Volume: 5 Issue: XII Month of publication: December 2017
DOI:

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor :6.887
Volume 5 Issue XII December 2017- Available at www.ijraset.com

Design and Implementation of Location Based Weather Application on Android Platform

Bhagyasri G. Patel[1]
[1]Assistant Professor, Department of Computer Science & Engineering, Faulty of Engineering Technology & Research, Bardoli, India

Abstract:* Android is currently the fastest growing mobile platform. One of the fastest growing areas in Android applications is Location Based Services (LBS). LBS is a platform that provides information services based on the current or a known location, supported by the Mobile positioning system. Presently, MOSDAC (Meteorological and Oceanographic Satellite Data Archival Centre) disseminates the weather forecast information through web. In this paper, we have proposed an application for disseminating location based weather forecast application on Android platform. It provides weather forecast information as per user's location or location of interest. This application provides several weather alerts for the events such as heavy rain, fog, smoke, and fire. It also provides Nowcast information for heavy rainfall alerts over Uttarakhand and Himachal Pradesh states of India and forecast information for cyclone all over India. As MOSDAC is a client-server application, selection of data communication protocols and data exchange formats plays important role. The following limitations of mobile device must be considered while developing a mobile application: memory size, security, response time, and battery consumption. In this paper, we have used RESTful web service, HTTP as communication protocol and JSON as a data exchange format for minimizing these limitations. For the weather forecast, we have used MOSDAC database. We selected substitution method over GPS because this method uses low energy location-sensing mechanisms compared to GPS. We also compare the battery consumption of MOSDAC Application with other such applications.

Keywords: Android; Location Based Services; Web service; Response time; Battery Consumption

I. INTRODUCTION

Weather forecasts and warnings are most important services provided by the meteorological profession. Weather forecasts are used by both government and private industries to protect life or properties and plan a wide range of daily activities. Some of the most commonly known severe weather advisories are high wind, flood, cyclone, smoke, fire, and fog. Presently, severe weather advisories and alerts are disseminated by MOSDAC (Meteorological and Oceanographic Satellite Data Archival Centre) [1] through web. In this paper, we have proposed a lightweight and interactive android application for dissemination of weather forecast details, provided by MOSDAC. We developed our mobile application on Android platform because it is one of the most widely used mobile operating system (OS) these days.

There are various application areas where weather forecast is used regularly. The aviation industry is sensitive to the weather and accurate weather forecasting is essential to manage and control air traffic. Farmers rely on weather forecasts to plan their day to day farming tasks. Forestry departments require weather forecast of wind, precipitations, and humidity for preventing and controlling wildfires. Electricity companies also rely on weather forecasts to predict the demand. Other commercial companies pay for weather forecasts so that they can increase the profits or avoid large losses. All these examples have been a motivation to carry out research in the field of weather information dissemination on smart phones.

Following weather information need to be disseminated in MOSDAC weather application: (1) Forecast - current weather conditions and 24, 48, and 72 hours forecast of following weather parameters: temperature, humidity, rain, wind speed, and sky conditions. Topographical information (land, mixed, and ocean) is provided at the user's point of interest. Weather alerts for cyclone and expected as well as observed cyclone tracks for visualization over a map. (2) Events - disseminating information for heavy rain, fog, smoke, and fire all over India (3) Nowcast - disseminating Nowcast (Half hourly forecast valid for 6 hours) information related to heavy rain and cloudburst over Himachal Pradesh and Uttrakhand states of India.

It is very important to keep the mobile application lightweight. The following parameters need to be focused while developing mobile application: memory capacity, security, response time, and battery consumption. In this paper, we have used the client-server architecture for the development of MOSDAC weather application. We have used RESTful web service, HTTP as a data exchange protocol, and JSON as a data exchange format for keeping the application lightweight. We have also proposed and implemented an algorithm for minimizing the battery consumption and response time [2].

The remainder of the paper is organized as follows. In Section II, we discuss related projects and features provided by these projects. We also discuss the limitations of these projects. Section III presents the architecture and methodologies used during application development. We discuss the system prerequisites and required features in Section IV. We discuss the obtained results for battery consumption and response time parameters in Section V. Discussion of the results are presented in Section VI. Finally, we conclude this paper stating the future plan in Section VII.

II. RELATED WORK

We have studied different Android applications providing weather forecast information. One of them is Accuweather [3]. Accuweather [3] provides access to the location based weather data via a simple RESTful interface to subscribers. Accuweather Application provides access to current weather data for any location on earth including 200,000 cities. Data is available in JSON, XML, or HTML format. This application collects data from professional as well as private weather stations. The majority of them are professional stations which are installed at large cities, airports etc. The service also collects weather data from private stations that are collected and installed by fans and weather devotee. This service also provides the details of following weather parameters: temperature, humidity, pressure, sea-level, wind speed, cloudiness, and rain.

