

# POWER SYSTEM ALARM APP USING SNS

Brandon Graeber

Monica Long

Patrick Martin

## **FINAL REPORT**

## Table of Contents

Concept of Operations.....	1
Functional System Requirements .....	14
Interface Control Document .....	29
Schedule and Validation Plans .....	37
Subsystem Reports .....	42

# POWER SYSTEM ALARM APP USING SNS

Brandon Graeber

Monica Long

Patrick Martin

## CONCEPT OF OPERATIONS

REVISION – 1.2  
16 February 2023

# CONCEPT OF OPERATIONS FOR

TEAM &lt;29&gt;

APPROVED BY:

---

Project Leader
Date

---

Prof. Jang
Date

---

T/A
Date

## Change Record

Rev	Date	Originator	Approvals	Description
.				
-	2/10/2023	Team		Draft Release
1.1	2/16/2023	Team		Revision 1.1
1.2	4/20/2023	Team		Revision 1.2

## Table of Contents

<b>1. Executive Summary</b>	21
<b>2. Introduction</b>	8
2.1. Background	8
2.2. Overview	8
2.3. Reference Documents	8
<b>3. Operating Concept</b>	9
3.1. Scope	9
3.2. Operational Description and Constraints	9
3.3. System Descriptions	9
3.3.1. Social Media Parser	10
3.3.2. Machine Learning	10
3.3.3. Companion Application	
3.3.4. Companion Website	11
3.4. Modes of Operations	11
3.4.1. Retraining Mode	11
3.4.2. Output Mode	11
3.5. Users	11
3.6. Support	12
<b>4. Scenarios</b>	12
4.1. Fire Near Power Lines	12
4.2. Search and Rescue	12
4.3. Unpredictable Natural Disaster	12
<b>5. Analysis</b>	12
5.1. Summary of Proposed Improvements	12
5.2. Disadvantages and Limitations	12
5.3. Alternatives	13
5.4. Impact	13

## List of Tables

No table of figures entries found.

## List of Figures

Figure 1: Overview of Proposed Solution .....	Page 10
---	---------



## **1. Executive Summary**

With the new level of communication social media has provided, an opportunity arises to more effectively communicate useful information to officials. Millions of users of social media constantly upload their experiences almost instantly. Using Machine Learning, the power system alarm app will be able to detect environmental conditions that may lead to power system losses. This Artificial Intelligence will use parsed Twitter data and will output its determination to a companion application and companion website to be used by power plant officials.

## **2. Introduction**

While Power Distribution Plants are generally concerned with increasing efficiency and power generation, forces beyond human control can evaporate hard work. Violent winds, wildfires, earthquakes and thunderstorms are just a few forces of nature that can destroy power lines and transformers. The objective of this project is to leverage machine learning to give engineers an extra warning system to complement their monitoring system.

### **2.1. Background**

Currently, power plants and distribution companies rely on internal monitoring systems to verify plant and grid operations. These systems rely on relays and sensors at substations, which monitor how electricity is flowing through power lines. This is useful for catching malfunctioning wires, electrical switches, insulators, and other components before they break and cause an outage. These sensors are extremely useful but cannot detect a wildfire that is about to hit the power lines or an earthquake that could reduce the stability of a transmission tower. Our proposed system will be used in conjunction with this system and only add additional information and context to any situation that arises. In the past, twitter users have shown more activity during disasters, and we expect this trend to continue, which is why monitoring social media usage to detect hazards is beneficial.

### **2.2. Overview**

While there is an internal monitoring system for power plants and grids, an external system that monitors environmental concerns would complement it. By gaining a few extra minutes of preparation, this system could save a tremendous amount of work, time, and money. This external monitoring system would use Twitter users tweets to detect any hazards to the power grid. These detections would be filtered and reported as notifications to our application. This system is demonstrated in Figure 1.

### **2.3. Reference Documents**

The following documents will be referenced during the design of the system:

The Twitter Privacy Policy

<https://twitter.com/en/privacy>

The Twitter User Agreement

[https://cdn.cms-twdigitalassets.com/content/dam/legal-twitter/site-assets/privacy-policy-new/Privacy-Policy-Terms-of-Service\\_EN.pdf](https://cdn.cms-twdigitalassets.com/content/dam/legal-twitter/site-assets/privacy-policy-new/Privacy-Policy-Terms-of-Service_EN.pdf)

Attention Is All You Need

<https://arxiv.org/pdf/1706.03762.pdf>

Scikit

[https://scikit-learn.org/stable/tutorial/text\\_analytics/working\\_with\\_text\\_data.html](https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html)

Pandas

[https://pandas.pydata.org/docs/user\\_guide/index.html](https://pandas.pydata.org/docs/user_guide/index.html)

Django

<https://docs.djangoproject.com/en/4.2/intro/tutorial01/>

AWS Elastic Beanstalk [https://docs.aws.amazon.com/elastic-beanstalk/?icmpid=docs\\_homepage\\_compute](https://docs.aws.amazon.com/elastic-beanstalk/?icmpid=docs_homepage_compute)

## **3. Operating Concept**

### **3.1. Scope**

Our system will provide relevant and real information to power companies about threatening events that could hinder their operations. This gives companies more leverage than their current internal systems provide, and can protect the company, as well as the people it provides power to. The power companies would be able to type their corresponding ZIP code to opt-in to receive notifications for that area. The companion application and companion website would provide the type of hazard and the location of the hazard so the power company could take appropriate action. Therefore, our product would be available to all companies associated with the power grid, as it could help protect us from a multitude of different unforeseen events.

### **3.2. Operational Description and Constraints**

Our system can be implemented in the control rooms of power generation and distribution companies to notify them of possible incoming danger to their operations. This will be provided by an android application and a website that will send out a notification to operators in the power grid of the possible danger. The system will monitor using keywords, including fire, tornado, etc, and pull tweets containing those keywords from Twitter which will require the use of a connection to the internet, which is a possible constraint. Other constraints include the inability to provide future information since the tweets will report immediate data.

### **3.3. System Descriptions**

This project can be split into three subsystems: A social media parser, a Machine Learning algorithm and trainer, and a user interface. Due to the scope of the user interface, it will be split up into two parts: a companion app and a companion website. The website implementation will be handled by the engineer responsible for the social media parser. The system and the information flow are demonstrated in Figure 1.

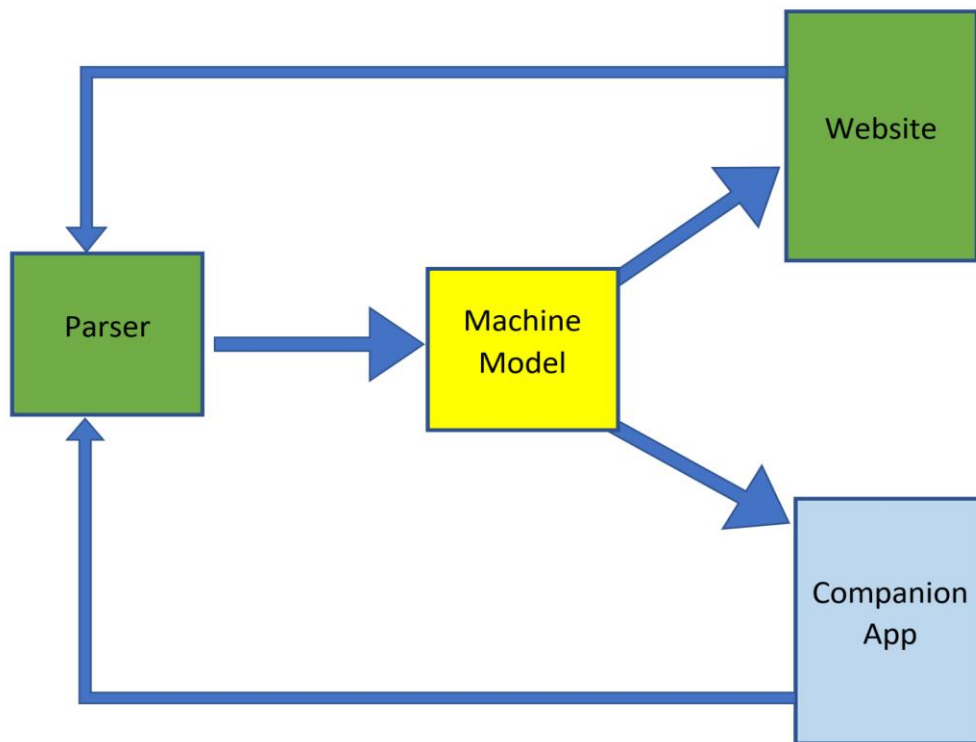


Figure 1: Overview of the Proposed Solution

### **3.3.1. Social Media Parser.**

The Social Media Parser will parse twitter using a list of keywords related to our three scenarios. The list of keywords will be gathered through related lists or manually entered and stored in a file that will be searched for matches by the python file. The parsing will be executed by a python file that will search incoming tweets for the keyword in a specific location, and if there is a hit it will take in the message, user, location, and time of the tweet. This will be collected into a .csv file with columns containing those parameters. This information will then be passed onto the Machine Learning subsection.

### **3.3.2. Machine Learning**

The Machine Learning subsystem will parse the incoming tweets to ensure consistent formatting and categorization. Using this cleaned data, the formatted tweets will be encoded, run through the model, and categorized into three categories: Irrelevant tweets, tweets to be concerned about, and imminent threat tweets. The Machine Learning subsection will utilize python to implement Natural Language Processing.

### **3.3.3. Companion Application**

Using the predictions made by the AI model, a user companion app will effectively and efficiently communicate with power system engineers that may be impacted by the events in question. The application will allow the user to type a ZIP code for the area that they want to

receive notifications for. The user will then receive all current and past alerts, up to one week, that are within a 25-mile radius of the ZIP code. The notification will include the category that the hazard has been sorted into, the keyword used to find the hazard, and the location that was flagged. The notification will also include a details section that will include the time, the user, and the message that was sent out.

### ***3.3.4. Companion Website***

Also using the predictions made by the AI model, a companion website will efficiently communicate with power system operators and engineers about events that could impact their area. The website will take in the user's ZIP code, just like the companion app, and only relay information in that specific area. The notification will be on the website Notifications page and will display the relevant information to the operator's computer screen. The notification will include the location of the event, the time of the tweet, the severity of the event, and the keyword flagged. The notification will also provide the other relevant information in the tables below the notification home.

## ***3.4. Modes of Operations***

There are two main modes of operation for our solution: a Retraining Mode and an Output Mode.

### ***3.4.1. Retraining Mode***

Language and the slang used in the world of social media changes all the time. Therefore, a retraining mode will be implemented in the design. During a specified time, the Machine Learning model will be retrained on the data. To assist in this, operators will be asked during normal operation to identify if the predicted category of the tweet was accurate. This data will be used to train on the latest data. During this retraining, the companion application will give users a warning stating that the model will be unavailable until the training is completed.

