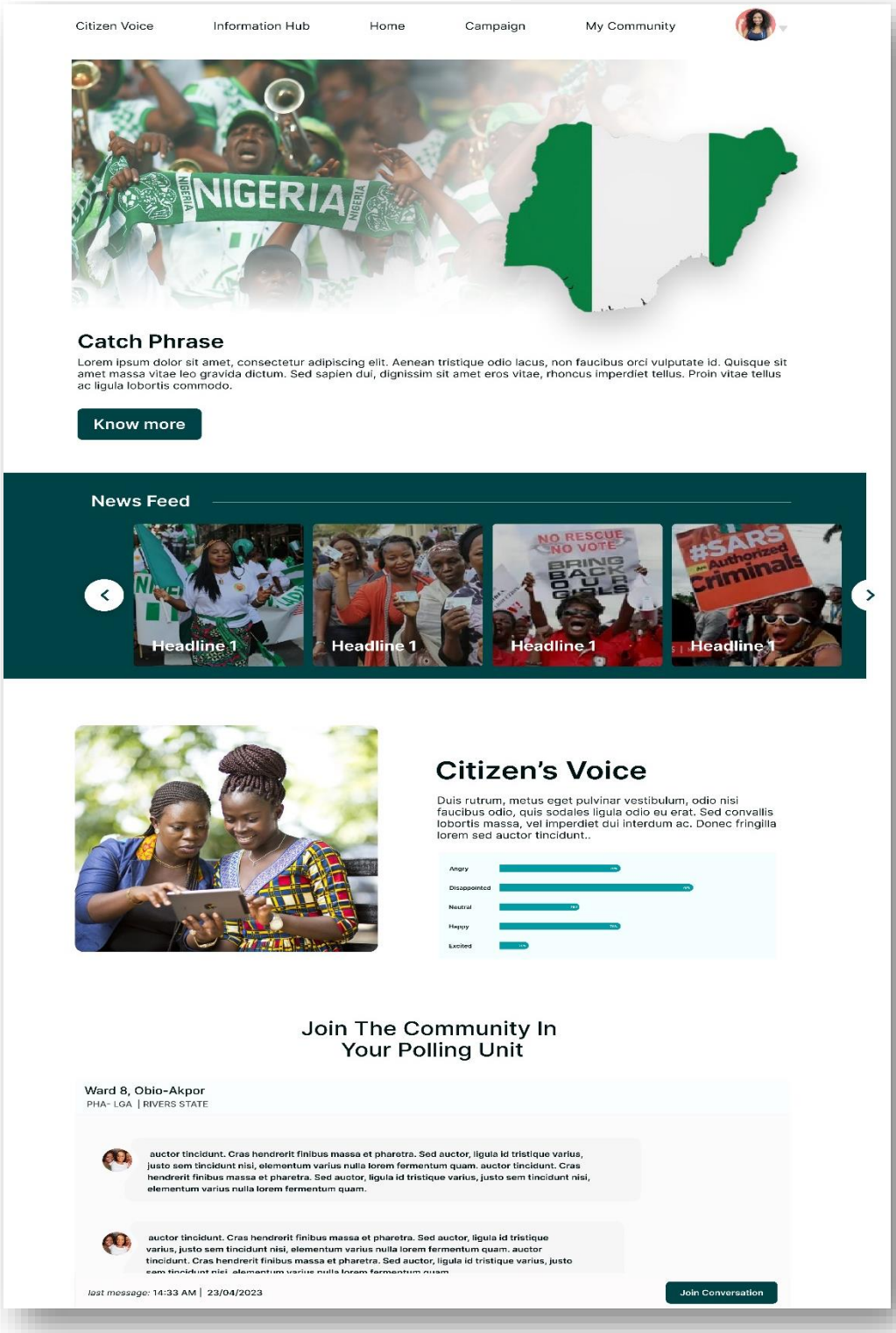
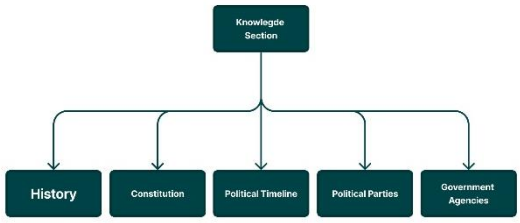



*Figure 2. Cropped version of the Home Page UI Design*

- <u>The Knowledge Section</u>: which also double as the information section will be amongst the largest sections of the platform as it will hold all the necessary information to educate the citizens on their current affairs, laws, constitution, institutions, public office holders, political timelines, and many others to help combat civic illiteracy.




This section will also have an integrated news section which will provide the citizens with breaking news, economic and international news on relevant issues.

*Figure 3. Sitemap of the Knowledge Section*

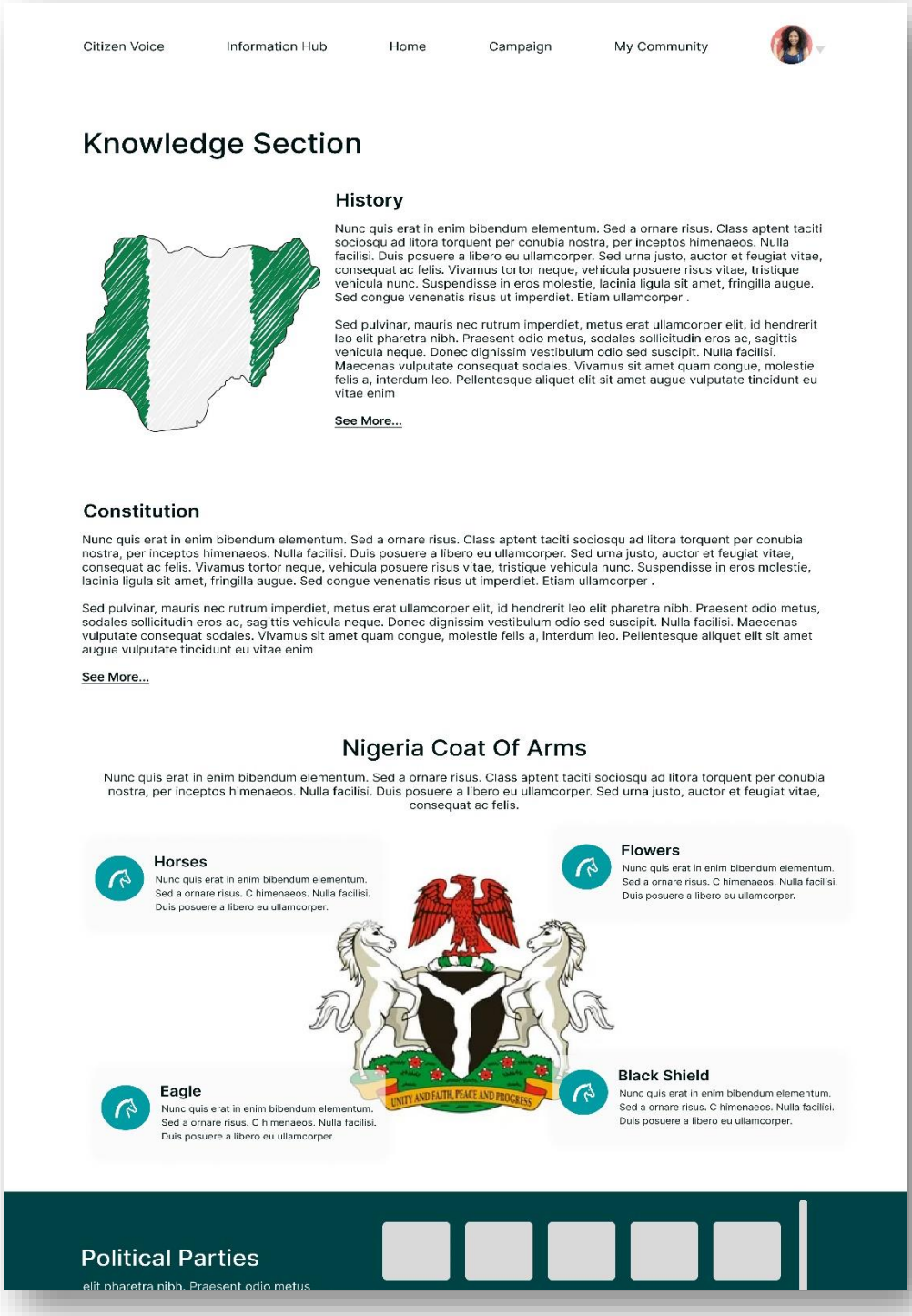



*Figure 4. Cropped Version of the Knowledge Section UI Design. (1)*


*Figure 5. Cropped Version of the Knowledge Section UI Design. (2)*

- Politicians' Profile: is an essential section of the platform will contain information about all public office holders, their affiliation, engagement and history. This will help promote accuracy of information on individuals who are in public service for the benefit and reference of citizens.

The citizens will also be provided a means to rate their public office holder using regular online polls, to express their (dis)satisfaction.

These ratings will be recorded over time for reference.