Weather Bug [4] is an Android application that brings the current weather and the forecast information on your android device. The application uses mobile device's GPS service to decide the region for fetching weather information. By default, this application updates its information every hour. The useful feature of this application is its battery monitoring option. This option will disable updates when the application detects that the phone battery level is below threshold level. This application provides information on the current temperature, wind speed and direction, wind chill, recorded highs and lows for that day, humidity level, amount of rain, and the sunrise and sunset times for the day. The application collects data from its own sensors and integrates data from sources such as the National Weather Service (NWS) and the World Meteorological Organization (WMO), providing users access to the biggest global network of professional weather stations.

In the application such as Accuweather and Weather Bug, originating source and dissemination source of data is different. In MOSDAC weather application, originating source and dissemination source of data is same. MOSDAC majorly uses Indian National Satellite (INSAT) Series data in conjunction with MEGHATROPIQUES data and initial boundary conditions from NCMRWF (India) and data are collected from ISRO's Automatic Weather Stations (AWS). Other applications also not provide the several weather alerts such as cyclone, heavy rain, fog, smoke, fire over Indian region. The parameters for MOSDAC weather application are temperature, humidity, wind speed, cloudiness and rain.

III. PROPOSED ARCHITECTURE

Location-based MOSDAC Weather Application is implemented on android platform, which is an open source operating system for mobile devices such as smart phones and tablet computers. Android was developed by Open Handset Alliance, led by Google. Android allows the developers, wireless operators, and handset manufacturer to make new applications and products at lower cost. So, Samsung, HTC, and many other companies are using this open source operating system for their smart phones with a very considerable price which is the cause of availability of smart-phones to almost every person. Android has its own language for developing location and internet based application [5].

The proposed mobile application is based on Service Oriented Architecture (SOA) where we have thin client like Android Phones. Services are self contained and communicate using JSON and XML messages. Services are cross platform, asynchronous, reliable, and secured. Once we have a web service in remote server with centralized database then we can use the same web service for different clients such as Android, iPhone, Blackberry, Windows phone, and Symbian phones. Different clients can use the common web service to save or retrieve the data. The architecture used for developing the Android application is shown in Figure 1.

A. Components of Architecture

While developing a client-server application, the communication between client and database server becomes imperative. Therefore, it is important to select appropriate web service, data exchange protocol, data exchange format, and Mobile positioning technologies while developing client-server application. There are mainly four components in the architecture.

1) *Web Services:* The Web services maintain client and service provider communications through protocols where client makes a request and service providers provide the responses. [6].
2) *Data Exchange Format:* Data exchange format defines the data structure for communication between the requester and the service provider to formulate requests and responses in a simple way. Examples could be XML, JSON and KML [7].

3) *Data Exchange Protocols:* For establishing connection between client and server in Mobile application, it is important to decide the best way to exchange information between client and server for minimizing the limitation of mobile devices such as response time, network traffic, and resource utilization. Examples could be Sockets and HTTP [8].
4) *Mobile Positioning Technologies:* Location based services are the mobile services in which the user's location information is used to provide a service. The location information consists of latitude, longitude, and altitude generated by any given positioning technique [9] Examples could be Cell-ID, GPS, and Wi-Fi based techniques.

B. *Flow of Architecture*

Here, we explain the data flow of the architecture, presented in Fig. 1. Android device fetches the input such as user's current location (latitude-longitude) using different mobile positioning technologies such as Global Positioning System, WiFi Positioning System, and Cell-id Positioning System. Android device then creates URL with run-time parameters (latitude-longitude) and calls the RESTful web service. Device uses HTTP to open the URL and awaits response. RESTful web service is responsible for querying database, retrieving results, converting them into JSON format and sending JSON response by using HTTP protocol. Mobile device receives JSON response and parses JSON objects.

For system perspective of MOSDAC Weather Application, we have maintained a central server which is a database containing wide range of information of weather stations and information related o weather parameters around all over India. We have also employed haversine formula [10] for calculating the distances between user's current location and the nearest the weather stations. Therefore nearest weather station can be fetched by using Haversine formula and weather details for the particular weather station will be provided to the user. The formula is presented in equations (1) to (3):

$$d = R.c \qquad (1)$$

Here, R = earth's radius (mean radius = 6,371km)

$$c = 2.atan^2\left(\sqrt{a}, \sqrt{(1-a)}\right) \qquad (2)$$

$$\Delta lat = lat2 - lat1$$
$$\Delta long = long2 - long1$$

$$a = 2\,arcsin\left(\sqrt{sin^2\frac{\Delta\theta}{2} + cos\,\theta_1\,cos\theta_2\,sin^2\frac{\Delta\phi}{2}}\right) \qquad (3)$$

Where "d" is the distance between two places considering as two points. "lat1", "lat2" stand for latitudes, "long1", "long2" stand for longitudes of two points and "Δlat" stands for difference in latitude of two points and "Δlong" stands for difference in longitude of two points.