### ***3.4.2. Output Mode***

During normal operations, social media, in this case twitter, will be parsed for relevant tweets. If a tweet is deemed either an immediate emergency or a cause for concern, the tweet will be sent to the relevant power grid operators.

## ***3.5. Users***

Power system operators are the target user for the system, as they can provide services and take down certain lines before they become a larger issue. The level of training is near minimal, as the only requirement on the consumer end is having an android device that can download the app and enable notifications. Our system will benefit all levels of the consumer given that the operator acts upon the notification, since it will protect power grid operations, protect equipment used by power companies, and protect the household consumer by keeping their power on.

### **3.6. Support**

Instruction will be given when the app is first downloaded, and when the website is opened to enter a ZIP code for the area the user would like to receive alerts for. The application and website will also have a help section that contains contact information for all troubleshooting purposes.

## **4. Scenarios**

### **4.1. Fire Near Power Lines**

If enough people tweet about fire within a 25-mile radius in a 10 minute time frame, a notification with fire hazard and location will be sent out to the corresponding power grid companies with attached tweets. The tweets should be useful to provide more context and allow the correct response.

### **4.2. Search and Rescue**

While this system is being primarily used to alert power grid operators, it may also be used as an early warning system for search and rescue. If a user tweets out that they are going to be camping in an area known to be risky, a warning can be sent to the relevant authorities so they may monitor the situation at their discretion.

### **4.3. Unpredictable Natural Disaster**

In an unpredictable disaster scenario, there may be a need to send information to multiple different authority bodies. If tweets are coming from areas from which natural disasters are more likely to occur, the number of tweets to activate the alert system will be lower.

## **5. Analysis**

### **5.1. Summary of Proposed Improvements**

- Provides location of potential hazard to allow for action to be taken quickly and precisely
- Sends alert to users with a relevant entered ZIP code based on location of hazard

### **5.2. Disadvantages and Limitations**

The Power System Alarm App will have some limitations which include:

- Only immediate hazards, no future hazard planning
- Not all immediate hazards are uploaded on twitter
- Our AI will analyze text, not images or videos
- The alarm is only effective if someone notices or hears it, it cannot do anything to help the power grid without human intervention
- Limited to the United States
- False negatives and false positives are possible

### **5.3. Alternatives**

While our system is a substantial improvement on the infrastructure, it is not a necessity. The following actions are valid alternatives to our solution.

- Waiting for police to report hazard to power grid
- Letting damages incur without action and assessing afterwards
- Having a team of professionals monitor power lines and report hazards

### **5.4. Impact**

While our system is mostly benign, it will impact people in the following ways:

- Monitoring social media usage and taking location could breach Twitter privacy policies
- Providing immediate updates on the state of the grid to the necessary people could reduce costs and endangerment
- Improve grid operation with an extra layer of protection for equipment and the people it provides

# POWER SYSTEM ALARM APP USING SNS

Brandon Graeber

Monica Long

Patrick Martin

## **FUNCTIONAL SYSTEM REQUIREMENTS**

REVISION – 1.1  
20 April 2023



# FUNCTIONAL SYSTEM REQUIREMENTS FOR Power System Alarm App Using SNS

PREPARED BY:

---

Author Date

APPROVED BY:

---

Project Leader Date

---

Prof. Jang Date

---

T/A Date

## Change Record

Rev	Date	Originator	Approvals	Description
1.0	2/19/2023	Team		Draft Release
1.1	4/20/2023	Team		Revision 1.1

## Table of Contents

<b>1. Introduction</b>	20
1.1. Purpose and Scope	20
1.2. Responsibility and Change Authority	21
<b>2. Applicable and Reference Documents</b>	21
2.1. Applicable Documents	21
2.2. Reference Documents	22
2.3. Order of Precedence	22
<b>3. Requirements</b>	22
3.1. System Definition	22
3.1.1. Twitter Parser	22
3.1.2. Machine Learning Algorithm	23
3.1.3. Companion Application	24
3.1.4. Companion Website	24
3.2. Characteristics	25
3.2.1. Functional / Performance Requirements	25
3.2.1.1. Machine Learning Recall	25
3.2.1.2. Overall Operational Speed	25
3.2.1.3. Twitter Parser Update Time	25
3.2.1.4. Functionality of User Application	
3.2.1.5. Functionality of Companion Website	25
3.2.2. Software Characteristics	26
3.2.2.1. Inputs	26
3.2.2.2. Outputs	26
<b>4. Support Requirements</b>	26
<b>Appendix A: Acronyms and Abbreviations</b>	27
<b>Appendix B: Definition of Terms</b>	28

## **List of Tables**

Table 1. Applicable Documents.....	19
Table 2. Reference Documents.....	20

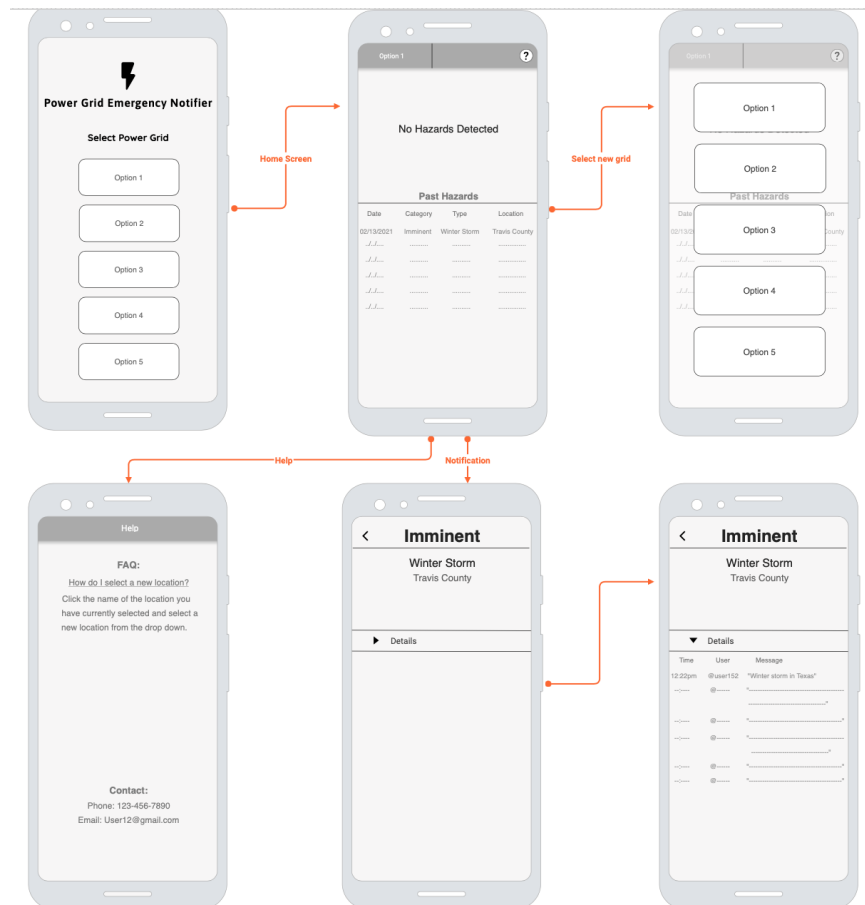
## **List of Figures**

Figure 1. User Interface of the Proposed System.....	20
Figure 2. Block Diagram of System.....	21
Figure 3. Flow Chart of the Twitter Parser Subsystem.....	23
Figure 4. Flow Chart of the Machine Learning Model.....	24
Figure 5. Flow Chart of the Companion Application.....	24
Figure 6. Flow Chart of the Companion Website.....	25

# 1. Introduction

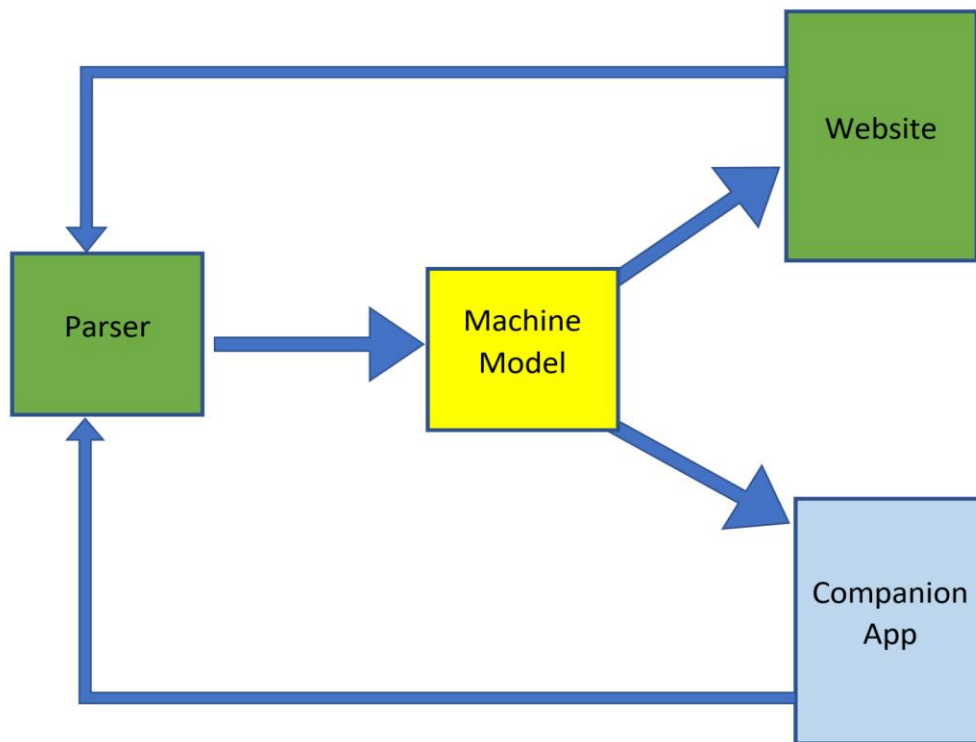
## 1.1. Purpose and Scope

This specification defines the technical requirements for the development of the Power System Alarm App. Figure 1 illustrates the user interface of the proposed system. All verification and validation requirements are outlined in the Verification and Validation Plan.



**Figure 1. User Interface of the Proposed System**

Figure 2 illustrates the plan for solving the problem of the proposed system, and how each subsystem interacts.



**Figure 2. Subsystems and Interaction of Proposed System**

### ***1.2. Responsibility and Change Authority***

The authority to request changes to the project lies with the subsystem designer of record. For the tweet apprehension system and the companion website Patrick Martin has the authority to make changes. For the Machine Learning System, Brandon Graeber has the authority to make changes, and for the companion application, Monica Long has the authority to make changes. For changes to the overall system, whether it be scope or function lies in a consensus vote among the three designers.