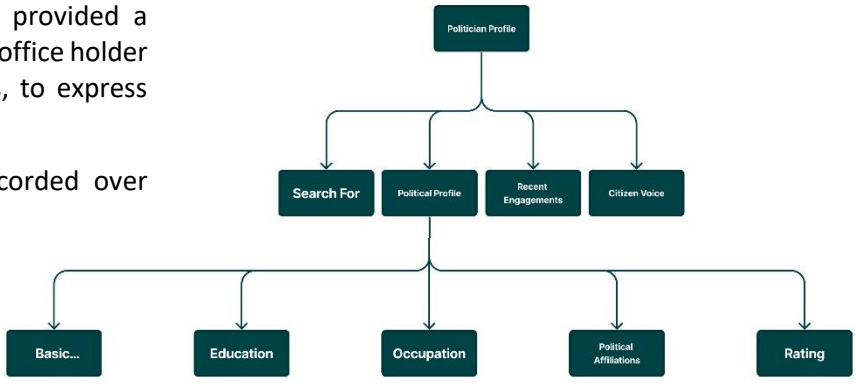



*Figure 6. Sitemap of the Politicians' Profile Section.*

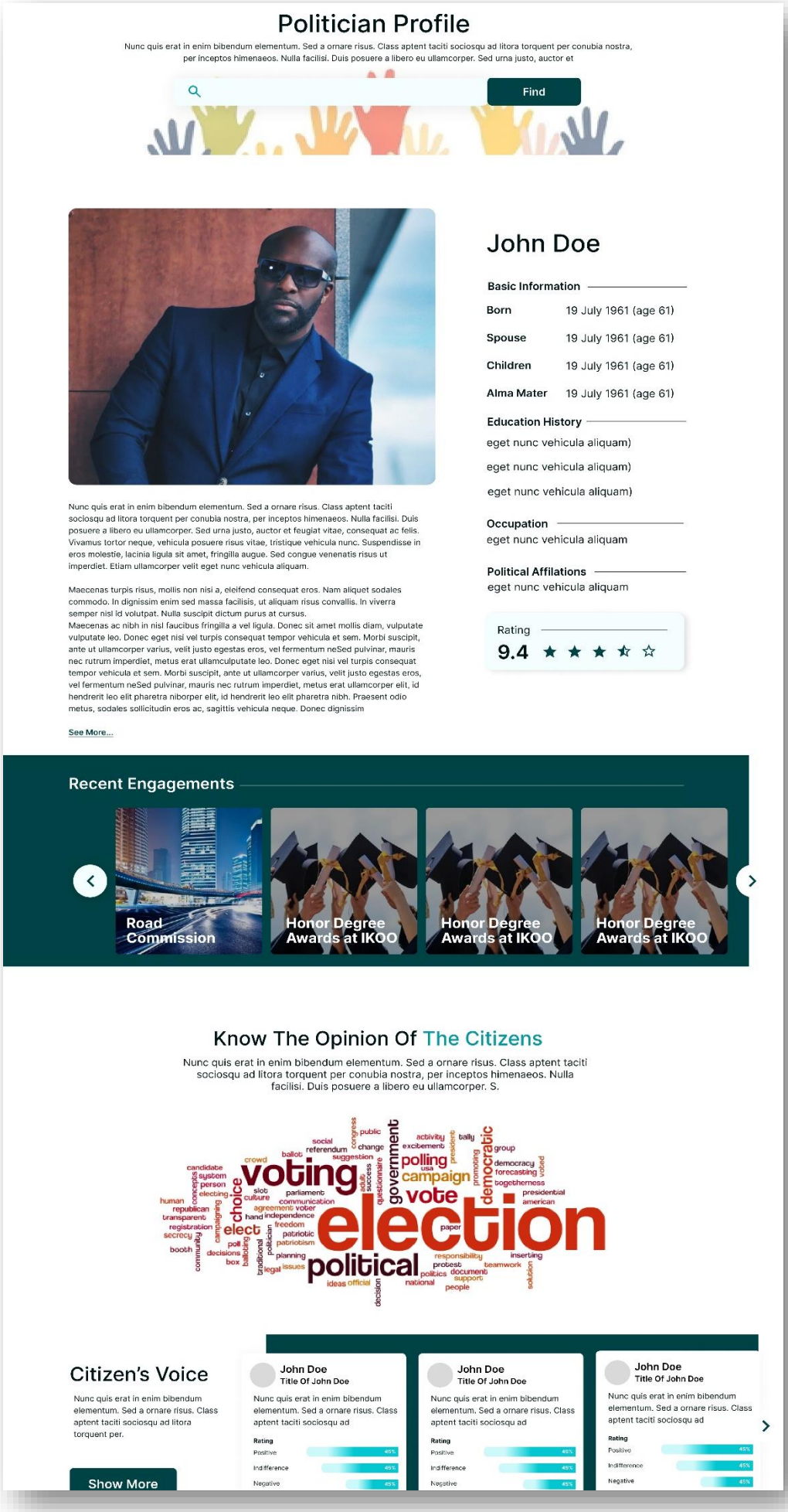
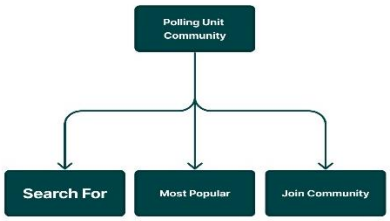



*Figure 7. Cropped version of the Politicians' Profile section UI Design.*

- Polling Unit Community: is the section of this platform aimed at promoting unity and courage amongst citizens during elections periods and in turn combating factors like voter apathy, election rigging at the polling unit level and election violence. The secret is in the numbers of the people. Once citizens show up in large number to partake in this fundamental civic duty, it becomes exponentially more difficult to perpetrate election rigging and violence, and also promotes a culture of civic responsibility.




This section will eliminate the feeling of isolation that comes with these civic responsibilities by connecting citizens together during these times to create a familiar and friendly ambience to carry out these duties. The more people get to know each and move in group, the safer they feel.

*Figure 8. Sitemap of the Polling Unit Community Section.*

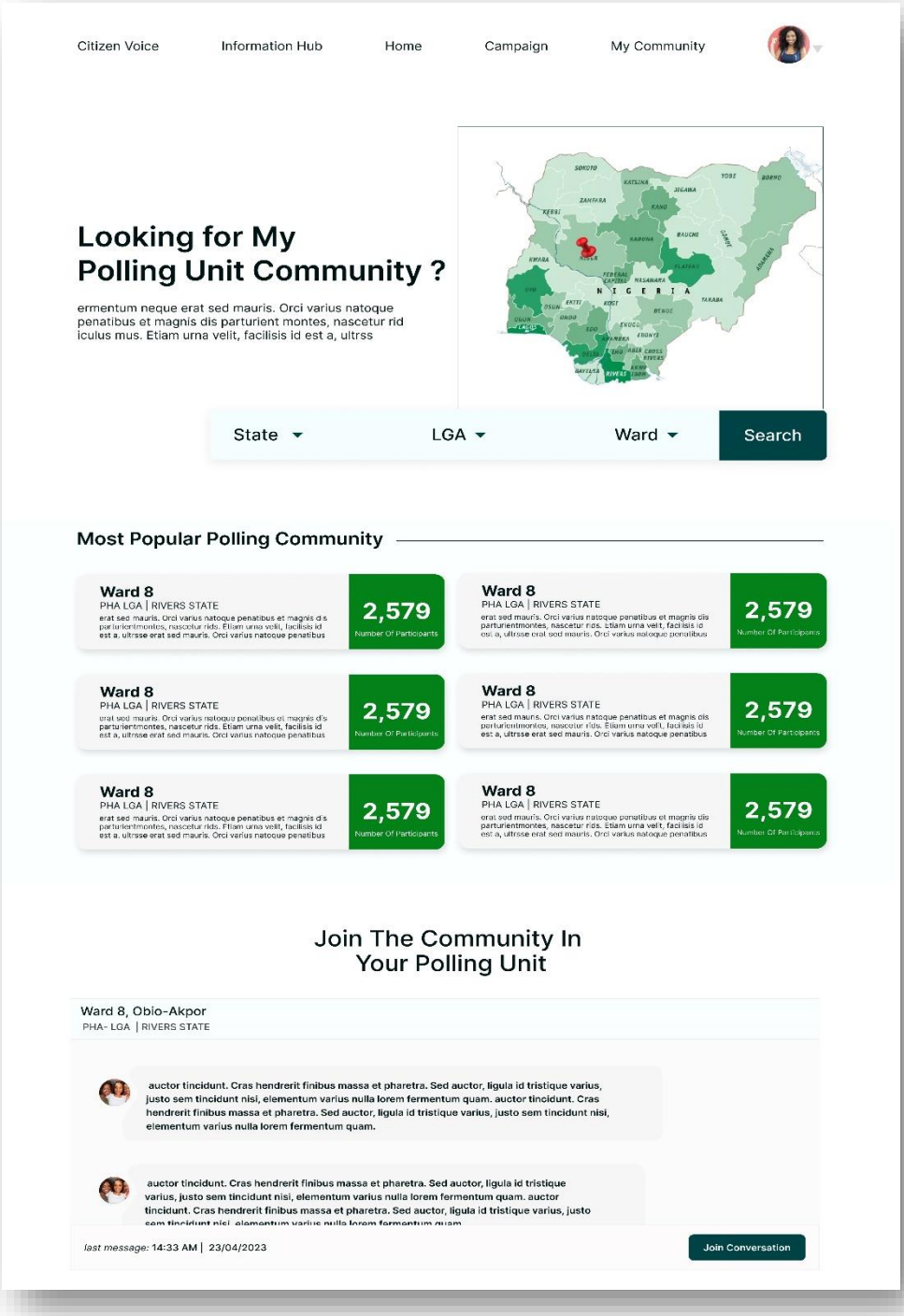



*Figure 9. Cropped version of the Polling Unit Community section UI Design.*

- Election Observer: is intended to aid the citizens during election. It will serve as an avenue for citizens to document their votes in real-time on a centralized platform, so as to make the election process somewhat more resistant to tamper and change. This platform will also have an integrated AI which will automatically extract table from result sheets and perform an automated collation in real-time.