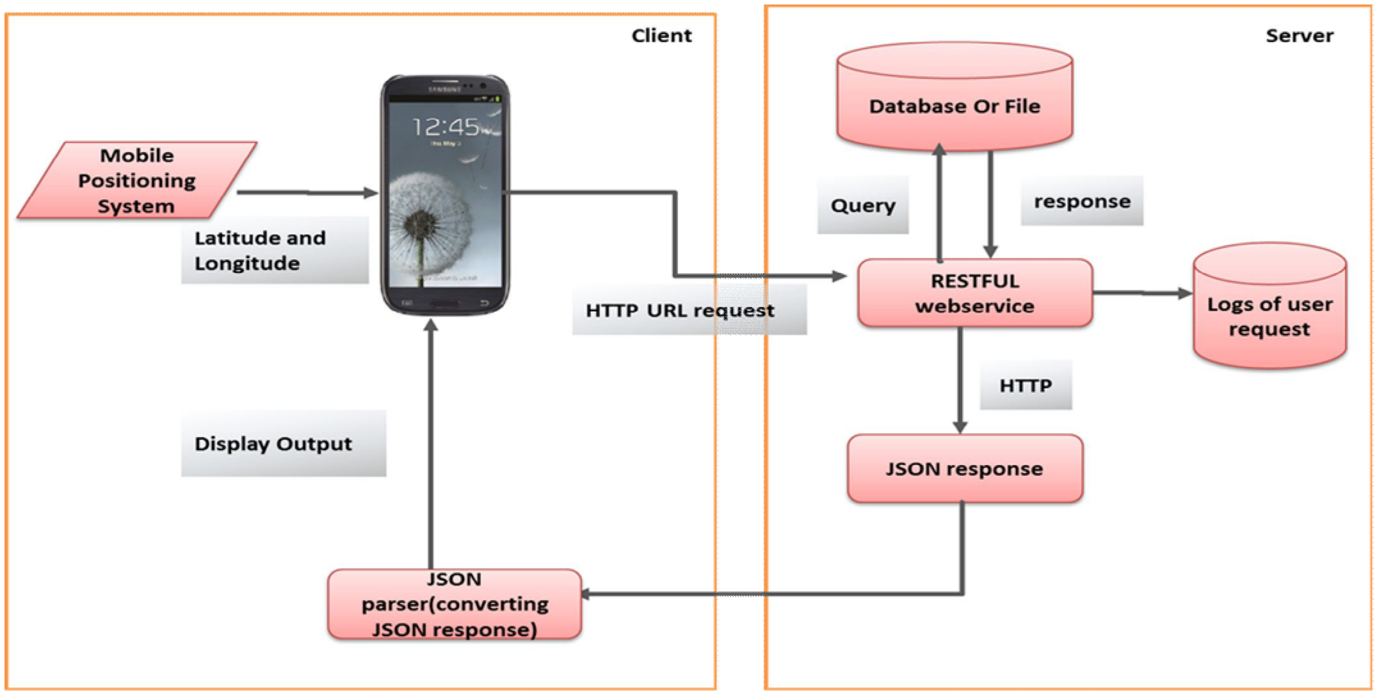


Fig. 1. Proposed Architecture of Application

IV. IMPLEMENTATION OF PROPOSED ARCHITECTURE

MOSDAC weather Application provides weather conditions for next 3 hours. It also gives 24, 48 and 72 hours forecast for parameters such as temperature, humidity, rain, wind speed, and sky conditions. Topographical information (land, mixed, and ocean) is provided at the user's point of interest. This application also provides several weather alerts for cyclone, fog, smoke, fire, heavy rain. Below, we describe the prerequisites and the features of our application.