## **2. Applicable and Reference Documents**

### ***2.1. Applicable Documents***

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

*Table 1. Applicable Documents*

Document Number	Revision/Release Date	Document Title
1	1997	Standard for Wireless Local Area Networks
2	2011	Standard for Low-Rate Personal Area Networks
3	3.10	Python Standard Library
4	1.1.3	Scikit
5	1.5.2	Pandas
6	2022	AWS Elastic Beanstalk

## **2.2. Reference Documents**

For a list of the documents referenced in our design, please refer to the the Concept of Operations document, Section 2.3

## **2.3. Order of Precedence**

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable report are considered to be for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

# **3. Requirements**

This section defines the minimum requirements that the development item(s) must meet. The requirements and constraints that apply to performance, design, interoperability, reliability, etc., of the system, are covered.

## **3.1. System Definition**

The following sections are meant to be functional definitions of each subsystem.

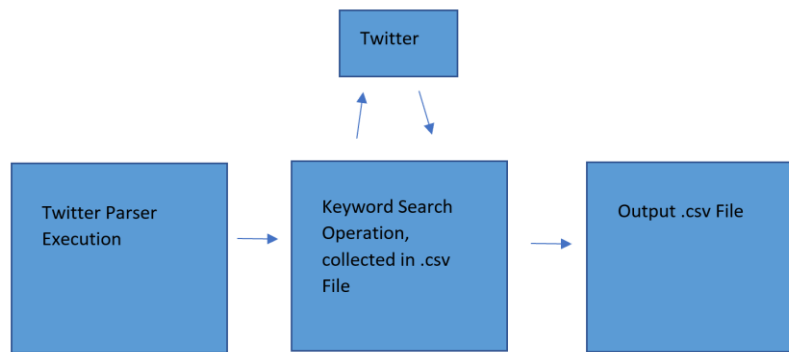
### **3.1.1. Twitter Parser**

The Twitter Parser will parse twitter using a list of keywords related to our three scenarios. The list of keywords will be gathered through related lists or manually entered and stored locally in order to look for matches by the accompanying python code.

The Python code will continually search through the tweets in the accompanying area, and will request data from twitter at a polling rate under the maximum API rate limit, which is 100,000 requests per day.

The tweets and data will be pulled into a .csv file, that will be the parameters of message, time, location, and user. This .csv file will then be sent over to our AI/Machine learning model for processing.





**Figure 3: Flow Chart of the Twitter Parser Subsystem**

### 3.1.2. Machine Learning Algorithm

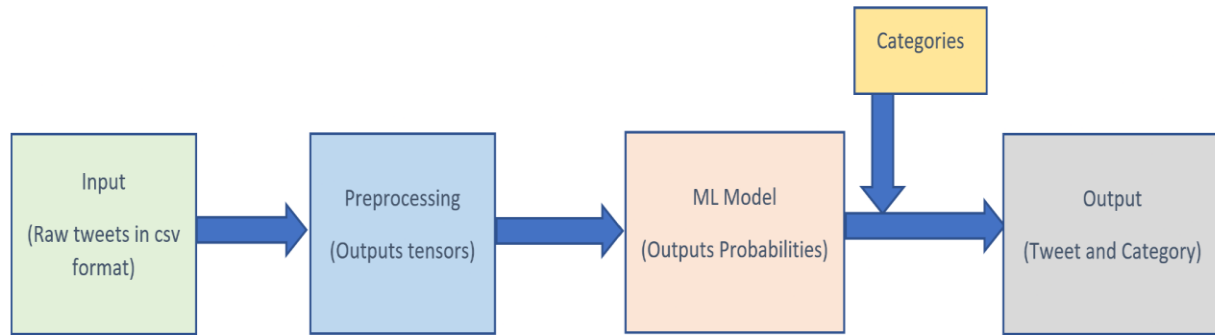
The Machine Learning Algorithm will take in the parsed twitter tweets and outputs the tweet into its respective category to the companion application, if the tweet is of sufficient category.

The ML subsystem will take in the parsed twitter tweets in a csv format. It will pull out the tweet and save a copy to be transmitted if needed. The original tweet will then be run into an encoder to convert the strings of text into numbers. The method of conversion will be a word for word encoder: every word (as long as it is in a pre-defined dictionary) will be encoded separately. These numbers will be saved into a tensor to be used in the model.

Next, the new tensor will be inputted into the machine learning model. If the model is in training mode, the model will be put into training and evaluation mode to update all the weights and biases associated with the model. During this mode, the accepted “categories” of the tweets will be available to use in the model.

If the model is operating in normal mode, the model will output a python tensor list of probabilities. These probabilities correspond to the individual categories of the system. The code will then take the greatest probability of an actual threat and output its respective category.

Finally, during normal operation, if the calculated category is at least of a concerning category, it will be passed along to the companion application in a csv format.

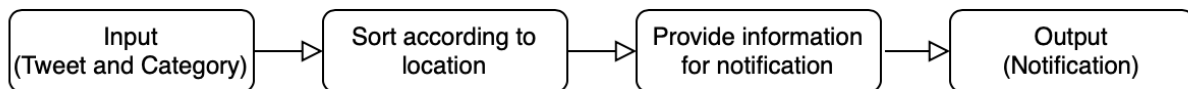


**Figure 4 : Flow Chart of the Machine Learning Model**

### 3.1.3. Companion Application

The Companion Application will take user input for the ZIP code the user is interested in seeing hazards for, the latitude and longitude of the ZIP code will be sent over to the Twitter Parser. The application will then take in the csv output from the Machine Learning Algorithm and identify the location of the hazard to determine the user entered ZIP codes that will be notified. It will then identify the category and the keyword of the hazard in order to form the content of the notification. Finally, it will sort for information that will be included in the 'Details' section, this is the time, user, and message from the tweets for more information.

Once the notification is formed it will be sent out to the application users with the appropriate ZIP code entered. This information will also be stored in the application for easy access after the hazard has passed.

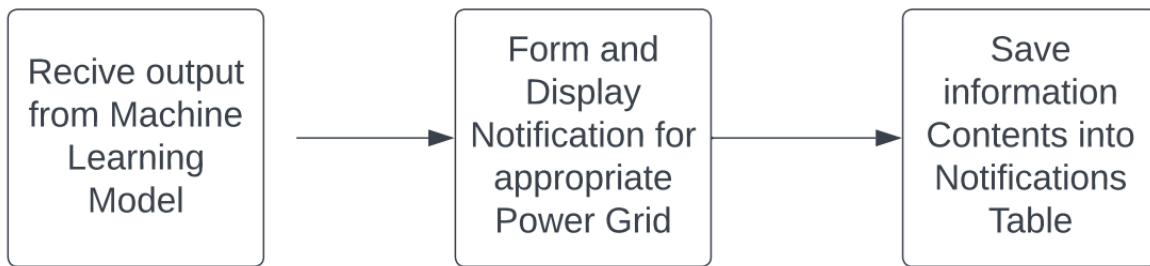


**Figure 5 : Flow Chart of the Companion Application**

### 3.1.4. Companion Website

The Companion Website will take in the .csv output from the Machine Learning Algorithm and identify the location of the hazard to determine the power grid that will be notified. It will then identify the parameters of choice, specifically the location, time, and threat level in order to form the content of the notification. Finally, it will sort for information that will be included in the accompanying table below the notification, specifically time, user, keyword, message, and threat level for easy access.

Once the notification is formed it will be displayed on the screen with the information mentioned. This information will also be stored in the table below the notification area for easy access after the hazard has passed.



**Figure 6 : Flow Chart of the Companion Website**

## **3.2. Characteristics**

### **3.2.1. Functional / Performance Requirements**

#### **3.2.1.1. Machine Learning Accuracy**

Our Machine Learning Algorithm shall meet a recall requirement of 0.85 (85%).

*Rationale: This is the core system performance requirement. A Machine Learning algorithm should be as accurate as possible when sending positive results.*

#### **3.2.1.2. Overall Operational Speed**

The entire system shall take no longer than 1 minute to complete. This time is to be measured from when an emergency tweet was tweeted to when the app receives and pushes a notification. The system will also update every 1 minute.

*Rationale: A System designed to save minutes should not be wasting the saved time. The system should be able to operate a real-world speeds*

#### **3.2.1.3. Twitter Parser Update Time**

The twitter parser shall refresh every minute. It shall repeat this process as long as needed.

*Rationale: In order to avoid overloading the ML model and have the timing correspond, we need to limit the refresh time. This refresh time will also stay below the maximum API rate limit of 100,000 requests per day.*

#### **3.2.1.4. Functionality of User Application**

The User Application shall have the following features:

- ZIP code input
- Help menu
- Past Hazards
- Current Hazard/Notification
- Details of Notification: Tweets to provide context

*Rationale: User Application should be user friendly, easy to understand, and easy to operate.*

#### **3.2.1.5. Functionality of Companion Website**

The Companion Website shall have the following features:

- Grid selection
- Help menu
- Current Hazard Notification
- Previous Hazard Table
- Details of Notification: Tweets to provide context

*Rationale: Companion Website should be user friendly, easy to understand, and easy to operate.*

#### **3.2.2. Software Characteristics**

##### **3.2.2.1. Inputs**

- a. The companion application and companion website will take the input from the user and send this input to the Twitter parser. The inputs will include the data pulled from the scraper, which will be a .csv file that is the input for the ML Algorithm. From the ML algorithm, it will deliver another input of a .csv file to the Companion Application and Companion Website, which will be used for notification delivery.
- b. No input data shall overload the next subsystem, which will be ensured by the timing.

*Rationale: The inputs will create a smooth flow of information.*

##### **3.2.2.2. Outputs**

The output will come from two applications, specifically the Companion Application and Companion Website. The Companion Application will form a notification, which will include the category of the hazard, the name of the hazard, and the location. The notification will also include a details section that will include the time, user, and message of the flagged tweets to provide context. The Companion Website will form the notification using the location, time, keyword, and threat level of the output. The table below will then contain all other important information, including the tweet.

*Rationale: The output will provide a timely notification so the user can act.*

## **4. Support Requirements**

The system requires an android phone with notifications turned on, to allow the application to send alerts. The application will be available on the Google Play Store for download and will have a help page with FAQs and contact information for any feedback or concerns.

## **Appendix A: Acronyms and Abbreviations**

ML	Machine Learning
AI	Artificial Intelligence
NLP	Natural Language Processing
RNN	Recurrent Neural Network
API	Application Programming Interface
MHz	Megahertz (1,000,000 Hz)
TBD	To Be Determined
FAQs	Frequently asked questions

## Appendix B: Definition of Terms

**Application Programming Interface** - a set of functions, programs and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.

**Machine Learning** - a subset of artificial intelligence that leverages different algorithms to help the machine learn patterns in each dataset.

