// Write-up of the idea under the theme **Disaster Management**

**1. PoC Solution design:**

**Technologies used:**

Java, MySQL(for DB), REST, OSGI, Maven and Apache Karaf as base container.

The implementation is simulated and shown in two REST calls which is accessed to determine the weather stats given a location/latitude-longitude of a place and will predict if drone must be launched in the area based on the signal strength and the stats.

Two REST calls are in place which predicts the live and the test weather stats and identifies the launch status of the drone.

Example:

GET <http://127.0.0.1:8181/aichallenge/v1/liveweatherstat?latitude=13.0336191&longitude=80.2686031&ASU=100>

Live weather works by giving the latitude, longitude and the ASU (Arbitrary Strength Unit is an [integer](https://en.wikipedia.org/wiki/Integer) value proportional to the received signal strength measured by the mobile phone). OpenWeatherMap 2.0 is used in predicting the weather status of the given area. The REST call to OpenWeatherMap is issued from the Java code

The outcome of the REST call is the weather condition in the area and prediction if the drone must be launched in the area.

The same is updated in the database table live\_table (in MYSQL) which is also used as learning/training set.

The second REST call works by taking the location in the URL (where latitude, longitude) is already mapped in the DB and predicts the weather stat and the drone launch status.

GET

<http://127.0.0.1:8181/aichallenge/v1/testweatherstat?location=Adyar&ASU=500>

Higher the value of ASU symbolizes a higher value of DBM which predicts the drone must be launched in that area. The location’s latitude and longitude are fetched from the table “location\_latlongmapper” in DB where the values are already fed by the CSV file.

**2.1 Set of DB commands needed to run these REST calls:**

MySQL as database is needed to execute this idea. Install MySQL and run these commands to create a database and the tables used in the code to store the values.

To load the csv file values in the MySQL, place the “Area\_LatLongMapper” csv file in the Data folder of MySQL after the database AI is created.

#Creating database

create database AI;

show databases;

use AI;

#Used AI database in the code to read and write to tables

#Creating Live data table

create table live\_table( area varchar(100), lat varchar(100), lon varchar(100), weather varchar(100), temp varchar(100), pressure varchar(100), humidity varchar(100), wind\_speed varchar(100), launchDrone varchar(100));

select \* from live\_table;

#Creating sample mapping of location with latitude and longitude

create table location\_latlongmapper(area varchar(100), lat varchar(100), lon varchar(100));

#Loading the values of csv into sample location mapping table

After the database is created, place the csv files inside the zip in the folder where database is available from installation directory – this will facilitate the load of the csv data into the sql table.

C:\ProgramData\MySQL\MySQL Server 8.0\Data\ai

load data infile 'Area\_LatLongMapper.csv' into table location\_latlongmapper fields terminated by ',' lines terminated by '\n' (area,lat,lon);

select \* from location\_latlongmapper;

**2.2 After the DB is setup, either use the apache-karaf in the zip which serves as the container or download the apache-karaf-3.0.8**

Inside etc folder of apache-karaf, in the file “org.ops4j.pax.web.cfg”, change the directory value where jetty.xml is pointed. Also, make sure the port 8181 is listening on the machine where the karaf is set-up.

Inside etc folder of apache-karaf, in the file customconfig.cfg, change the username, password and port value which is used for connection from the Java code to the database. The default values which comes with installation of MYSQL is given in that file.

Once apache-karaf is setup, from the bin, start the karaf bat file which will launch the karaf console and make sure the application is ready to receive the calls.

**2.3 Java code**

“weather.prediction” given in the zip file is a maven project source code. The classes are OSGI’fied to be able to run in a modular way.

The main class is WeatherPredictorImpl which invokes the REST of the classes. The main advantage of using OSGI is achieving code modularity and reusable design. Thus, the need for OSGI container (Apache Karaf). Build the maven project and place the jar file inside apache karaf deploy folder.

**3. Algorithm/Models**

1. REST is used to simulate the weather prediction and determination in the given area.

2. Basic concept of reinforcement learning is used to determine if the drone must be launched in the given area given the condition of the weather and signal strength based on the value of ASU/DBM.

3. Values of initial score, reward and error factor are used to calculate the prediction of the drone launch. The calculation is given in the class “DroneLaunchDeterminer.java”.

**4. Competitive analysis/Business impact**

This idea is useful in the case where the signal strength is not available in the area during the aftermath of a disaster such as floods. The drone uses Ericsson’s 5G Radio dot system which can serve the signal to the area where there is no/very less signal strength.

This implementation provides a simulation which identifies the weather status and predicts if the drone must be launched in that area. Very useful in terms of disaster management and for helping the people to make necessary calls during that time.