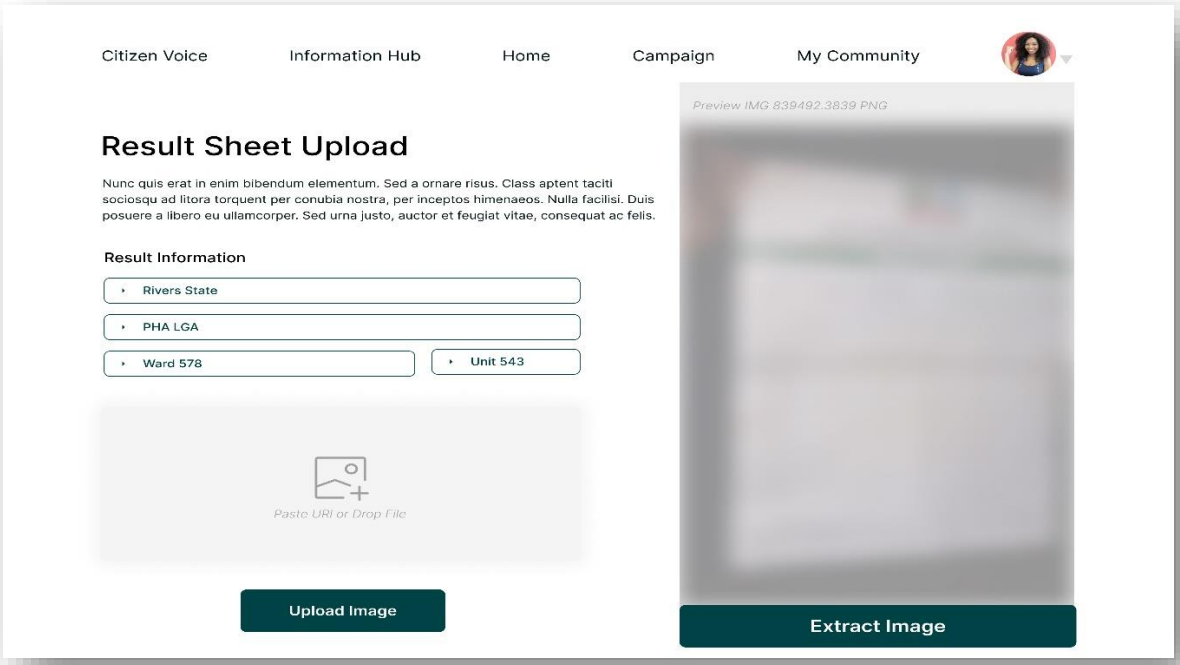



*Figure 10. Election Observer UI Design.*

- Citizens' Voice: is another crucial section of this platform. It will be unfair to paint the picture that there are no goodwill public office holders in our institutions, and this is exactly what this section aims to amplify. This section will summarize the citizens' thoughts and make them easier to understand and grouped by well-meaning public office holders who seek to lead a citizen-driven/centred administration.

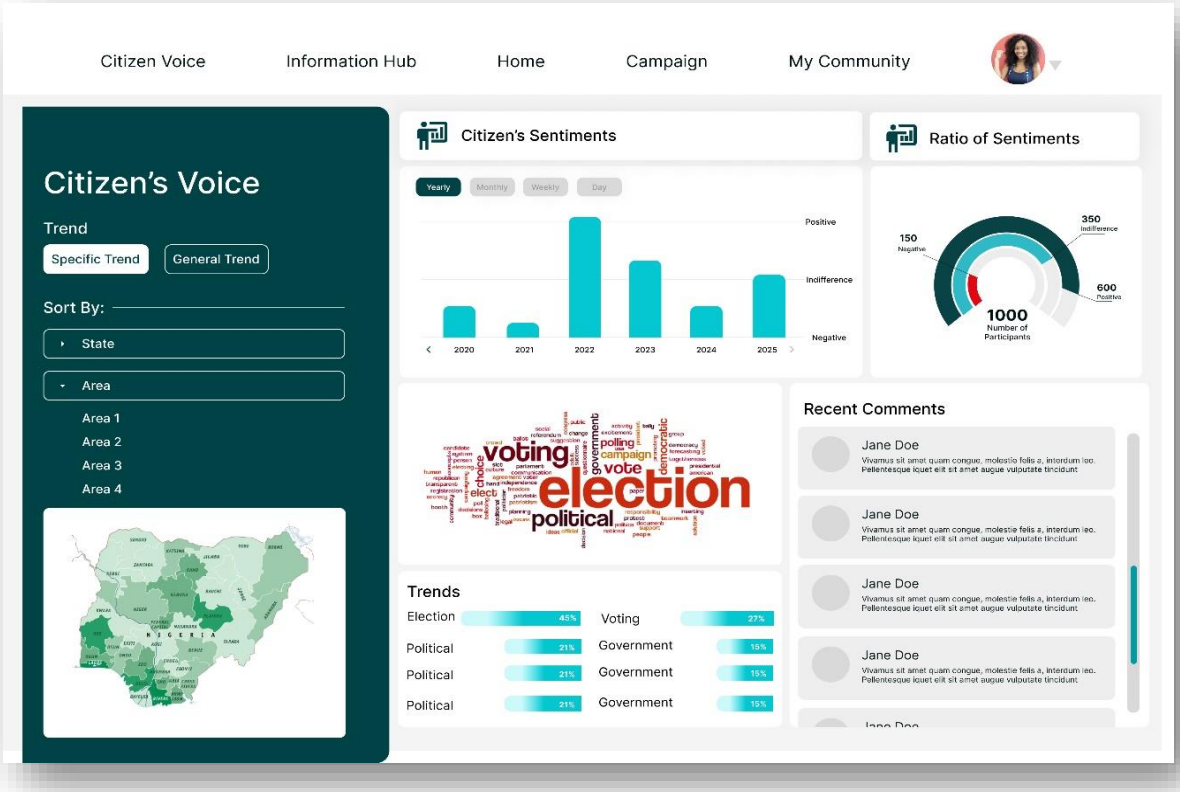



*Figure 11. An Interactive Dashboard which summarizes the Sentiment and Discussions of the Citizens (while keeping them anonymous).*

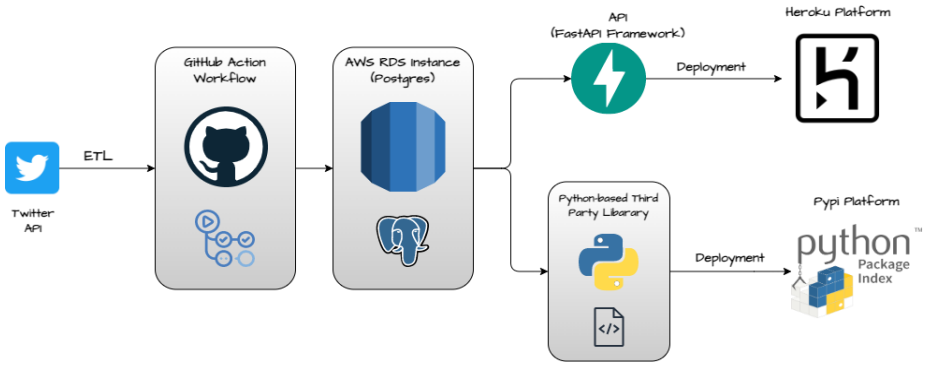
The citizens' voice section will be the entire focus for the proof of concept covered in the next part of this report, where a mini-version of this section will be built using various different tools, platforms and Artificial Intelligence models.

Other sections of this platform (which aren't expressly stated in this report) are also considered for future implementation and scaling, like the Crowd Funding section where citizens will be able to visit the party's profile page where their preferred candidate is registered and donate to his or her campaign. At the end of the implementation of this platform, the hope is eliminate excuses like "I don't know enough about those running for office to cast my vote.", "My preferred candidate cannot afford the expenses of election campaign so I won't be voting", "I do not know how to donate to the campaign of my preferred candidate so I can't get actively involved in the civic process" amongst many others. The people will be given the opportunity to throw their collective weight behind their preferred candidate and efficiently too, which will give the sense of personalizing the democratic process which truly should be working for them. Another crucial section is the Market Place section which will harness the entrepreneurial potential and business oriented nature of Nigerians, as is common amongst citizens of underdeveloped/developing nations. This section will also breed collaboration amongst citizen thereby bolstering the unity that is crucial for a people to act with one mind.