# POWER SYSTEM ALARM APP USING SNS

Brandon Graeber

Monica Long

Patrick Martin

## INTERFACE CONTROL DOCUMENT

REVISION – 1.1  
29 April 2023

# INTERFACE CONTROL DOCUMENT

## FOR

### Power System Alarm App Using SNS

PREPARED BY:

---

Author

---

Date

APPROVED BY:

---

Project Leader

---

Date

---

Prof. Jang

---

Date

---

T/A

---

Date



## Change Record

Rev	Date	Originator	Approvals	Description
.				
1.0	2/20/2023	Team		Draft Release
1.1	4/29/2023	Team		Revision 1.1

## Table of Contents

<b>1. Overview</b>	34
<b>2. Definitions</b>	34
<b>3. Physical Interface</b>	34
<b>4. Electronic Interfaces</b>	34
4.1. Twitter Parser	34
4.2. Machine Learning Algorithm	35
4.3. Companion Application	35
4.4. Companion Website	36
<b>5. Communications / Device Interface Protocols</b>	36
5.1. Wireless Communications	36

## List of Tables

No table of figures entries found.

## List of Figures

Figure 1. Companion Application

34

### 1. Overview

This document describes the interfaces between the subsystems of the project. Because this project is linear in nature, there will be two main interfaces: one for the interface between the parser and the machine learning model, and the other for the interface between the machine learning model and the user application.

### 2. Definitions

During the development of this solution, the following definitions shall be used.

ML	Machine Learning
AI	Artificial Intelligence
NLP	Natural Language Processing
Transformer	Machine Learning Model for NLP
RNN	Recurrent Neural Network
Python	A programming language associated with AI and ML
API	Application Programming Interface
Android Studio	App Development Environment for Android Devices
Java	A programming language associated with App Development
TBD	To Be Determined

### 3. Physical Interface

Because this solution is software based, there is no physical hardware included, with the exception for an android phone to download and use the application.

### 4. Electronic Interfaces

#### 4.1. Twitter Parser

The Twitter parser will take the information from the companion application and the companion website in the following format:

{Latitude; longitude}

The Twitter parser will pass information to the Machine Learning Model in the following format:

{ user ; message ; location ; time sent }

This data is collected from searching through twitter in the specific location radius specified in the parser code. All information will be taken in from tweets that have key words or phrases

from the database, but only the five above categories will be sent as the input for the Machine learning algorithm.

## 4.2. Machine Learning Algorithm

The ML algorithm will take the information from the Twitter Parser and predict which category it belongs to. If the tweet belongs to either the warning or the emergency categories, it will pass on information to the companion application. The ML will also log the incoming tweets and predicted categories for retraining purposes. The logged information will be in the following format:

{ user ; message ; category predicted }

During retraining, operators will be given the log and verify the categories the ML algorithm predicted. A training set will be derived from the log with the following format:

{ user ; message ; correct category }

The following information will be sent to the companion application and website in the following format:

{ Category ; user; message ; location ; time sent ; keyword }

## 4.3. Companion Application

The companion application will output the following to the Twitter Parser:

{latitude; longitude}

The companion application will take the information from the ML algorithm in the format:

{ Category ; user; message ; location ; time sent ; keyword }

The location will first be used to sort the hazard into the appropriate user entered ZIP code, which will establish the users receiving the notification. The category, location, and keyword will also be used in the main heading of the notification. Followed by the user, message, and time sent included in the “Details” section of the notification. This is visualized below in Figure 1.

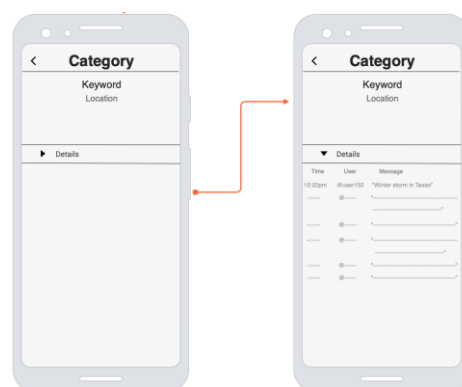


Figure 1. Companion Application

#### **4.4. Companion Website**

The companion website will output the following to the Twitter Parser:

{latitude; longitude}

The companion website will take the information from the ML algorithm in the format:

{ Category ; user; message ; location ; time sent ; keyword }

The location will first be used to sort the hazard into the appropriate user entered ZIP code, which will establish the users receiving the notification. The location, time, category, and keyword will also be used in the main heading of the notification. Followed by the message included in the table below the notification home.

### **5. Communications / Device Interface Protocols**

#### **5.1. Wireless Communications**

While the system will be running on a server, the user application will need to be able to communicate with it over the internet or cellular connection. We will use Amazon's Elastic Beanstalk to handle communication between the user application and the machine learning model.

# POWER SYSTEM ALARM APP USING SNS

Brandon Graeber

Monica Long

Patrick Martin

## **SCHEDULE**

REVISION – 1.2  
29 April 2023

## Execution Timeline



## ELECTRICAL & COMPUTER ENGINEERING

TEXAS A&M UNIVERSITY



	Feb 9 2/9/2023	Feb 23 2/23/2023	Mar 9 3/9/2023	Mar 23 3/23/2023	Apr 6 4/6/2023	Apr 20 4/20/2023		
<b>Reports and Presentations</b>								
CONOPS								Not Started
FSR / ICD (Feb 22)								In Progress
Execution / Validation Plan (Feb 22)								Completed
Midterm Presentation (Mar 8)								Behind Schedule
Status Update Presentation (Mar 29)								
Final Presentation (April 19)								
Final Report (Apr 19)								
<b>Twitter Parser</b>								
Establish Python to Twitter Connection								
Test Tweet Response Time								
Create Dataset for words or phrases to extract								
Test Operation and Elapsed Time								
Finalize code and Integration								
Manually validate the parsed data								
<b>Companion Website</b>								
Learning programming languages								
Website Creation								
Minimal User Interface								
Visual Concept Design								
Zip Code validation								
Full Featured Website								
<b>Machine Learning</b>								
Dataset Creation/Labeling								
Dataset Extension								
Dataset Preprocessing and Conversion to tensors								
Model Creation								
Recurrent Neural Network Building								
Choose Optimizer and Loss Function								
Saving and Loading Model								
Hyperparameter ML testing								
<b>Companion Application</b>								
Visual Concept Design								
Learning Java								
App Creation								
ZIP code to lat/long output								
Improve App Navigation								
Implement Map View with Markers								
Notification Page								
Store ZIP code								
App Improvement with Android Device								



# POWER SYSTEM ALARM APP USING SNS

Brandon Graeber

Monica Long

Patrick Martin

## **VALIDATION**

REVISION – 1.2  
29 April 2023

## Validation Plan



## ELECTRICAL & COMPUTER ENGINEERING TEXAS A&M UNIVERSITY



Paragraph #	Test Name	Success Criteria	Methodology	Status	Responsible Engineer
3.2.1.1	Machine Learning Recall	85% Recall	A separate testing dataset of 100 samples will be sent into the model with the parameter of interest being the recall	Passed	Brandon
3.2.1.2	Overall Operational Speed	< 1 minute completion time per tweet	Three category 1 and 2 tweets will be sent to the ML model. The timer will stop after the alerts are received from the companion application and website	Untested	Team
3.2.1.3	Twitter Parser Update Time	1 minute refresh/completion time	The Twitter parser will be timed to determine the speed. The time between each program execution shall be such that the overall refresh time is 1 minute or less	Passed	Patrick
	Twitter Parser Tweet Verification	Tweet Detection Response Time	The twitter parser will be ran after a tweet is posted, and should add the new tweet to the dataset in real time.	Passed	Patrick
3.2.1.4	Functionality of User Application	Fully featured application	Individually test each feature	Passed	Monica
	Separate App Sections			Passed	Monica
	Ability to Connect to a Server			Untested	Monica
	Notification System Response Time	< 1 minute		Untested	Monica
3.2.1.5	Functionality of Website	Fully Featured Website	Individually test each feature	Passed	Patrick
	Ability to Connect to a Server			Untested	Patrick
	Map Integration			Untested	Patrick
	Website Update Time	< 1 minute		Passed	Patrick

**Performance on Execution Plan**

All goals and objectives have been met, and only integration and upgrades to the systems will be made moving forward.

**Performance on Validation Plan**

The validation plan has a few variables that will be validated once the subsystems have been integrated.

# POWER SYSTEM ALARM APP USING SNS

Brandon Graeber

Monica Long

Patrick Martin

## SUBSYSTEM REPORTS

REVISION – 1.0  
20 April 2023

# SUBSYSTEM REPORTS FOR Power System Alarm App Using SNS

PREPARED BY:

---

Author Date

APPROVED BY:

---

Project Leader Date

---

Prof. Jang Date

---

T/A Date

## Change Record

Rev	Date	Originator	Approvals	Description
1.0	4/29/2023	Team		Draft Release

## Table of Contents

<b>1. Introduction</b>	<b>48</b>
<b>2. Social Media Parser Subsystem Report</b>	<b>49</b>
2.1. Subsystem Introduction	49
2.2. Subsystem Details	49
2.3. Subsystem Validation	50
2.4. Subsystem Conclusion	52
<b>3. Machine Learning Model Subsystem Report</b>	<b>53</b>
3.1. Subsystem Introduction	53
3.2. Subsystem Details	53
3.3. Subsystem Validation	54
3.4. Subsystem Conclusion	57
<b>4. Companion Application Subsystem Report</b>	<b>58</b>
4.1. Subsystem Introduction	58
4.2. Subsystem Details	58
4.3. Subsystem Validation	59
4.4. Subsystem Conclusion	62
<b>5. Companion Website Subsystem Report</b>	<b>63</b>
5.1. Subsystem Introduction	63
5.2. Subsystem Details	63
5.3. Subsystem Validation	65
5.4. Subsystem Conclusion	66

## List of Tables

No table of figures entries found.



## **List of Figures**

Figure 1: Example Tweets Output for College Station.....	50
Figure 2: Runtime Validation.....	51
Figure 3: Tweet Detection Response Time.....	51
Figure 4: Training Dataset Breakdown.....	54
Figure 5: Testing Dataset Breakdown.....	55
Figure 6: Scoring for Batch Model Testing.....	55
Figure 7: Example Batch Model Testing.....	56
Figure 8: Final Model Testing and Validation.....	56
Figure 9: Charlie Validation Results.....	56
Figure 10: Visualization of companion application.....	58
Figure 11: ZIP code length validation.....	59
Figure 12: Valid ZIP code validation.....	60
Figure 13: Internet connection validation.....	61
Figure 14: Orientation validation.....	61
Figure 15: Notifications Page.....	63
Figure 16: Pop-Up Notification.....	64
Figure 17: Help Page.....	65
Figure 18: Zip Code Validation - Incorrect Zip Code.....	65
Figure 19: Zip Code Validation - Inputting a Word.....	66

## **1. Introduction**

The power system alarm app using social networking service will use social networking to find potential hazards to power lines and systems and alert power grid companies. The system is broken down into social media parser, machine learning model, companion application, and companion website. The system obtains the users location to search for hazards in their area on Twitter, it then sorts the obtained tweets into non-threat, moderate threat, and immediate threat. The moderate and immediate threats are then sent to the companion application or website to alert the user. Each subsystem has been carefully designed and rigorously tested. As each subsystem has been validated to function correctly and meet all requirements, a straightforward approach to integrating them into the complete system specified in the CONOPS, FSR, and ICD is established.