A. Prerequisites

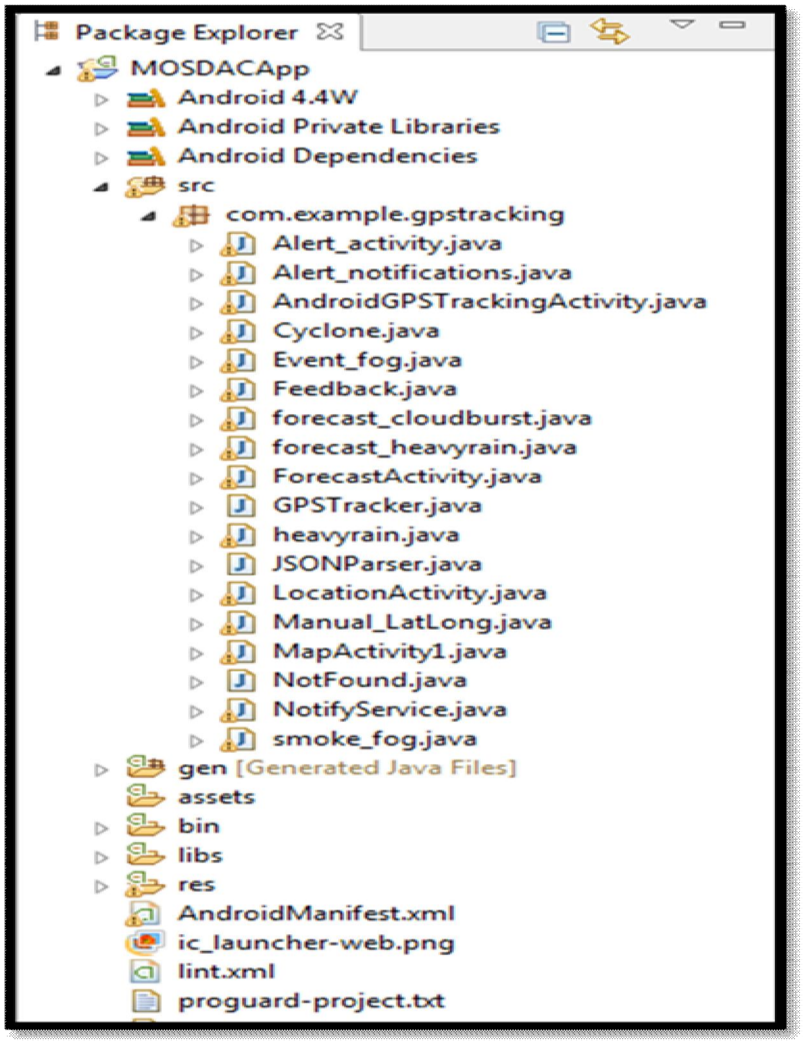
1) The application is first implemented on Android Development Toolkit (ADT) 4.4 (Kitkat) version. This application can also run on higher versions of android.
2) Internet connection is required to download the MOSDAC Weather Application.

GPS or Network must be available in handset. It is useful to determine the current location of user.

B. Programming Details

Android programming project file structure is generated using the Eclipse project. The project structure is shown in Figure 2. The android manifest file presents essential information about the application to the Android system. The system must have this information before it can run any of the application's code. This project structure consists following two elements in manifest file.

1) Activity (activities) **-** <Activity> tag usually represents a display screen. For example, displaying weather information is one Activity and displaying the weather alert information is another Activity.
2) Service (service functions) **-** use <Service> tag is used when Android application does not require the display on the android device, but it should be a long run in the background. Such as: Notification service.



```
Package Explorer
MOSDACApp
  Android 4.4W
  Android Private Libraries
  Android Dependencies
  src
    com.example.gpstracking
      Alert_activity.java
      Alert_notifications.java
      AndroidGPSTrackingActivity.java
      Cyclone.java
      Event_fog.java
      Feedback.java
      forecast_cloudburst.java
      forecast_heavyrain.java
      ForecastActivity.java
      GPSTracker.java
      heavyrain.java
      JSONParser.java
      LocationActivity.java
      Manual_LatLong.java
      MapActivity1.java
      NotFound.java
      NotifyService.java
      smoke_fog.java
  gen [Generated Java Files]
  assets
  bin
  libs
  res
  AndroidManifest.xml
  ic_launcher-web.png
  lint.xml
  proguard-project.txt
```

Fig. 2. PROJECT FILE STRUCTURE TABLE

C. Internal architecture of MOSDAC Weather App

The internal architecture of MOSDAC weather application is illustrated in Fig. 3. There are main two components on the client side: activity and service. In the com. example. gps tracking package, we have different activities such as GPS Tracking, Location, and Forecast etc. For the notification purpose, we have one service called Notify service. In the GPS Tracking activity current location is obtained by using location Manager class which is already available in android api. This location information is passed to the server. On the server side, we have used Servlet. In com. example. weather Details package, we have different servlets such as Weather Details, Date forecast details, Get forecast Details, Heavy Rain etc. Client location information is passed to Weather Details servlet using RESTful web service which is a URI based web service. Here, com. example. information package is used to get and set the attributes which are used in Servlet. RESTful web service processed the request and generates the response by querying the database. This response is converted into JSON data format by using JSON parser and weather information is shown on the Andoid phone.

D. Features of the application

1) Whenever the application is launched, user will see three options which are presented in Fig 4. There is an option named 'Google Map' that enables user to take a look at his current location. It is demonstrated in Fig 5.

