Table of Contents

1. INTRODUCTION

1.1 PURPOSE

The aim of the program is to build a web page which uses twitter API for detection certain natural disasters such as earthquakes, typhoons ect. With constantly listening tweets around the world. The ultimate purpose of the application is to build an early warning system which may save lives of people.

1.2 DOCUMENT CONVENTIONS

DB stands for database.

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

This project is designed as a class activity which may lead a bigger research in future. Anybody interested in the topic are welcome to read and contribute.

1.4 PROJECT SCOPE

This projects scope is huge. But for beginning and simplicity first I limited it just for detection of earthquakes. This means the application should listen for tweets which are or can be related to Earthquakes and also should try to detect the location for the tweet. Density of tweets in an area is a deciding factor to detect the location for earthquake. The aim is to determine as fast as possible and sending notifications for each registered user. (Maybe only the users which close to detected area of earthquake)

1.5 REFERENCES

I strongly suggest you to read this official twitter blogpost below.
How the USGS uses Twitter data to track earthquakes:
https://blog.twitter.com/official/en_us/a/2015/usgs-twitter-data-earthquake-detection.html

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

There should be a database which records for tweets about earthquakes which should record data for further researches. When a detected earthquake has also officially verified by an institute like Kandilli Rashatanesi or USGS (United States Geological Survey) we can assume that our system is working well.

And we are going to store user information for our application. We may inform them with a notification to their phones. If there are no internet we can also send a SMS to users which informs them about earthquake. We also need to store phone number of our registered users.

For tweets:

-Tweet content:
It should contain whole tweet which has some keywords about Earthquake.
-Owner of tweet:
It should contain name of the owner
-Location of the tweet:
It should try to locate and store where is the twit coming from.
-IsVertified:
Vertification should be stored true if the tweet is officially verified.

For users:

-User Name:
It should contain user name.
-User Password:
It should cointain password for the user.
-User Adress:
It should contain address of the user. Regardless where user is user will be able to notify any detection about the place where is the entered address located. (The app should also provide location information.)
-Registered Locations:
The other places which user decide to listen for earthquakes may stored.

2.2 PRODUCT FEATURES

The features of our database system will be here as entity-relationship model in future.

2.3 USER CLASS and CHARACTERISTICS

The user should only be able to change information about himself. (Username, Address). User can select other places to listen. Not now but in further development cycles user maybe connect with their relatives and be notified if one of the relatives in a dangerous area.

2.4 OPERATING ENVIRONMENT

-Client/server system
-Operating System: Multiple
-Database: mySql (for beginning).
-Platform: Python\Django

3. SYSTEM FEATURES

3.1 DESCRIPTION and PRIORITY

The earthquake detection is the highest priority for this system and it is also the hardest part to achieve. And after detecting correctly notifying users is an issue. And also creating an user account.

3.2 STIMULUS/RESPONSE SEQUENCES

Twitter api related:
-Listen earthquake related twits
-Create earthquake notification
-Send notification to the users which following the location information
-Store information about tweets
Application related:
-Create user
-Store user information to db
-Let user select destinations to listen
-Notify when earthquake notification catched

3.3 FUNCTIONAL REQUIREMENTS

DATABASE:

A database is a collection of information that is organized so that it can be easily accessed, managed and updated.

CLIENT/SERVER SYSTEM:

The term client/server refers primarily to an architecture or logical division of responsibilities, the client is the application (also known as the front-end), and the server is the DBMS (also known as the back-end).
A client/server system is a distributed system in which,

- Some sites are client sites and others are server sites.
- All the data resides at the server sites.
- All applications execute at the client sites.

4. EXTERNAL INTERFACE REQUIREMENTS

4.1 USER INTERFACES

-Frontend software: Vue.js or React
-Backend software: Python/Django

4.2 HARDWARE INTERFACES

-Mac
-A modern browser

4.3 SOFTWARE INTERFACES

Software used	Description
Backend	Phyton/Django
Database	MySql
Frontend	Vue.js

4.4 COMMUNICATION INTERFACES

This project shall support all types of web browsers.

5. NONFUNCTIONAL REQUIREMENTS

This part will be updated