## **2. Social Media Parser Subsystem Report**

### **2.1. Subsystem Introduction**

The Social Media Parser is designed to take in the inputs from the Companion Application or the Companion Website and return the applicable tweets as the outputs to the Machine Learning Model. These outputs are contained within a 25-mile radius of the requested Zip Code and will contain a keyword representing the potential hazard. The Parser has been extensively tested and can provide real-time data as requested.

### **2.2. Subsystem Details**

The Social Media Parser depends on one specific thing to operate, and that is the access to the Twitter Developer Portal, through the Twitter Developer Program. Access to the Portal was hard to come by but was finally achieved in the last week of the semester. From the Twitter Developer Portal, I was able to receive access keys to pull data and tweets directly from Twitter. These “API Keys” are what I used to operate the Tweepy python library, which is Twitter's official API for scraping data.

The subsystem begins by laying out the access keys and authenticating the access to twitter. From here, the scraper uses the latitude and longitude of the specific area defined as the location search basis, which will be the center of the 25-mile radius used in the search query. This 25-mile radius is the maximum distance supported by the library. Next, the code opens a CSV file, `Capstone_Words_and_Phrases`, containing a list of words or phrases to be flagged in tweets as a possible hazard. The contents of this file contain a single column of words or phrases that relate directly to natural disasters and common events that could contribute to a power system hazard, such as heavy thunderstorms, lightning, transformer issues, and more.

This file is then iterated through, and each word is set as the search query for an impending pull request to twitter, using the Twitter API's search function. For each tweet that contains one of the flagged words or phrases and has a location within the specified radius, the program collects the time the tweet was created, the user who posted the tweet, the text of the tweet, and the name of the location associated with the tweet, all into a list that is received as the “public\_tweets” list.

This action requires a check however, since the Twitter API's location search looks at both the location of the tweet, and the location that the user has defined in their Profile. Thankfully, we can differentiate between the two by looking at the tweet collected, since if a tweet is reported to have a location and it is displayed in the “Location” column of the received information, it means that the tweet itself had the location tagged to it. If the

“Location” column of the received information comes back empty, this means that the Twitter User has declared that location in their Profile, which means the tweet itself could have been sent from anywhere, and therefore is not a reliable location. Therefore, this is checked inside of the loop, and is processed before moving on to the next word.

If a received tweet is determined to be a hit, with a flagged tweet and a valid location, it is appended to a data list, which is later sent to a Pandas dataframe. The pandas dataframe was chosen because of its ease of operation to send to a file called tweets.csv file. This dataframe has all of the appropriate tweets at the end of the loop operation and is then sent to a .csv file which will be sent as the input to the machine learning model. This file contains the sections of Time, User, Tweet, and Location.

Time	User	Tweet	Location
2023-04-27 02:07:55+00:00	leugiM_21	*Me sitting next to a @LAFD firefighter in training*	College Station, TX
2023-04-26 19:57:21+00:00	patrickcapstone	downed power lines	College Station, TX
2023-04-26 20:03:21+00:00	patrickcapstone	Floods	College Station, TX
2023-04-26 20:03:13+00:00	patrickcapstone	floods	College Station, TX
2023-04-21 10:15:56+00:00	grid_events	Iola, TX (8:46 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Outage eventâ€¦ https://t.co/a7tKWRYH98	Texas, USA
2023-04-21 09:45:57+00:00	grid_events	College Station, TX (8:31 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Oâ€¦ https://t.co/lSzTgmpjH5	Texas, USA
2023-04-21 09:45:57+00:00	grid_events	College Station, TX (8:29 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Oâ€¦ https://t.co/bYlgUEl2VJ	Texas, USA
2023-04-21 09:30:57+00:00	grid_events	Bryan, TX (8:22 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Outage evenâ€¦ https://t.co/l8mj5VFsbY	Texas, USA
2023-04-21 09:30:56+00:00	grid_events	College Station, TX (8:16 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Oâ€¦ https://t.co/6HSjuiRhaC	College Station, TX
2023-04-21 09:15:56+00:00	grid_events	Iola, TX (8:00 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Outage eventâ€¦ https://t.co/vtHd9e5uYf	Texas, USA
2023-04-21 05:16:00+00:00	grid_events	Iola, TX (6:33 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Outage eventâ€¦ https://t.co/ZAOWGrKJNl	Texas, USA
2023-04-21 10:15:56+00:00	grid_events	Iola, TX (8:46 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Outage eventâ€¦ https://t.co/a7tKWRYH98	Texas, USA
2023-04-21 09:45:57+00:00	grid_events	College Station, TX (8:31 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Oâ€¦ https://t.co/lSzTgmpjH5	Texas, USA
2023-04-21 09:45:57+00:00	grid_events	College Station, TX (8:29 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Oâ€¦ https://t.co/bYlgUEl2VJ	Texas, USA
2023-04-21 09:30:57+00:00	grid_events	Bryan, TX (8:22 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Outage evenâ€¦ https://t.co/l8mj5VFsbY	Texas, USA
2023-04-21 09:30:56+00:00	grid_events	College Station, TX (8:16 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Oâ€¦ https://t.co/6HSjuiRhaC	College Station, TX
2023-04-21 09:15:56+00:00	grid_events	Iola, TX (8:00 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Outage eventâ€¦ https://t.co/vtHd9e5uYf	Texas, USA
2023-04-21 05:16:00+00:00	grid_events	Iola, TX (6:33 PM)Grid Power Outage Event&gt;&gt; The Ting Network detected an Electric Utility Grid Power Outage eventâ€¦ https://t.co/ZAOWGrKJNl	Texas, USA
2023-04-27 14:49:48+00:00	patrickcapstone	heavy thunder storms, 9:49am	College Station, TX
2023-04-27 08:26:01+00:00	patrickcapstone	heavy thunder storms, 9:25am	College Station, TX

Figure 1: Example Tweets Output for College Station

Figure 1 above gives the example output of a search in the College Station area that was performed on April 27th, 2022. As we can see there is data provided for the exact date and the exact time and is displayed in the Coordinated Universal Time (UTC). The Twitter API retrieves all data from the past seven full days, and therefore we will have to only consider tweets sent on that specific day when we integrate the subsystems.

### 2.3. Subsystem Validation

The Social Media Parsers validation had two feasible tests to complete, specifically the runtime being shorter than 1 minute, and the Tweet Detection Response Time. All tests are subject to the parser collecting the correct data, which has been proven by the Twitter API searching the tweets based off a singular word and location, ensuring that each received tweet is of the correct form.

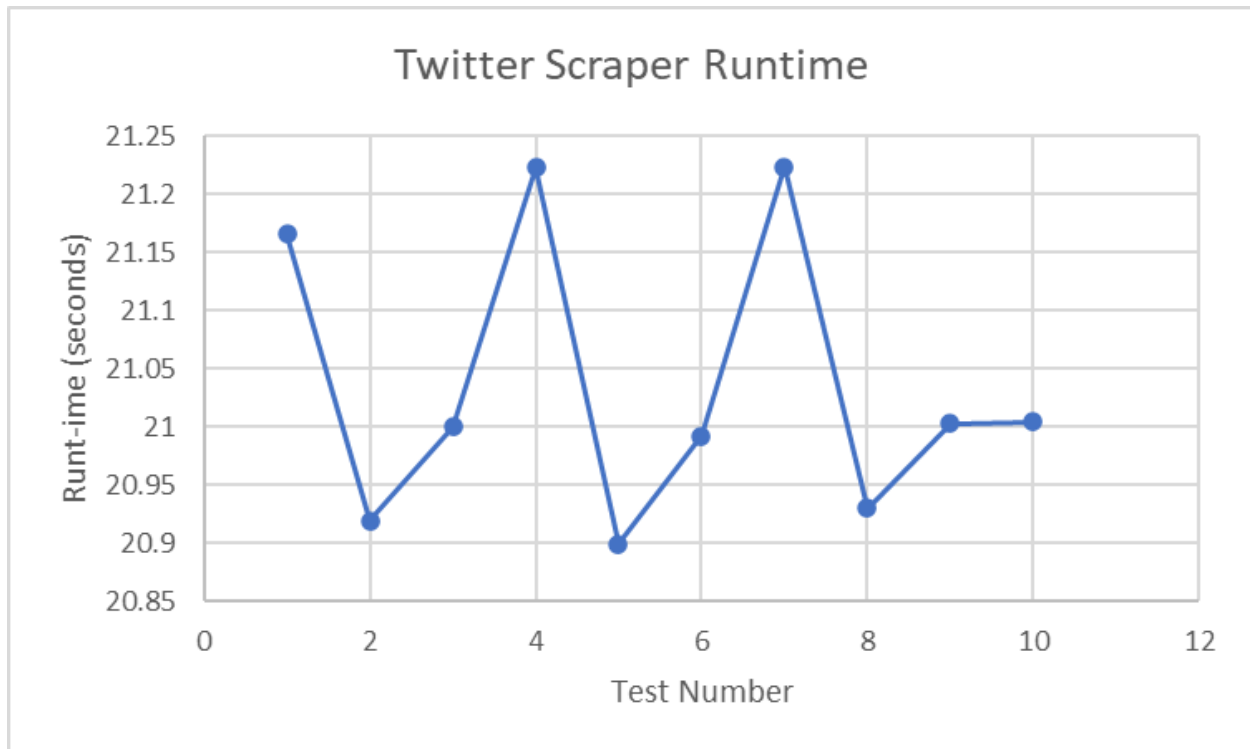


Figure 2: Runtime Validation

For testing the Runtime validation, I added a timer to the python code, and collected the time for 10 consecutive tests, on the night of a terrible thunderstorm, which would provide the most flagged tweets. Above in Figure 2, we can see that the execution time of the Twitter Parser had an average of 21.036 seconds, which is in fact under 1 minute.

We are aware that for each word or phrase that is added into the initial Capstone\_Words\_and\_Phrases.csv, we expect to add about 0.4 seconds to the total runtime. Therefore, the list has around 50 words or phrases currently that we have determined to be relevant. This number will change in the future of the project and into integration, as we will get the total runtime once the system is integrated and would also like to keep this total runtime under a minute.