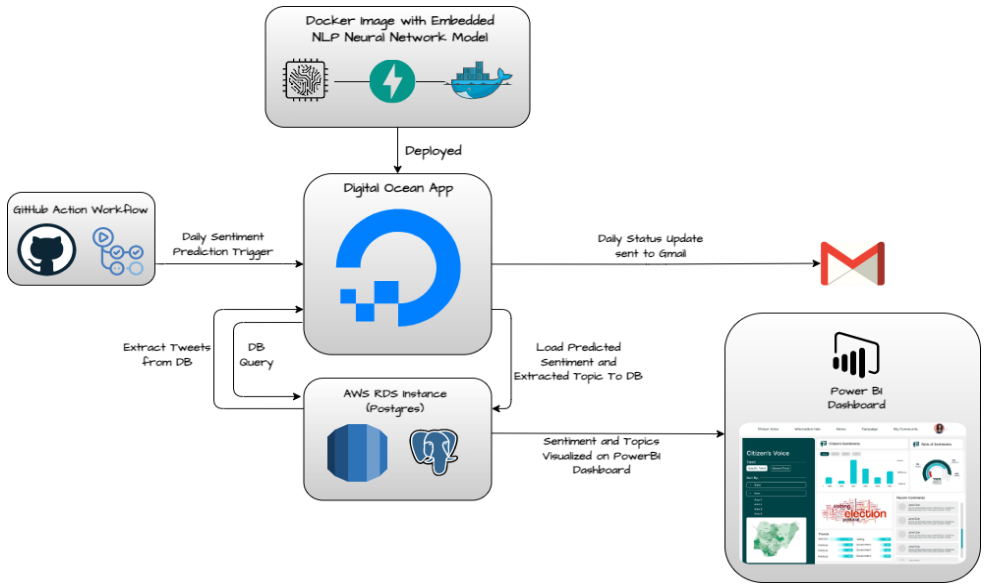
## 2. Proof of Concept

This section of the project focuses on building a miniature version of the Citizens' Voice/Sentiment section of the platform, which could be then scaled up during implementation. In this section, data automatically extracted from citizens (through ETL pipelines) are stored in a remote AWS Postgres RDS instance, which is then later extracted and used to predict the sentiment of the citizens using a custom trained Natural Language Processing (NLP) Deep Neural Network (DNN) model, also used to model trending topic from the citizens' discussions using Gensim LDA method. The predicted sentiments and extracted topics are then stored on the database instance to be later extracted and used to update a published PowerBI Dashboard periodically and automatically for non-technical user (well-meaning public office holders), so as to help them make data-driven decisions. The data in the database is also made available to technical users in a python IDE through a custom built library hosted on Pypi, and in other non-python environments through a rest API built on the FastAPI framework and hosted on Heroku.

This section of the project was carried out in two phases. The first phase covered the data preparation, ETL workflow automation, Database creation, API development and deployment, and Third Party Library (TPL) development and deployment.




While the second phase covered the transformation of the data for training, the training of various NLP model (including Deep Neural Network Models), topic modelling using Gensim LDA, integration of these trained models in an automated workflow, and deploying all necessary application on cloud platforms like Digital Ocean using Docker Containers. Daily email update was also integrated into these deployed applications.



However, it is important to note that the project flow overview and the in-depth description will not be presented in these different phases as explained in the paragraph above but as one single phase encompassing all the milestones covered in this proof of concept.

## Project Flow Overview

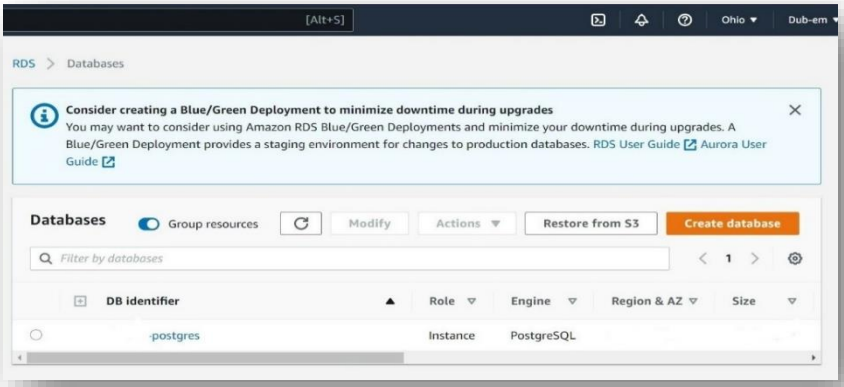
This sub-section covers a high level overview of the project flow.

- AWS RDS Postgres Resource Creation
- Twitter Data Extraction and Loading
- Twitter Data Extraction Process Automation (using GitHub Action)
    - Daily Update through Email
- API Development and Deployment (using Heroku) to expose data to technical end-users in python or other languages
- Python TPL Development and Deployment (on Pypi) to expose data to technical end-users in Python IDE
- Script building for Data Cleaning
- Data Transformation (Using Gensim) and Model Training (using Tensorflow, Pytorch etc.)
- Topic Extraction (using Gensim LDA)
- Implementation of Trained Model and Topic Extraction (into scripts) for periodic and automated prediction and information extraction
- Dockerization of Scripts into images and testing using Docker Containers
- Hosting Docker images on Digital Ocean App, and deploying with Kubernetes
- Developing a daily trigger to initiate prediction and send update through email
    - Using Cron Jobs (through GitHub Action Workflow)
    - Integrating task schedule into the source code using Python Schedule Library.
        - Convert the scripts into Docker Images.
        - Create a Digital Ocean Droplet (Ubuntu based)
        - Connect to Droplet locally through secure shell (SSH)
        - Pull remote Docker image into the droplet.
        - Build and run container using image.
- Connect to AWS RDS instance using PowerBI
    - Build Dashboard to reflect the daily prediction and topics extracted from the Twitter data
    - Dashboard updates automatically
    - Host Dashboard on a URL for non-technical end-users.

## Project Flow In-depth Description

In this section, a more detailed explanation of the project flow will be presented with pictorial references.

- AWS RDS Postgres Resource Creation:




An RDS Postgres instance was created on the Amazon Web Service Cloud Platform to house all the project's data storage needs. This instance was then connected to and managed through pgadmin4

*Figure 12. AWS RDS Postgres Instance*

- Twitter Data Extraction and Loading:

```
@task(max_retries=3, retry_delay=datetime.timedelta(minutes=30))
def get_data():
    api_key = "your api_key"
    api_key_secret = "your api_key_secret"
    access_token = "your access_token "
    access_token_secret = "your access_token_secret"
    auth = tweepy.OAuthHandler(api_key,api_key_secret)
    auth.set_access_token(access_token,access_token_secret)

    api = tweepy.API(auth)

    keywords = ['Buhari OR APC OR  PeterObi OR Tinubu OR PDP OR Atiku OR LabourParty']
    #keywords = ['Buhari','APC', 'PeterObi','Tinubu','Atiku']
    #it seems the api does not return every tweet containing at least one or every keyword, it returns the only tweets that contain
    #solution was to use the OR in the keywords string as this is for tweets search only and might give errors in pure python
    limit = 10000

    tweets = tweepy.Cursor(api.search_tweets, q = keywords,count = 200, tweet_mode = 'extended',geocode='9.0820,8.6753,450mi', unti
```

After the database was ready, data was extracted from Twitter using Twitter API. Specific filters were used to limit the type of data extracted. This data was then cleaned, transformed and loaded unto the database.

*Figure 13. Source code for the Twitter Data Extraction process.*

- Twitter Data Extraction Process Automation (using GitHub Action): The ETL workflow was then automated using cron jobs in GitHub Actions. The daily workflow run was set to 04:00 WAT.

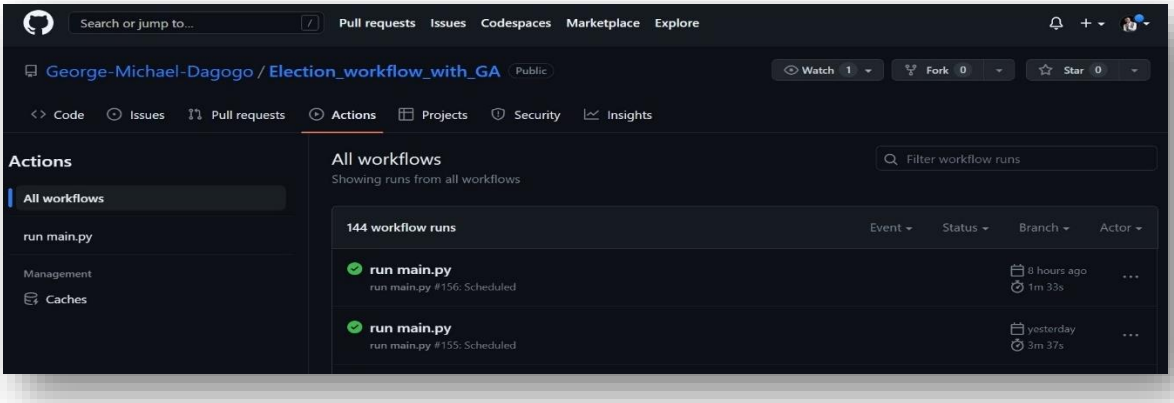
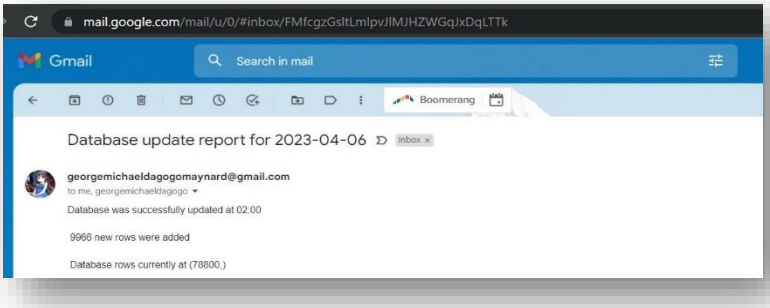



*Figure 14. Daily Workflow using GitHub Actions*

- Daily Update through Email:

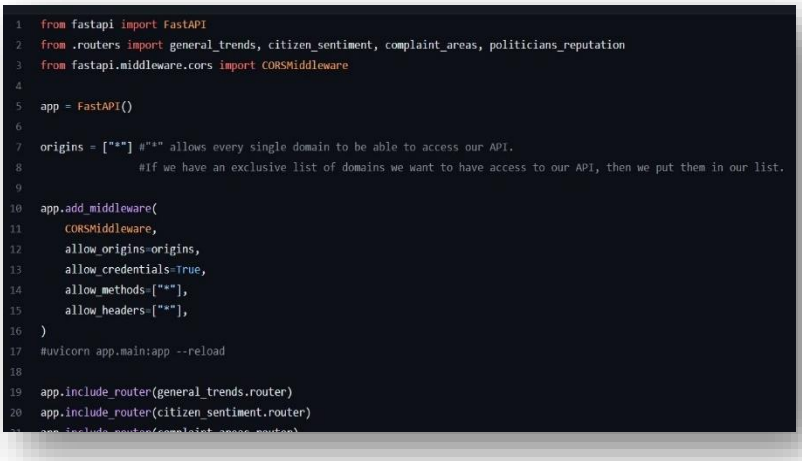



A daily email updated was embedded in the source code using the SMTP python library.