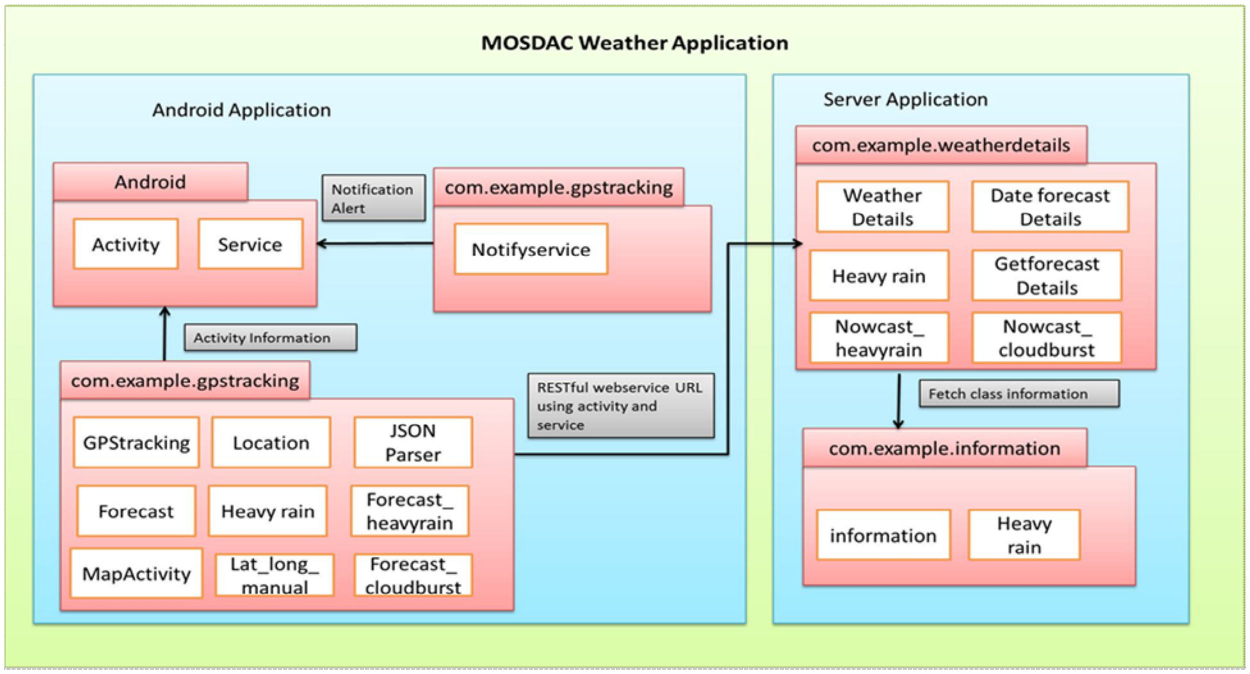


Fig. 3. Internal Architecture of MOSDAC Weather Application

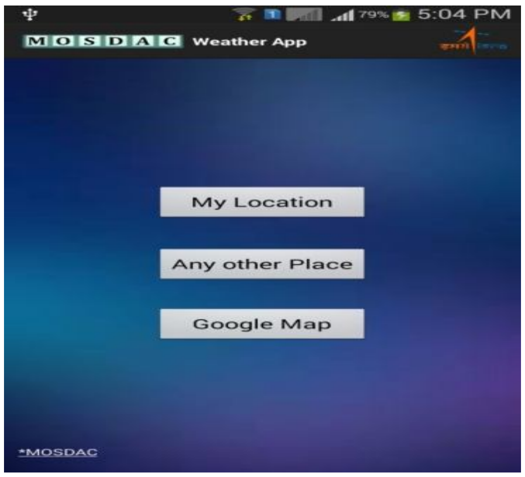


Fig. 4. Main menu of the applicaion

If user wants to get the forecast details from location on the Google map then this activity is called. GPSTracking activity gets the latitude and longitude value for the particular location from the Google maps and redirects the latitude and longitude to the Location activity.

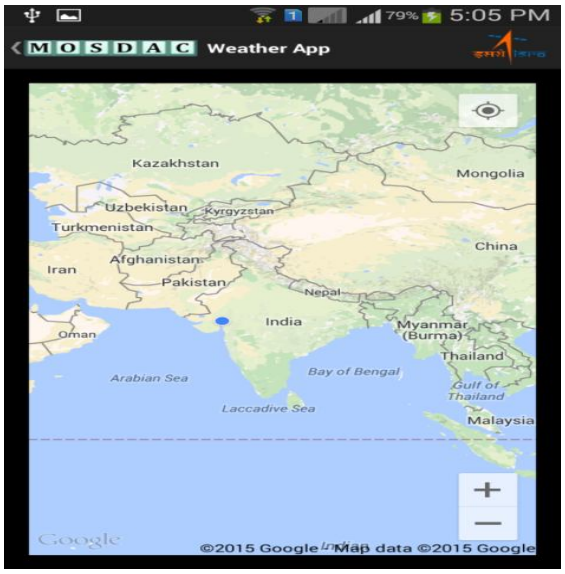
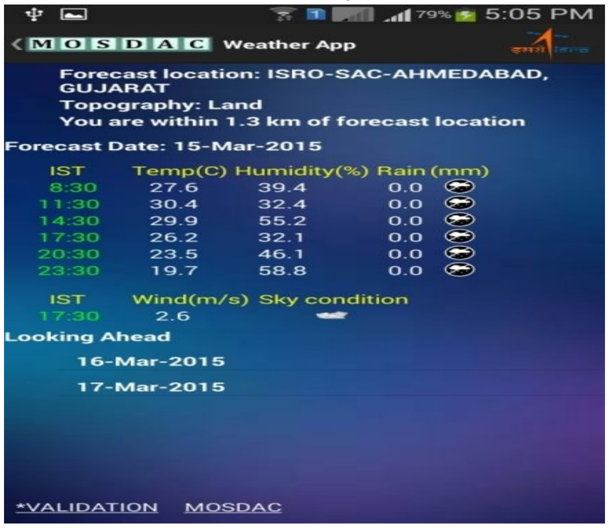