Time	User	Tweet	Location
2023-04-27 14:49:48+00:00	patrickcapstone	heavy thunder storms, 9:49am	College Station, TX

Figure 3: Tweet Detection Response Time

The other validation that was performed was the Tweet Detection Response Time, and this was achieved by sending a tweet with the appropriate location tag, as well as a flagged word, and then running the python script to see if that tweet was picked up.

Above in Figure 3 we can see that the tweet was sent by myself at 9:49 am, during the final demo, and was picked up within the same minute that it was sent using the method described above. This was tested a multitude of times throughout the validation processes and picked up the respective tweet each time.

## **2.4. *Subsystem Conclusion***

Overall, the Social Media Parser works correctly, and is able to complete its most necessary function of collecting real time data. Improvements will need to be made regarding runtime once final integration is completed, and the words or phrases that are searched for will be edited to ensure that the total throughput is not bottlenecked by the parser, while still finding every tweet.

### **3. Machine Learning Model Subsystem Report**

#### **3.1. Subsystem Introduction**

The machine learning subsystem is designed to take input from the social media parser and output the same tweet to the companion app and website if the model confidently predicts that the tweet indicates a danger to the power grid.

#### **3.2. Subsystem Details**

It accomplishes this by taking the tweet text as a string input, converting it into an intermediate object called a tensor, and then predicting the corresponding category from the tensor. The output of this process is a list of probabilities, all summing to 1, corresponding to the individual categories. The index of the greatest probability is the machine learning model's prediction.

From this point, if the index or category of the tweet is at least of type 1 or 2, it has the chance to be passed on to the next subsystem. If the model's prediction probability is greater than a minimum confidence level, in this case 85%, then it will be passed along to the companion interfaces.

The first step in this subsystem is acquiring a dataset to train the future models on. For this project, I used a subset of a kaggle dataset called Natural Language Processing with Disaster Tweets. While this dataset provided a good base for my subsystem, it was a very general dataset that included key phrases that would not be applicable to the project. Therefore, I created a subset of the dataset for the project; however, the size of the subset was about 5000 entries long. This is not ideal for training a NLP model, so I utilized a subset of the previous ECEN 403 group's dataset (based on the recent Texas Ice Storm and the California 'Campfire' Wildfire). I wanted my model to remain a decent general case model, so I only used subsets of their data to supplement my growing dataset. Finally, I started manually adding tweets to my dataset by searching Twitter and copying the relevant results to my dataset. The Final Dataset is represented in Figure 4.

disaster\_training.csv Dataset with (6722 Samples)

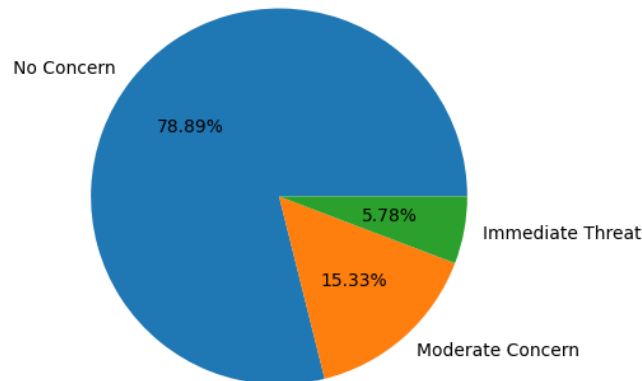


Figure 4: Training Dataset Breakdown

The next step in my subsystem is building and training different models. For models to be able to understand human speech, at least to an acceptable level, the model must have an idea about time or position. For example, a model must be able to tell the difference between “I thought that movie would be bad, but it was actually good” and “I thought that movie would be good, but it was actually bad”. Therefore, my model would include at least one Long-Short Term Memory (LSTM) Layer. Without getting technical, this layer allows for a model to ‘understand’ human sentences better.

Along with the LSTM layer, there will be some dense layers. To find out what combination of parameters would produce the best model, I created a batch model trainer program to cycle through each parameter (LSTM Layers, LSTM Nodes, Dense Layers, Dense Nodes).

### 3.3. Subsystem Validation

To validate the Machine Learning models, I used a separate testing set of 101 different tweets roughly split evenly across the 3 categories (See Figure 5). I then asked the model to predict the categories of the tweets and recorded different metrics to verify the efficiency of the model. Each model was tested on the following metrics: Unconfident Tweets, Confidently Right Tweets, Confidently Wrong Tweets, Perfect Predictions, Recall, Precision, and the Accuracy between category 1 and 2.

The models were then scored based on the values in Figure 6. The absolute values of the scoring do matter as much as the relationship among them. For example, a model



that is confidently wrong as much as it is confidently right not a good model and thus should be penalized.

For the purposes of this report Figure 7 serves as an example of the testing procedure done to acquire the final model parameters. A total of 125 models were tested; but for brevity, only a selection shall be displayed.

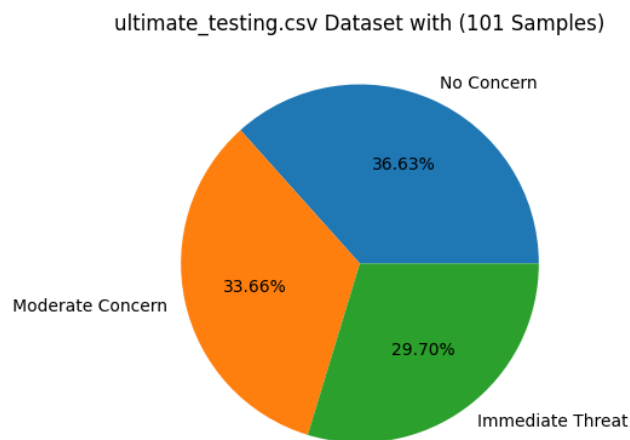


Figure 5: Testing Dataset Breakdown

Scoring						
Unconfident	Confident Right	Confident Wrong	Perfect Preds	Recall	Precision	Inner Accuracy
-4	4	-8	5	0.1	0.03	0.01

Figure 6: Scoring for Batch Model Testing

Model Parameters				Scoring							
LSTM Layers	LSTM Nodes	Dense Layers	Dense Nodes	Unconfident	Confident Right	Confident Wrong	Perfect Preds	Recall	Precision	Inner Accuracy	Total Points
1	64	1	64	25	46	14	56	93.1	83.1	74.1	262.682
1	64	1	128	17	53	15	53	89.7	82.5	71.2	299.363
1	64	1	256	7	62	16	55	91.4	85.5	69.8	377.575
1	64	1	512	20	50	15	55	91.4	84.1	71.7	285.552
1	64	1	1024	12	57	16	57	93.1	87.1	70.4	347.765
1	64	2	64	26	46	13	50	93.1	83.1	63	236.571
1	64	2	128	20	50	15	57	93.1	84.4	74.1	295.721
1	64	2	256	18	50	17	55	96.6	86.2	66.1	277.975
1	64	2	512	20	52	13	60	91.4	86.9	77.4	334.693
1	64	2	1024	20	51	14	56	91.4	85.5	71.7	302.594

Figure 7: Example Batch Model Testing

The top 10 models were then retested for consistency and re-ranked. Finally, the top 3 models from that batch were then retested for a final time. Figure 8 shows the final 3 models and the validation results.

					Scoring							
	l-layers	l-nodes	d-layers	d-nodes	Unconfident	Confident Right	Confident Wrong	Perfect Preds	Recall	Precision	Inner Accuracy	Total Points
Alpha	2	256	3	64	15	62	24	63	94	85.1	63.5	323.588
Beta	2	64	2	128	14	69	18	61	91	83.6	63.9	393.247
Charlie	1	128	4	512	19	64	18	69	89.6	87	73.3	393.303

Figure 8: Final Model Testing and Validation

As evident from Figure 8, the models codenamed Beta and Charlie were the best models with each of them being essentially equal in overall quality. For the purposes of this project, the model codenamed Charlie will be used for the final Demo.

```

Model Version (0.8_charlie)
Unconfident Tweets : 19
Very Confident Right : 64
Very Confident Wrong : 18
Perfect Predictions : 69
Good Enough Predictions : 85
Modified Recall : 89.6 %
Modified Precision : 87.0 %
The Inner Accuracy (between 1 and 2) is : 73.3 %

Time elapsed : 2.181 seconds

```

Figure 9: Charlie Validation Results

### **3.4. Subsystem Conclusion**

Overall, the machine learning model subsystem works to a satisfactory level, but it can still be improved. The models have difficulty understanding 'time based' tweets (i.e. tweets about a historical disaster instead of a current one) and 'photography' tweets (i.e. tweets about paintings or pictures of former/fictional disasters)

## 4. Companion Application Subsystem Report

### 4.1. Subsystem Introduction

The companion application subsystem is designed to take user input of a location and output any current and past hazards, up to one week, for a 25-mile radius of their input location. This companion application will be available on any android device and includes a contact page for concerns or questions. The application has been tested to verify that it functions properly and executes all tasks successfully and accurately.

### 4.2. Subsystem Details

Figure 10 includes visualization of the application.