This can also serve as some form of health check.

*Figure 15. Daily email update.*

- API Development and Deployment (using Heroku) to expose data to technical end-users in python or other languages:



```python
from fastapi import FastAPI
from .routers import general_trends, citizen_sentiment, complaint_areas, politicians_reputation
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

origins = ["*"] #"*" allows every single domain to be able to access our API.
                #If we have an exclusive list of domains we want to have access to our API, then we put them in our list.

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
#uvicorn app.main:app --reload

app.include_router(general_trends.router)
app.include_router(citizen_sentiment.router)
```

An API was developed using the FastAPI frame work, and was hosted on Heroku.

The API was intended to give technical users access to the dataset available in the database for other possible future research.

*Figure 16. API Development.*

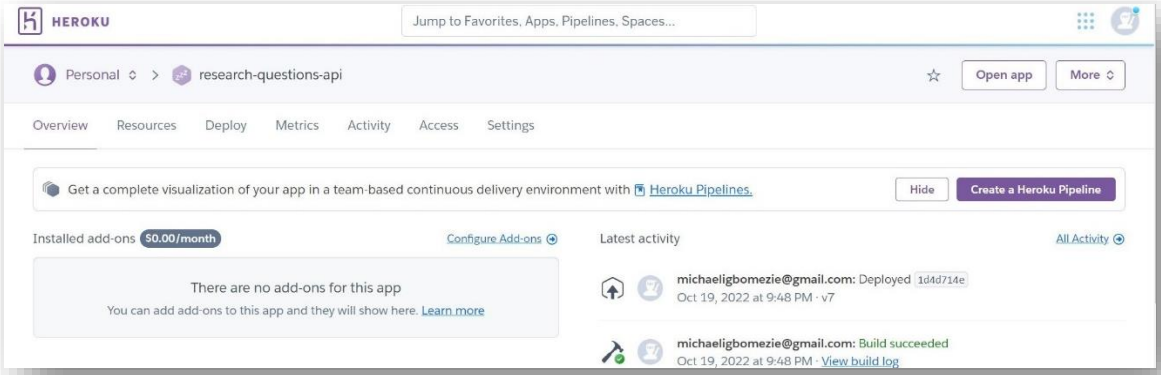



*Figure 17. API Deployment on Heroku Cloud Platform.*

- Python TPL Development and Deployment (on Pypi) to expose data to technical end-users in Python IDE:

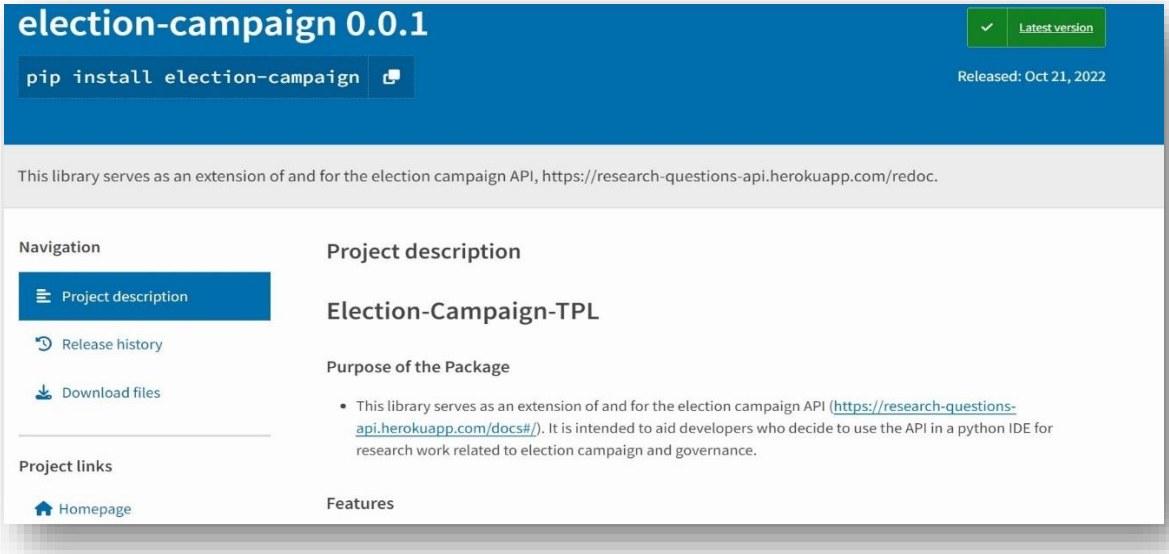
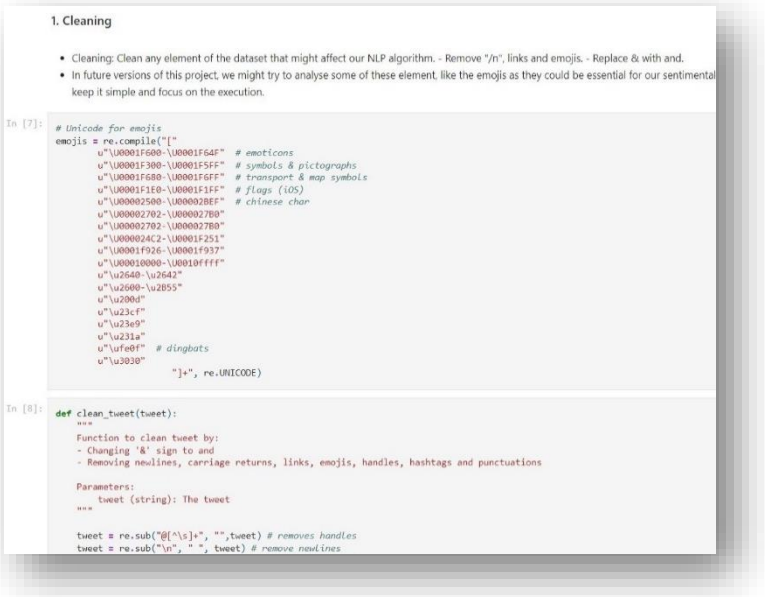



*Figure 18. Custom Python TPL deployed on Pypi.*

- Script building for Data Cleaning:



1. Cleaning

- Cleaning: Clean any element of the dataset that might affect our NLP algorithm. - Remove "/n", links and emojis. - Replace & with and.
- In future versions of this project, we might try to analyse some of these element, like the emojis as they could be essential for our sentimental keep it simple and focus on the execution.

In [7]:
```
# Unicode for emojis
emojis = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\U00002500-\U00002BEF"  # chinese char
        u"\U00002702-\U000027B0"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U00010000-\U0010ffff"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u200d"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\ufe0f"  # dingbats
        u"\u3030"
                    "]+", re.UNICODE)
```

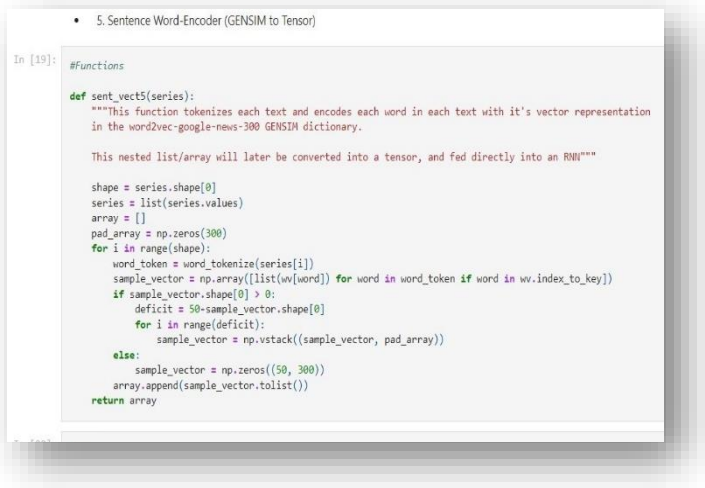
In [8]:
```
def clean_tweet(tweet):
    """
    Function to clean tweet by:
    - Changing '&' sign to and
    - Removing newlines, carriage returns, links, emojis, handles, hashtags and punctuations

    Parameters:
        tweet (string): The tweet
    """

    tweet = re.sub("@[^\s]+", "",tweet) # removes handles
    tweet = re.sub("\n", " ", tweet) # remove newlines
```

The data is cleaned in preparation for model training. All elements of the data considered irrelevant to the intention of a tweet is cleaned from the data using regular expressions.

These elements include but are not limited to numbers, emails, URL links, emoji etc.