Fig. 5. Weather info using Google map

2) The Location activity is responsible for connecting the web service. This webservice allows user to query the database and obtain the results. If the main thread which controls elements of the User Interface is blocked by waiting for incoming connections, the interface will appear to freeze. if main thread is blocked for long enough, it will cause an error message dialog to be displayed to the user allowing him/her to force close the application. This is not the experience expected and so an additional thread is created to handle connections between Android client and MOSDAC server. Android provides a class called AsyncTask to make it easier for the developer to handle threading. It makes it possible to publish updates from the background thread to the UI thread, but also contains methods for on Pre Execute () and on Post Execute () which are first invoked on the UI thread before and after the thread works on background tasks. This makes it possible to add a loading dialog to show onPreExecute (), and to dismiss when onPostExecute () is called. By handling the workload and waiting for connections in this way it is possible to show a user friendly dialog and avoid possible freezing problems. An inner class for each section which requires a HTTP connection has been created inside the Location activity which is responsible for communicating with the web service and parsing the JSON response. Parameters such as latitude and longitude are sent to the servlet using HTTP POST. The weather information is shown in Fig 6. Weather Information for the next day is shown in Fig. 7.



Forecast location: ISRO-SAC-AHMEDABAD, GUJARAT
Topography: Land
You are within 1.3 km of forecast location
Forecast Date: 15-Mar-2015

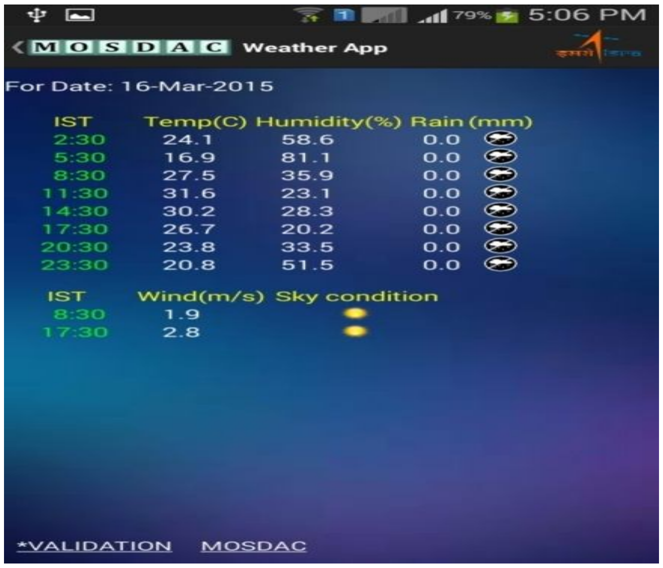
IST	Temp(C)	Humidity(%)	Rain (mm)
8:30	27.6	39.4	0.0
11:30	30.4	32.4	0.0
14:30	29.9	55.2	0.0
17:30	26.2	32.1	0.0
20:30	23.5	46.1	0.0
23:30	19.7	58.8	0.0

IST	Wind(m/s)	Sky condition
17:30	2.6	

Looking Ahead
16-Mar-2015
17-Mar-2015

*VALIDATION MOSDAC

Fig. 6. Weather information for requested station


5:06 PM

MOSDAC Weather App

For Date: 16-Mar-2015

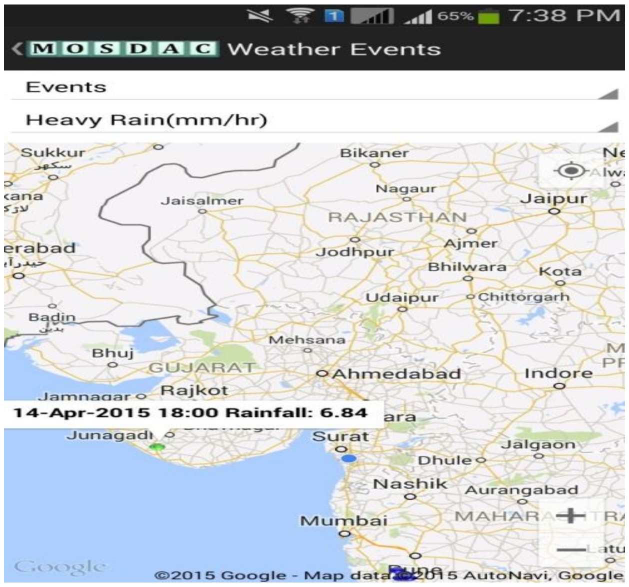
IST	Temp(C)	Humidity(%)	Rain (mm)
2:30	24.1	58.6	0.0
5:30	16.9	81.1	0.0
8:30	27.5	35.9	0.0
11:30	31.6	23.1	0.0
14:30	30.2	28.3	0.0
17:30	26.7	20.2	0.0
20:30	23.8	33.5	0.0
23:30	20.8	51.5	0.0