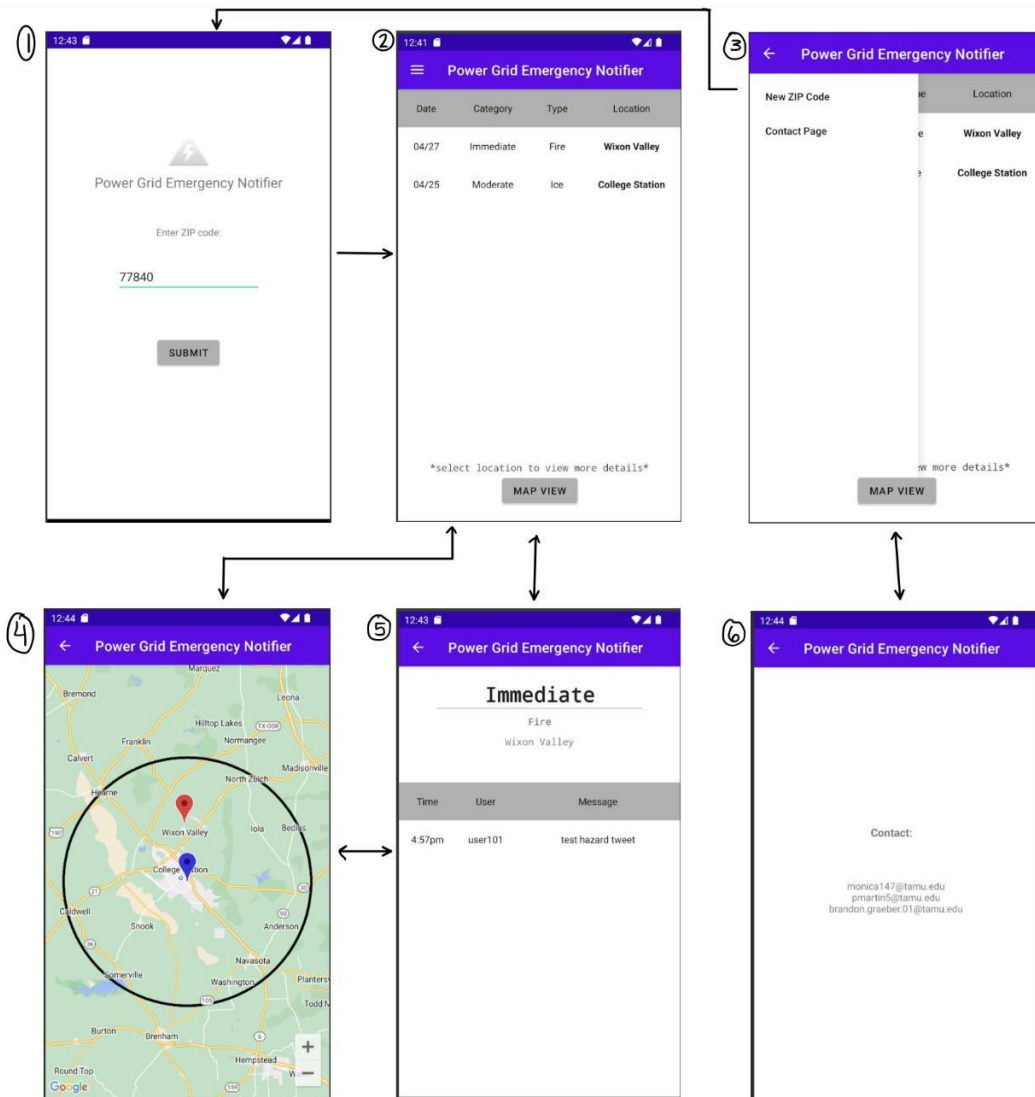
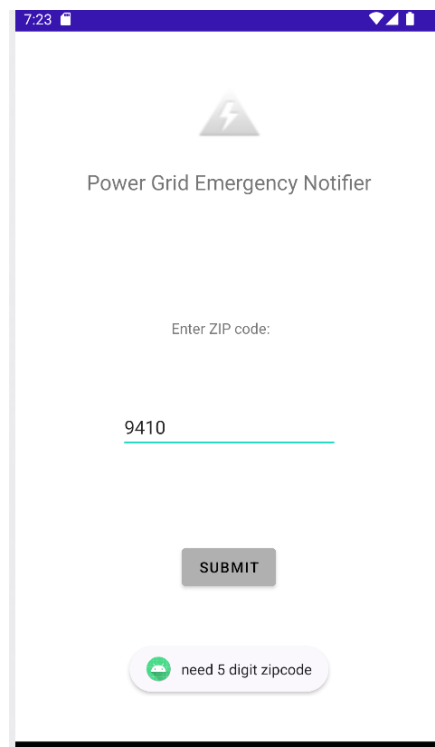


Figure 10: Visualization of companion application

Before the user has entered a ZIP code the landing page will be the first page in *Figure 10*, which will allow the user to enter a valid ZIP code of their choosing that will be converted to latitude and longitude to be sent over to the parser for scraping Twitter. Once the user has entered their ZIP code, the application will store the entered zip code until they select 'New ZIP code' on the navigation menu. After the ZIP code has been stored the landing page for the user will become the second page in *Figure 10*. The second page will include a navigation menu for the user that is shown on the third page on *Figure 10*, which includes the option to enter a new ZIP code or access the contact page. The second page also includes a button to view the map view that is located on page four. As prefaced on the second page, if the user selects the location on the hazard the fifth page will open to show more details of the event selected. The fifth page will also open if a marker is selected on the map view which is page four on *Figure 10*. Finally, the contact page is shown as page six and can be accessed through the navigation menu on the third page.

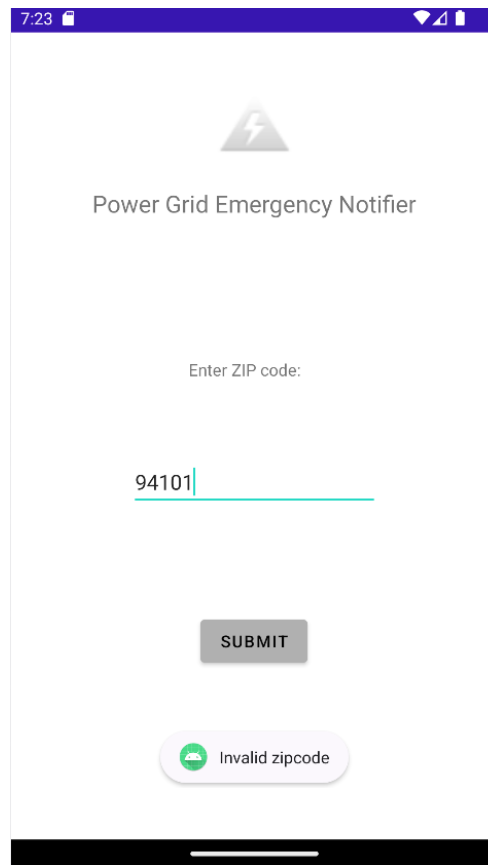
### 4.3. Subsystem Validation

This subsystem has been validated in the areas of user input, navigation, internet connection, and orientation. The user input does not allow the user to type anything other than numbers, it also verifies the length of the ZIP code is five digits as shown in *Figure 11*.

The image is a screenshot of a mobile application interface. At the top, there is a status bar with the time 7:23 and various icons. Below the status bar is a header with a lightning bolt icon and the text "Power Grid Emergency Notifier". The main content area has a label "Enter ZIP code:" followed by a text input field containing the number "9410". Below the input field is a "SUBMIT" button. At the bottom, there is a green circular icon with a lightning bolt and the text "need 5 digit zipcode", indicating a validation error because the entered ZIP code is only four digits long.

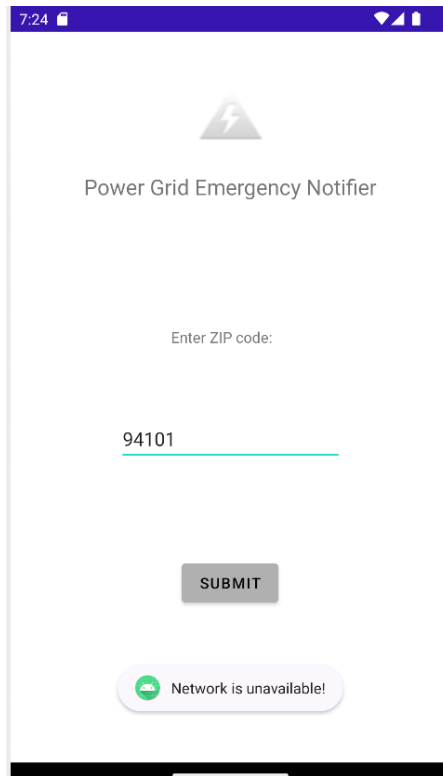
**Figure 11:** ZIP code length validation

Once the user has typed a five-digit number the application will check if it is a valid ZIP code and if it is not, the user will receive a notification as shown in *Figure 12*.



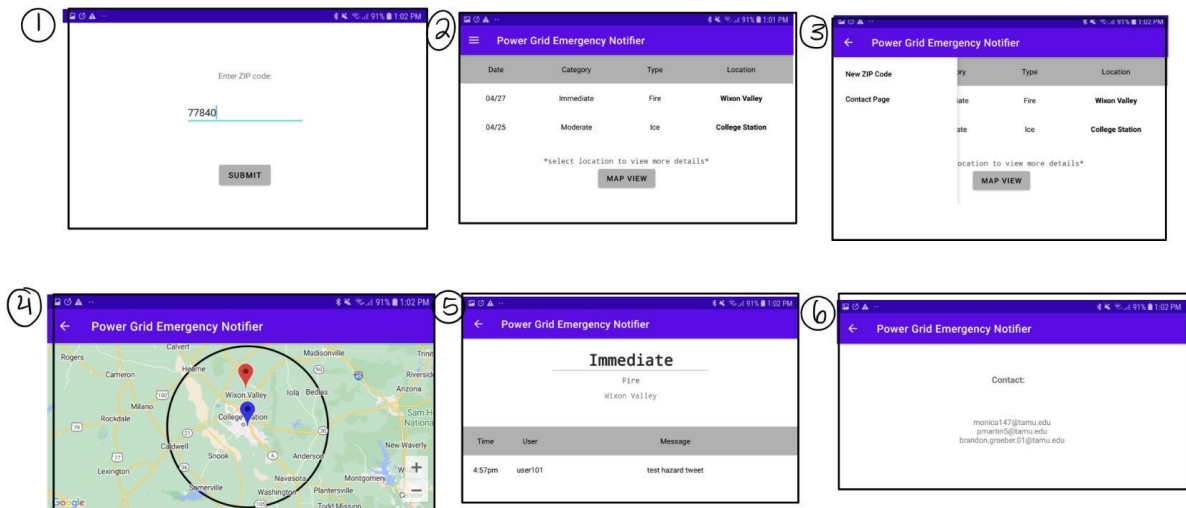
**Figure 12:** *Valid ZIP code validation*

Navigation is another validation that has been passed through all pages being accessible and having back buttons or navigation menus. Another validation is internet connection, the application uses the internet to convert the users ZIP code to latitude and longitude, if there is no internet access the application will send a notification as shown in *Figure 13*, instead of crashing the application.



**Figure 13:** Internet connection validation

Finally, the application has been validated for orientation, the application works in both portrait and landscape as shown in *Figure 14*.



**Figure 14:** Orientation validation

#### **4.4. Subsystem Conclusion**

Each part of the subsystem was shown to work correctly. The user application effectively takes user input and displays current and past hazards in a list and map format with each hazard being clickable for more information. When integrated with other subsystems, it will be able to take in real time information to report to the user in an efficient manner.