*Figure 19. Data Cleaning for Model Training.*

- Data Transformation (Using Gensim) and Model Training (using Tensorflow, Pytorch etc.):



- 5. Sentence Word-Encoder (GENSIM to Tensor)

In [19]:
```
#Functions

def sent_vect5(series):
    """This function tokenizes each text and encodes each word in each text with it's vector representation
    in the word2vec-google-news-300 GENSIM dictionary.

    This nested list/array will later be converted into a tensor, and fed directly into an RNN"""

    shape = series.shape[0]
    series = list(series.values)
    array = []
    pad_array = np.zeros(300)
    for i in range(shape):
        word_token = word_tokenize(series[i])
        sample_vector = np.array([list(wv[word]) for word in word_token if word in wv.index_to_key])
        if sample_vector.shape[0] > 0:
            deficit = 50-sample_vector.shape[0]
            for i in range(deficit):
                sample_vector = np.vstack((sample_vector, pad_array))
        else:
            sample_vector = np.zeros((50, 300))
        array.append(sample_vector.tolist())
    return array
```

After the data has been cleaned, each instance is looped through and tokenized. Each token of each instance is converted to its vector representation using the Gensim Word2Vec-Google-News-300 corpus.

A maximum length of input is chosen, with which all instances longer than this length is truncated, and all instances shorter than this length are padded.

*Figure 20. Data Transformation using Gensim corpus.*

- Sentiment Method 5 (with Dataset 5)

In [34]:
```
#Dataset5 combined with SimpleRNN

training5 = tf.convert_to_tensor(training5, dtype=tf.int64)

label5 = to_categorical(label5, num_classes)
```

In [205…
```
model5 = Sequential([
    SimpleRNN(50, input_shape = (training5.shape[1], training5.shape[2]), return_sequences = False),
    Dense(num_classes, activation='softmax'),
])
```

In [ ]:
```
model5.compile(loss='categorical_crossentropy', optimizer = Adam(learning_rate=0.0001), metrics = ['accuracy'])
hist5 = model5.fit(training5, label5, epochs = epochs, batch_size = 50, validation_split=0.2, verbose=2)
```

In [207…
```
scores_dict['model5'].append(hist5.history["val_accuracy"][epochs - 1])
```

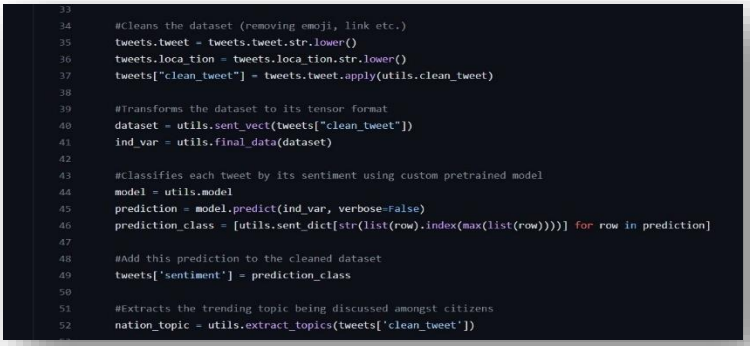
In [243…
```
if os.path.isfile('./models/sentmodel5.h5') is False:
    model5.save('./models/sentmodel5.h5')
```

*Figure 21. One of the Trained and Saved Models (Simple RNN in this instance).*

- Topic Extraction (using Gensim LDA):


*Figure 22. Topic Extraction using Gensim LDA.*

- Implementation of Trained Model and Topic Extraction (into scripts) for periodic and automated prediction and information extraction:



```
#Cleans the dataset (removing emoji, link etc.)
tweets.tweet = tweets.tweet.str.lower()
tweets.loca_tion = tweets.loca_tion.str.lower()
tweets["clean_tweet"] = tweets.tweet.apply(utils.clean_tweet)

#Transforms the dataset to its tensor format
dataset = utils.sent_vect(tweets["clean_tweet"])
ind_var = utils.final_data(dataset)

#Classifies each tweet by its sentiment using custom pretrained model
model = utils.model
prediction = model.predict(ind_var, verbose=False)
prediction_class = [utils.sent_dict[str(list(row).index(max(list(row))))] for row in prediction]

#Add this prediction to the cleaned dataset
tweets['sentiment'] = prediction_class

#Extracts the trending topic being discussed amongst citizens
nation_topic = utils.extract_topics(tweets['clean_tweet'])
```

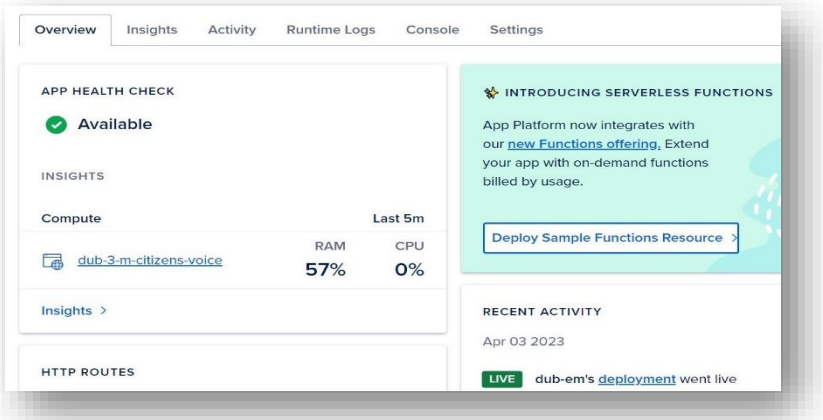
Having trained the model, the entire process is converted into a workflow, which automatically pulls data from the database, predicts on it, extracts topics and loads the result back into the database

*Figure 23. Automation of the Sentiment Prediction and Topic Extraction Process.*

- Dockerization of Scripts into images and testing using Docker Containers:


*Figure 24. Scripts are converted into Docker Images and pushed to Docker Hub.*

- Hosting Docker images as a Digital Ocean App, and deploying with Kubernetes:




The hosted Docker image was connected to a Digital Ocean App and hosted on Kubernetes.

It was hosted as an API so as to pass the health check. This eventually raise some issues with scalability which were later addressed.

*Figure 25. Docker Image hosted as a Digital Ocean App.*

- Developing a daily trigger to initiate prediction and send update through email: Having been hosted as an API, the app needed a daily trigger to initiate the workflow through an endpoint. Two methods were implemented for this, with the latter being a solution to the scalability of the first.
    - Using Cron Jobs (through GitHub Action Workflow):

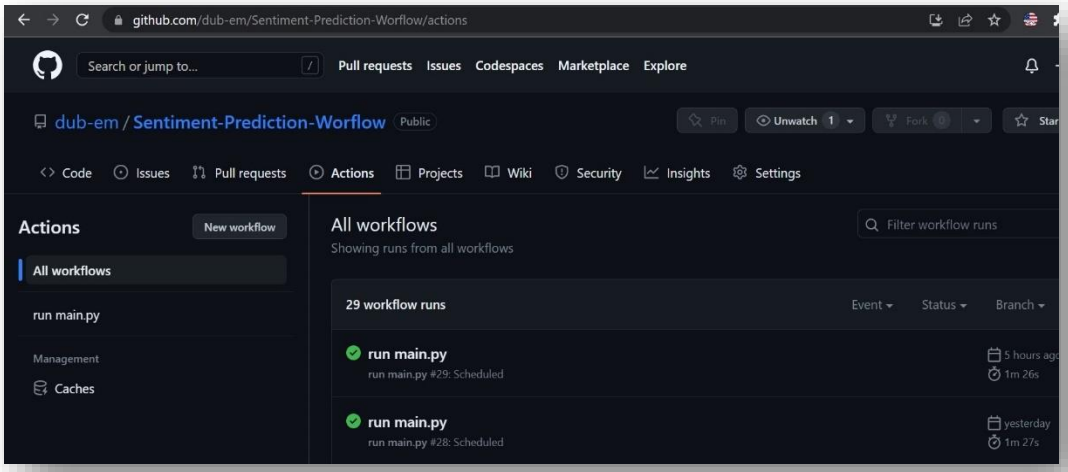



*Figure 26. Workflow automated using GitHub Action to trigger daily prediction.*

This solution worked, but as it turned out it wasn't sufficient when the number of rows predicted on in a day was scaled up. To quickly explain the reason for this, API GET requests have a timeout limit of 100 seconds (600 seconds for enterprises), so if the time complexity of the deployed workflow causes it to exceed the stipulated time limit for a request on a given number of rows, the daily requests will always return a 503 HTTP status code (timeout error). As a result of this scaling obstacle, whichever solution proposed to fix this had to either reduce the time complexity by distributed computation, or isn't even restricted by the 100 seconds limit in the first place.

- Integrating task schedule into the source code using Python Schedule Library: This solution fixed the problem encountered in the previous approach. The workflow was hosted dockerized (as a simple script, not an API) and hosted on an Ubuntu based Digital Ocean droplet. However, in order to periodically check when the container was non-responsive, an email update was integrated into the scheduled workflow.