IST	Wind(m/s)	Sky condition
8:30	1.9	●
17:30	2.8	●

*VALIDATION MOSDAC

Fig. 7. Weather information for next day

For disseminating the information related to heavy rainfall, an activity named heavy_rain is created. The heavy_rain activity connects component A (running on client side) and component B (running on server side) But the problem is with the size of text file. In rainy days, the size of heavyrain.txt file goes up to approximately 6 MB. The file contains details of 3, 63, 272 points or rainfall locations. So, it is not possible to parse the huge JSON response by using RESTful web service and show it on Google map. Therefore, we have applied the topographic algorithm on each and every location of the file to check whether the point is on land, ocean or mixed. If the location is in the ocean then that particular point is removed from the file. After applying the algorithm, the file size gets reduce to 5 KB. This algorithm is useful to minimize the response time. Moreover, it also helps to overcome the memory problem. The reduced file is then converted into JSON response and sent to the Android client. The heavy rainfall information of Junagadh district in Gujarat is shown in Fig. 8.


7:38 PM

MOSDAC Weather Events

Events

Heavy Rain(mm/hr)

14-Apr-2015 18:00 Rainfall: 6.84

Fig. 8. Heavy rain information Junagadh district in Gujarat

For disseminating the Nowcast information, two activities are created named forecast_heavyrain and forecast_cloudburst. Forecast_heavyrain activity is responsible for connecting to the web service to allow user to query the database and obtain the results. Here, PostgreSQL database is used for Querying the results or storing the data. Database can store the data. This activity is showing information related to heavy rainfall and cloudburst over Uttarakhand and Himachal Pradesh. The Nowcast information related to heavy rain and cloudburst is shown in Fig. 9 and Fig. 10.

Dissemination of smoke/fire and fog alerts has been tested over test-setup. This test-setup includes local GIS database and Android emulator. This development can easily be operationalised with same tuning and server related configurations after complete chain is ready.

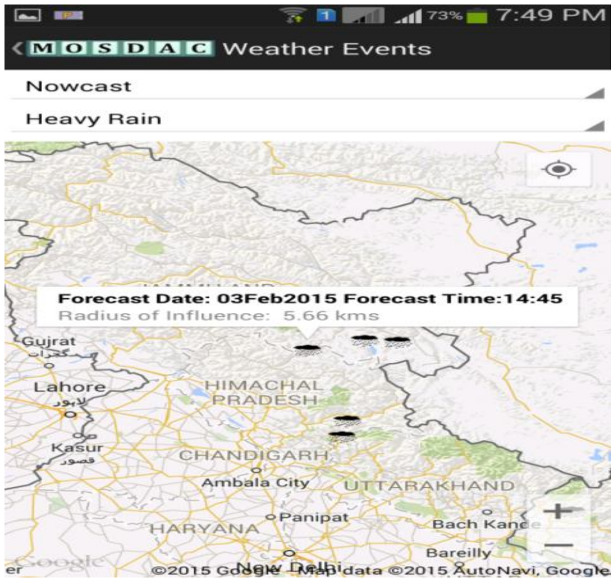


Fig. 9. Nowcast information for heavy rain

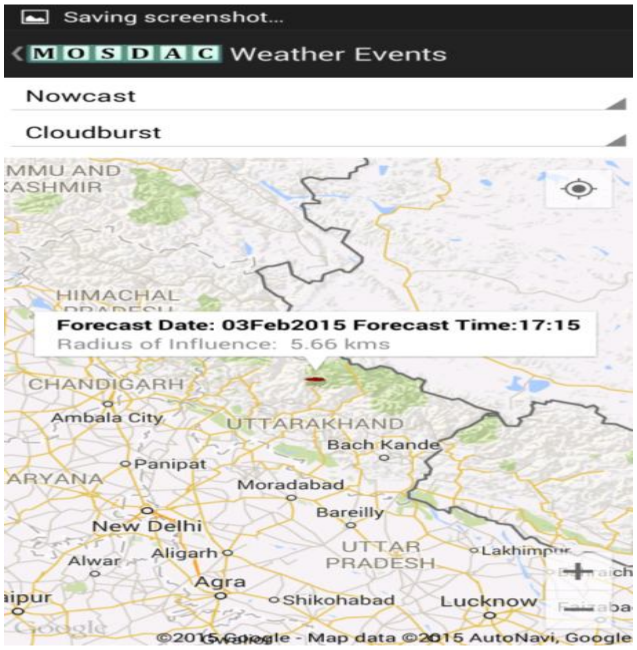


Fig. 10. Nowcast information for cloudburst over uttarakhnad and Himachal Pradesh

V. RESULTS

A. *Results for Battery consumption*

We have used substitution technique for minimizing the battery consumption. Substitution technique uses low energy location-sensing mechanisms as compared to GPS. Low energy location-sensing mechanisms are WiFi, and cell-id positioning. PowerTutor [11] is used to determine the battery consumption in MOSDAC Weather Application and other such weather applications. PowerTutor allows user to monitor the power consumption of the phone or monitor the power consumption of applications running in the mobile phone. Table I shows battery consumption in different scenario: (a) when GPS is on in application, (b) when notifications are activated in applications, and (c) when notification and GPS are not activated in applications.