## 5. Companion Website Subsystem Report

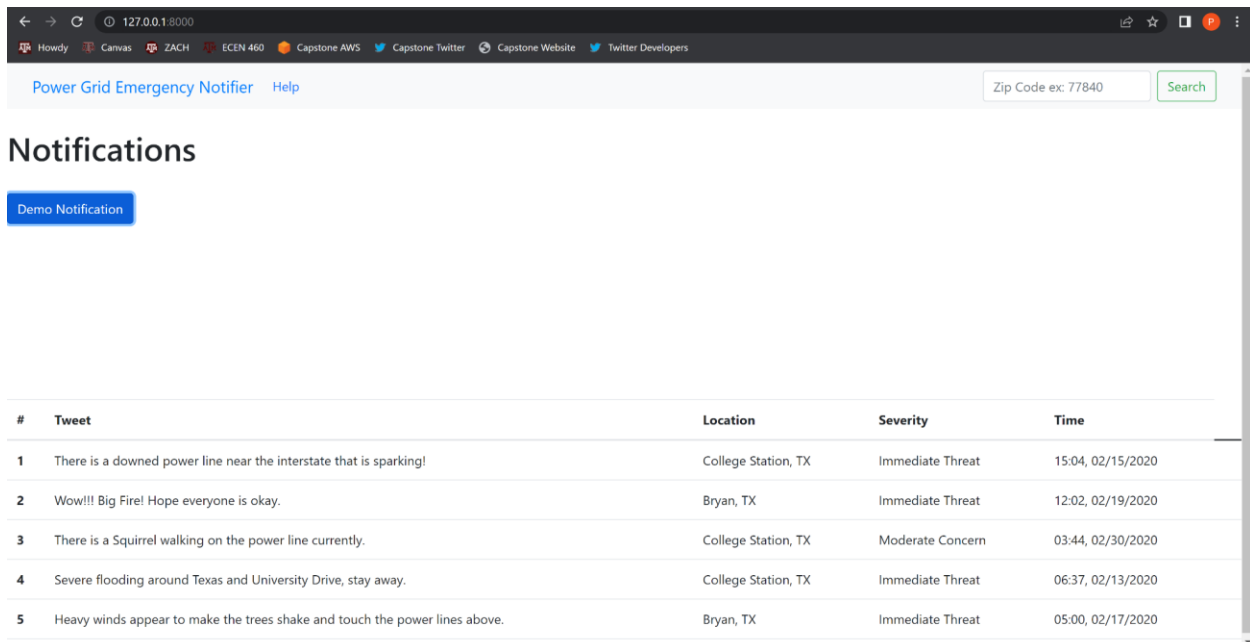
### 5.1. Subsystem Introduction

The Companion Website is designed to take in the inputs from the Machine Learning Model and display a notification when a hazard is found. This notification will contain the category, location, time, and keyword. Below the notification region will be a table of current and past notifications that will include the Tweet, date and time, location, and category of the tweet. The website includes a notifications and help page and has been thoroughly tested for its operation.

### 5.2. Subsystem Details

The Companion application was built using the Django Framework for the backend development of the website. The Django framework that is used in the website handles all low-level details of the HTTP protocols and presents the template that we built all pages, structures, and views from.

The pages that were built on the framework used HTML for formatting, and Bootstrap framework for making the User Interface look good. Below are the different pages and what they do.



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000'. The page title is 'Power Grid Emergency Notifier' with a 'Help' link. A search bar contains 'Zip Code ex: 77840' and a 'Search' button. Below the header, there is a 'Demo Notification' button. The main content area features a table with the following data:

#	Tweet	Location	Severity	Time
1	There is a downed power line near the interstate that is sparking!	College Station, TX	Immediate Threat	15:04, 02/15/2020
2	Wow!!! Big Fire! Hope everyone is okay.	Bryan, TX	Immediate Threat	12:02, 02/19/2020
3	There is a Squirrel walking on the power line currently.	College Station, TX	Moderate Concern	03:44, 02/30/2020
4	Severe flooding around Texas and University Drive, stay away.	College Station, TX	Immediate Threat	06:37, 02/13/2020
5	Heavy winds appear to make the trees shake and touch the power lines above.	Bryan, TX	Immediate Threat	05:00, 02/17/2020

Figure 15: Notifications Page

As seen in Figure 15, the Notifications Page has a navigation bar that uses the bootstrap navbar tool, a query for entering the Zip code of the respective area, A button to demonstrate the format of the notification that will be shown in Figure 16, and a table displaying the current and past information from past notifications in the area.

In the top right of Figure 15, we can see the query bar which recommends a specific format for the zip code. This bar takes in the zip code as an input, and checks to make sure that the zip code is strictly a five-digit number, with no other punctuation or words. This will be displayed in the validation section below. If the format is correct, the zip code will be saved to the website, and will later be sent to the Twitter Scraper as an input during integration.

At the top of the page, we can see the navigation bar that can switch between the home and help pages and includes the query bar. At the bottom of the notifications page, the table is set up to report the notifications in the specific area. The table reports the Tweet, Location, Severity, and Time of the tweet. The table is formatted to fit the webpage and will vertically expand as the number of notifications goes up.

The screenshot shows the 'Power Grid Emergency Notifier' interface. At the top, there is a navigation bar with 'Power Grid Emergency Notifier' and 'Help' links. On the right, there is a 'Zip Code' input field with the example '77840' and a 'Search' button. The main heading is 'Notifications'. Below it, there is a 'Demo Notification' button. A modal window titled 'POSSIBLE DANGER' is open, displaying a notification with the following details: 'Immediate Threat' (category), 'College Station, TX' (location), '15:04, 02/15/2020' (time), and 'Fire' (keyword). The modal has a 'Close' button. Below the modal, there is a table with the following columns: '#', 'Tweet', 'Location', 'Severity', and 'Time'.

#	Tweet	Location	Severity	Time
1	There is a downed power line near the interstate that is sparking!	College Station, TX	Immediate Threat	15:04, 02/15/2020
2	Wow!!! Big Fire! Hope everyone is okay.	Bryan, TX	Immediate Threat	12:02, 02/19/2020
3	There is a Squirrel walking on the power line currently.	College Station, TX	Moderate Concern	03:44, 02/30/2020
4	Severe flooding around Texas and University Drive, stay away.	College Station, TX	Immediate Threat	06:37, 02/13/2020
5	Heavy winds appear to make the trees shake and touch the power lines above.	Bryan, TX	Immediate Threat	05:00, 02/17/2020

Figure 16: Pop-Up Notification

In Figure 16 above, we can see that the notification dims the screen around the notification, then outputs the necessary data including the category, location, time, and keyword that is received from the machine learning model. The notification itself is a Modal derived from the bootstrap library, and acts as a pop-up notification that requires the user to close it out to ensure that they see it. Within the Modal is the Bootstrap Danger alert that shows the data from the machine learning model in red, as another way to get the user's attention.

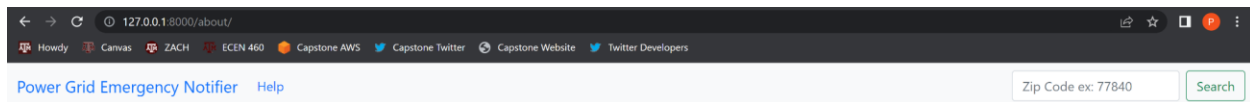


Figure 17: Help Page

Above in Figure 17 we have the Help page, that gives directions to contact any of the members if help is needed. This page can be accessed by clicking the help tab at any time.

### 5.3. Subsystem Validation

Website validation can be divided into four specific tests, specifically zip code input validation, fluid switching between pages, notification testing, and webpage update response time.

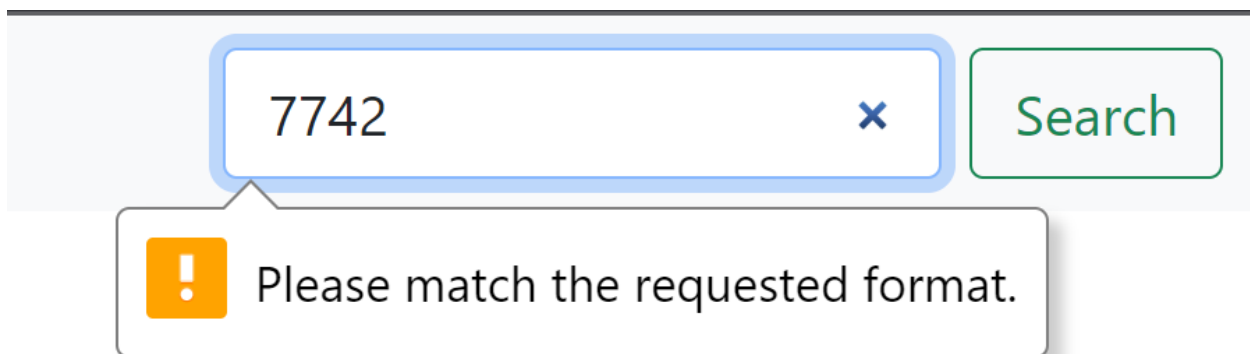


Figure 18: Zip Code Validation - Incorrect Zip Code

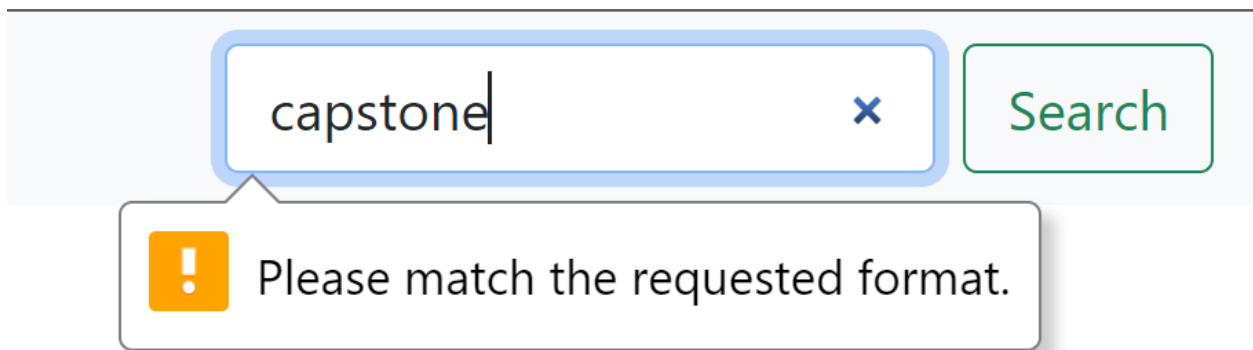


Figure 19: Zip Code Validation - Inputting a Word

As we can see from the above in Figures 18 and 19, the zip code query will only take in and save values that are five digit numbers and will block the search button and request to match the requested format if the value entered is not a 5 digit number. Unlike the companion application, this does not check to see if it is a valid zip code but will in fact check this when we integrate in the future.

We can see above in Figures 15 and 16 that the pop-up notification does in fact work, and when exited out of the notification, it will bring you back to the notifications screen. and also within Figures 15, 16, and 17 that the pages can be fluidly switched between pages. The update time is currently real time updates, since the website is hosted on a local machine. This will change in the future when we integrate onto a server and make the website public, where the server response time, and overall operational time will determine the timing of the Companion Website. All code representing the website can be found in the GitHub repository, where validation of the website's operation can be seen.

#### **5.4. Subsystem Conclusion**

The Companion Website is fully operational and works effectively. The website displays current and past notifications within the applicable radius that the user inputted. Moving forward into integration, a map view and zip code conversion to latitude and longitude will be added to parallel the Companion app. Once integration with the other subsystems is complete, the Companion Website will quickly and efficiently take in real time information and display the applicable data to the user.