    - Convert the scripts into Docker Images:

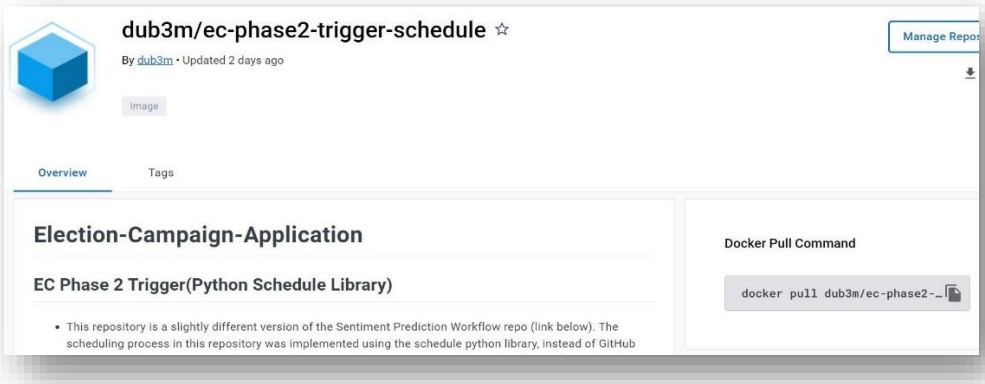



*Figure 27. Workflow is converted to a Docker Image.*

- Create a Digital Ocean Droplet (Ubuntu based):

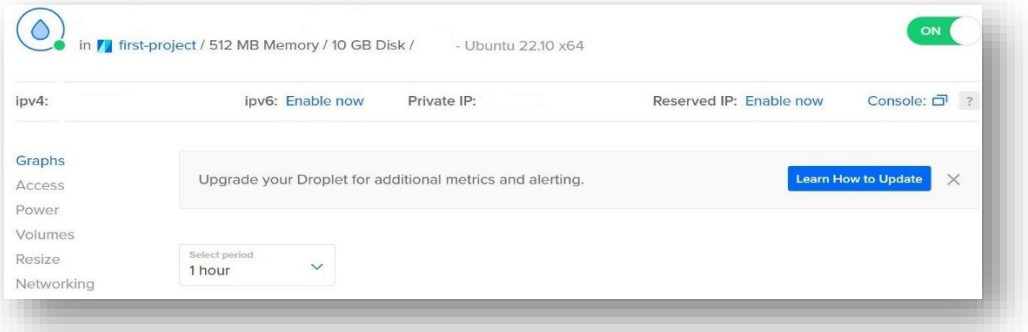



*Figure 28. Ubuntu based Digital Ocean Droplet.*

- Connect to Droplet locally through secure shell (SSH):


*Figure 29. Local VM for connecting to remote Droplet.*

- Pull remote Docker image into the droplet:

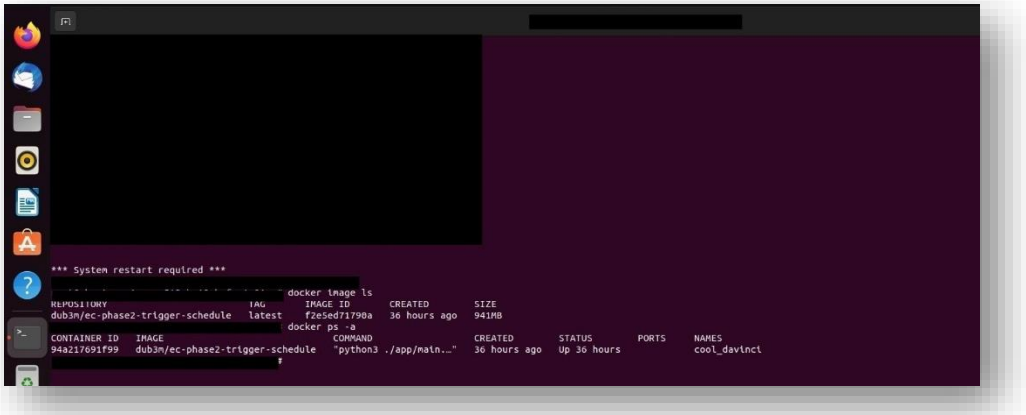



*Figure 30. Connecting to remote Droplet through SSH and running the Docker Image.*

- Build and run container using image:

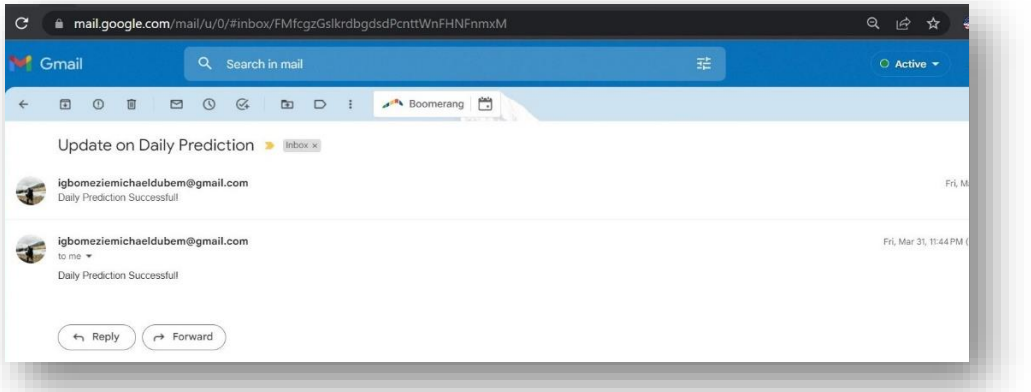



*Figure 31. Docker Image hosted as a Digital Ocean App.*

It's quite obvious that the proof of concept was done across various platforms, using various tools, and in a decentralized manner, which might not be advisable (at best) or applicable (at worst) during production due to the complexity introduced by many different platforms interacting with each other, and how difficult this can make the troubleshooting process (as was experienced during the deployment phase of the project) and even lead to dependency and possible version conflict issues. Although the use of Docker container played a crucial role in simplifying a lot of the handshakes across platforms, this decentralized approach was taken to simply broaden the learning scope and not make the project or acquired knowledge platform dependent. However, it is important to note that during upscaling and production, a single all-encompassing platform will be better for execution.

## Prediction Model Training and Comparison

This subsection of the report glosses over a rather in-depth approach to the NLP related phase of the project. Various models were trained for this purpose, and only the best was selected and used to create the workflow which was deployed on Digital Ocean. The result of the model training phase is as follows;

Table 1. Model Performance

| Data Transformation Method | Validation Accuracy |
|---|---|
| **Dense MLP + Gensim Encoder and PCA** | 0.43 |
| **Dense MLP + Custom Character Encoder** | 0.45 |
| **Multinomial Naive Bayes + Bag of Words** | 0.51 |
| **SimpleRNN + Gensim Encoder and PCA** | 0.54 |
| **SimpleRNN + Keras Encoder** | 0.55 |
| **Dense MLP + Gensim Encoder and Average** | 0.60 |
| **SimpleRNN + Custom Character Encoder** | 0.61 |
| **SimpleRNN + Gensim Encoder and Average** | 0.61 |
| **SimpleRNN + Tensorflow Tensor** | 0.61 |
| **SimpleRNN + Tensorflow Tensor** | 0.61 |
| **LSTM + Tensorflow Tensor** | 0.61 |

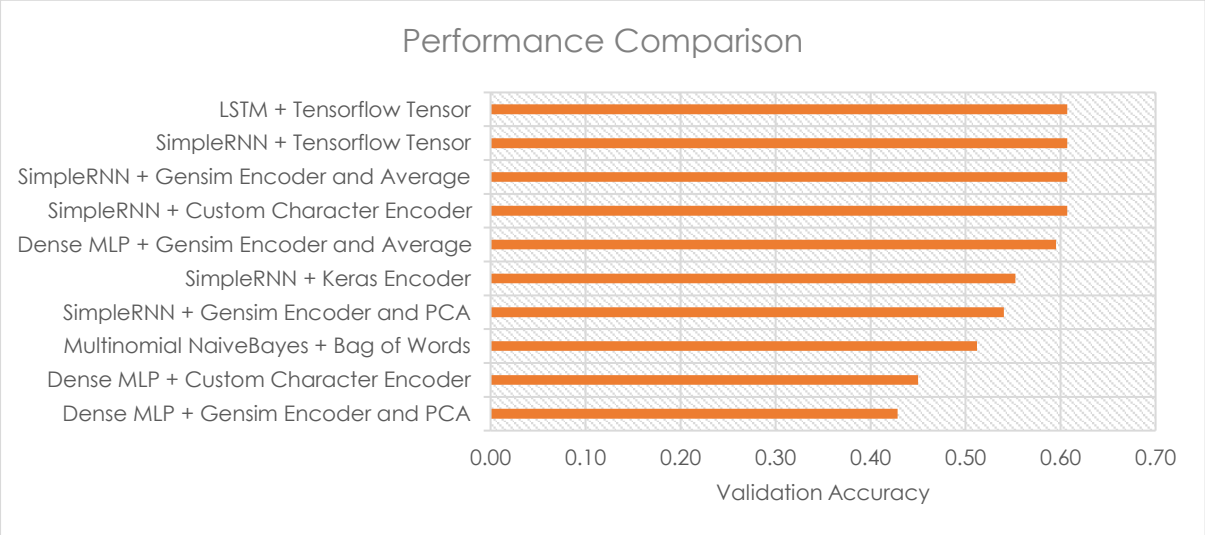



*Figure 32. Model Performance Comparison from the Training Phase.*

For a much in-depth understanding of the Model Development process that gave birth to these results, please refer to the full report in this repository (https://github.com/dub-em/Election-Campaign-Application-Phase2/blob/main/SenitmentAnalysis_Report.pdf) or the earlier posted LinkedIn Article (https://www.linkedin.com/pulse/citizens-sentiment-michael-igbomezie) written on this phase of the project. This phase took a rather deep dive into NLP, Deep Neural Networks and how they could be applied for this requirement in the project.

## Languages, Tools, Software and Platforms used

Due to the numerous tools used, only relevant tools and those easily remembered are stated below.

- Languages: Python, SQL
- Relevant Packages: BERT, Tensorflow, Pytorch, Sklearn, Pandas, Numpy, NLTK, Gensim, FastAPI, Psycopg2, SQLAlchemy, Prefect, Tweepy, Request, Schedule
- Software: VS Code, Jupyter, PGAdmin4, Docker Desktop
- Version Control: Git
- Cloud and Container Services: AWS, Heroku, Digital Ocean, Pypi, GitHub and GitHub Action, Docker Hub
- Visualization: PowerBI
- Operating Systems: Windows, Ubuntu

## 3. Result and Conclusion

The purpose of this project was to demonstrate the proof of concept of a section of the Citizens' Voice Platform. It was an end-to-end solution that covered problem identification, solution proposal, objective formation, project flow outline and milestone formation, conceptual design/planning, detailed planning, implementation (which involved database creation, data gathering, ETL, Model training, workflow development, scheduling and deployment, and development and deployment of supporting tools), visualisation, dashboard creating and publishing for both technical and non-technical user, and conclusion report for proper documentation of the entire process.

The final part of this project involved connecting the database through PowerBI, developing an effective dashboard and publishing it for anyone to refer to (like well-meaning public office holders who care to know how their constituents are reacting to the decisions they are making). This published PowerBI dashboard can be accessed at;

(https://app.powerbi.com/view?r=eyJrIjoiYWMyN2E1YWYtZmQ3NC00YWNlLTg4N2UtOThiZWJiNjQ1MzdjIiwidCI6IjA1Mzg1Yjk4LWE0MTMtNGUzMC04YzhiLTliNDQ5ZTgyMTc3NiJ9)

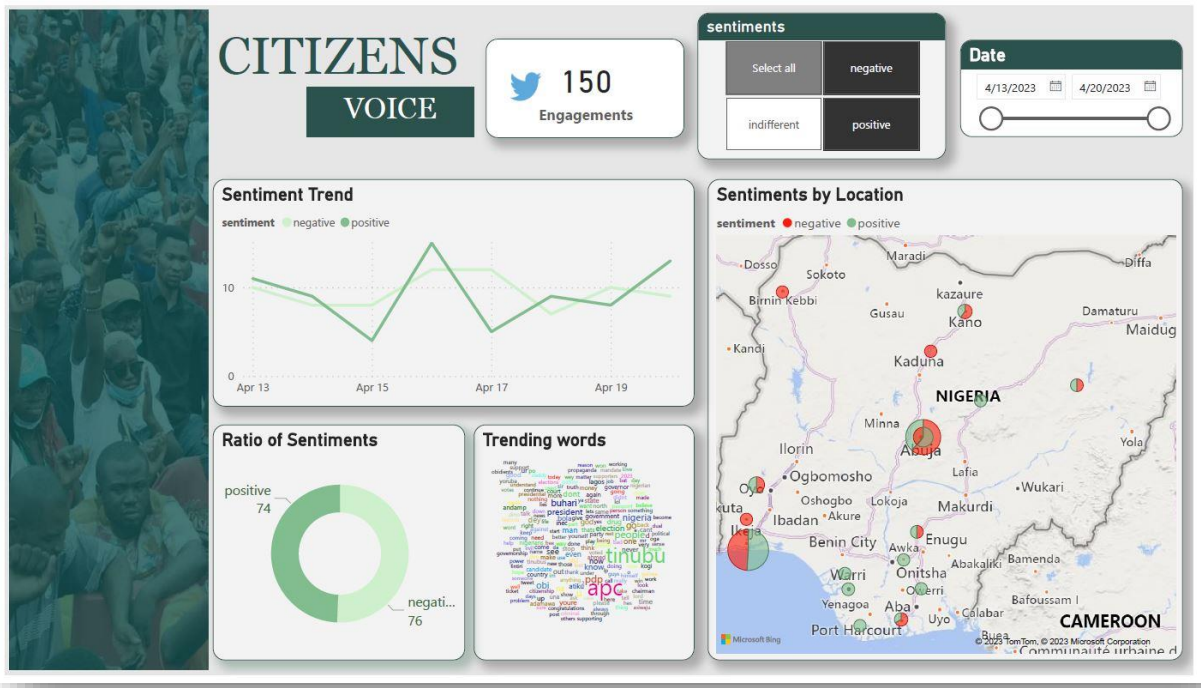



*Figure 33. Published PowerBI Dashboard showing the Citizens' Sentiment and Trending Discussions.*

## Obstacles

There were numerous obstacles experienced during this project, most of which were simply avenues to learn and expand knowledge. However, the obstacle listed below are those that will be addressed before the platform is scaled up to process millions of entries and produce effective predictions.

- Data Quality: There is a popular saying in the data community that the model is only as good as the data. It will be a rookie's mistake to ignore this saying when developing predictive models. This is part of the reason domain knowledge is quite crucial when handling and harnessing data.

The performance of the trained model were largely average largely due to poorly labelled data. When data is poorly labelled, it becomes difficult for any model, no matter how big to find a pattern with which to generalize well.

- Workflow Time Complexity: As earlier stated in this report, the workflow had a polynomial time complexity which certainly didn't work well with the stipulated time limit for API request (even for enterprise users), even for as little as 1000 rows processed per day. Despite having found a solution to the 100 seconds API request limit, the time needed to process 2000 rows through the deployed workflow still took roughly 6 minutes. When this is scaled up to 80,000 rows per day, this process could last over 4 hours to parse and process through the deployed workflow. It is quite obvious that if there is any hope to scale this process up to millions of rows for a country like Nigeria with over 200 million citizens, this current solution will have to be improved upon.

- Cross-platform Integration: Although not as hindering as the previous two stated obstacles, this factor still played a significant role in making forensic analysis of error logs immensely more difficult than expected. Hopping from one platform to another tracing the source of error in code execution isn't an enjoyable process.

## Solution and Future Scaling

To effectively scale the platform to account for a country like Nigeria, it is obvious that these obstacle will need to be addressed.

- Data Quality: The data labelling process for the predictive models will simply need to be conducted by trained professional in the field of human psychology. This will reduce the chances of mislabelling of data instances, which will help the models find a good pattern in the dataset to carry out its task effectively. Additionally, long, complex, multiple and compound sentence will need to be broken up so as to effectively differentiate sentiments to also help with better pattern recognition.

- Workflow Time Complexity: One good approach to this problem is the integration of Big Data. The numerous entries can be converted into RDD partitions using Apache Spark and the computation can be run on a cluster (on platforms like Data Bricks) to significantly reduce the running time of the entire process.

- Cross-platform Integration: To fix this issue, the production and scaling up of the platform simply needs to be done on an all-in-one cloud platform that integrates well with other platforms when needed. A platform such as AWS or any other good platform that fits this criteria can be used for this purpose. Easy integration and troubleshooting could remove many unnecessary complexities in the development or production phase.

## 4. Collaborators

### Project Coordinator (Team Lead)

Michael Dubem Igbomezie

- Email: michaeligbomezie@gmail.com
- LinkedIn Profile: https://www.linkedin.com/in/michael-igbomezie-2901a5122
- GitHub Profile: https://github.com/dub-em

### Project Collaborators (Team Members)

George Michael Dagogo

- Email: georgemichaeldagogo@gmail.com
- LinkedIn Profile: https://www.linkedin.com/in/michael-dagogo/
- GitHub Profile: https://github.com/George-Michael-Dagogo

Abiye Danagogo

- Email: abiyedanagogo@gmail.com
- LinkedIn Profile: https://www.linkedin.com/in/abiye-danagogo/?originalSubdomain=ng
- GitHub Profile: https://github.com/AbiyeDanagogo

Chineye Idemili

- Email: idemilichinenye@gmail.com
- LinkedIn Profile: https://www.linkedin.com/in/chinenye-idemili-204aaa14b/
- GitHub Profile: https://github.com/Chinenye001

Nkwelle Daniel

- Email: nkwellearinze@gmail.com
- LinkedIn Profile: https://www.linkedin.com/in/daniel-nkwelle-823a7a235/
- GitHub Profile: https://github.com/daniel024-cyber

Gbobie Claire

- Email: gbobieclaire@gmail.com
- LinkedIn Profile: https://www.linkedin.com/in/gbobieclaire/

## 5. Project Links

### Phase 1

- LinkedIn Article: https://www.linkedin.com/pulse/citizens-voice-michael-igbomezie/
- GitHub Repository: https://github.com/dub-em/Election-Campaign-Application

### Phase 2

- LinkedIn Article: https://www.linkedin.com/pulse/citizens-sentiment-michael-igbomezie
- GitHub Repository: https://github.com/dub-em/Election-Campaign-Application-Phase2

### Additional Links

#### GitHub Repository

- Phase 2 Implementation: https://github.com/dub-em/Election-Campaign-Application-Phase2-Implementation
- Sentiment Prediction Workflow: https://github.com/dub-em/Sentiment-Prediction-Worflow
- Phase 2 Trigger: https://github.com/dub-em/EC-Phase-2-Trigger

#### Docker Hub Repository

- Phase 2 Implementation: https://hub.docker.com/r/dub3m/citizens-voice
- Phase 2 Trigger: https://hub.docker.com/r/dub3m/ec-phase2-trigger-schedule

#### PowerBI Dashboard

Published Link:
https://app.powerbi.com/view?r=eyJrIjoiYWMyN2E1YWYtZmQ3NC00YWNlLTg4N2UtOThiZWJiNjQ1MzdjIiwidCI6IjA1Mzg1Yjk4LWE0MTMtNGUzMC04YzhiLTliNDQ5ZTgyMTc3NiJ9