TABLE I. BATTERY CONSUMPTION IN DIFFERENT SCENARIO FOR DIFFERENT WEATHER APPLICATIONS

Application Name	Battery consumption When GPS is activated	Battery consumption When Notifications are activated	Battery consumption When GPS and Notifications are not active
Accuweather	43.0 J	87.8 J	11.5 J
Weather Channel	79.1 J	67.8 J	11.7 J
Weather	37.2 J	34.7 J	2.7 J
MOSDAC Weather App	32.8 J	13.0 J	2.0 J

B. Results for response time

The core components of the MOSDAC weather application were described in the previous section. These components are tested to verify their desired performance. The performances of these components are tested using the response time of the application for different activities to provide a good reference framework. Response time of application is shown in Table II.

TABLE: RESPONSE TIME OF APPLICATION

Activity	Response Time Test 1(ms)	Response Time Test 2(ms)	Response Time Test 3(ms)
Location Activity	23627	23300	22582
Forecast Activity	12828	12752	12471
forecast_heavyrain	997	995	1047
forecast_cloudburst	1034	985	970
Heavy_rain	987	965	957

VI. DISCUSSION OF RESULTS

A. Analysis of the results of battery consumption

By comparing MOSDAC weather application with other such applications, we can conclude that our application uses less battery than other applications in different scenarios.

In the first experiment scenario, GPS is enabled for all the applications. MOSDAC Weather Application consumed less battery than any other applications. There may be the following reasons for more battery consumption in other applications: other applications are using heavy widgets in order to make the application attractive. This can drastically reduce battery life. Other applications automatically detect user’s location at a fixed interval. But it is better not to detect user’s location automatically through GPS. It is better to detect user’s location only when user is requesting for weather details.

In the second experiment scenario, alerts for the notifications are enabled for all the applications. There may be the following reasons for more battery consumption in other applications: we have created our own notification service for pushing the notifications. Notification interval set to half an hour. Therefore, Battery consumption is less in our application.

In the third experiment scenario, all the applications use lesser battery compared to previous scenarios. Other applications consume more battery compare to our application because these applications use live wallpapers and heavy widgets.

B. Analysis of the results of response time

Response time for location activity is higher than any other activity in MOSDAC weather application. The reasons for the same are: 1) Getting coordinates for the location in this activity 2) Connection with the database, implement Haversine formula in the query and fetch the nearest location for the weather details. 3) Topographic algorithm is also implemented for this activity 4) Getting the forecast dates from the database by using query.

VII. CONCLUSION

In this paper, we presented architecture and implementation details of location based MOSDAC weather application. Our application provides current weather conditions and detailed 24, 48 and 72 hour forecast of heavy rainfall, fog, smoke and fire at every half an hour. Moreover, the application also gives Nowcast (Half hourly forecast valid for 6 hours) information related to heavy rainfall and cloudburst over Himachal Pradesh and Uttrakhand - states of India. This application will be helpful to have weather information during any emergency situations. We have also compared the battery consumption of MOSDAC Application with other such applications. Our future plan is to incorporate fog, smoke and fire alerts. Furthermore, the features of MOSDAC weather application can be extended by using mobile usage patterns and machine learning algorithms for minimizing battery life. This application has been given to the Disaster Management Support Programme (DMS) of ISRO and has been made operational on http://www.mosdac.gov.in.

REFERENCES

[1] Meteorological and Oceanographic Satellite Data Archival Centre(MOSDAC) Web portal and http://mosdac.gov.in/login.jsp. Accessed 22nd April 2015.

[2] Bhagyasri G. Patel, Vipul K. Dabhi, Utkarsh Tyagi, Pushpalata B. Shah, “A Survey on Location Based Application Development on Android platform” is presented in the IEEE International Conference on Advances in Computer Engineering & Applications (ICACEA’15) at Ghaziabad.

[3] Openweatermap Website.http://www.openweathermap.com/forecast. Accessed 22nd April 2015.

[4] Weatherbug website.http://weather.weatherbug.com/mobile.html. Accessed 22nd April 2015.

[5] Android, “http://en.wikipedia.org/wiki/Android_(operating_system)

[6] B. Rajitha, “Performance of Web Services on Smart Phone Platforms”, IOSR Journal of Computer Engineering (IOSR-JCE), Volume 14, Issue 5 (Sep. - Oct. 2013).

[7] R. Aher, N.Gupta, A. Khairnar, K. Thanki, A.Gaidhani ,"Bluetooth Based Notification System for Blind People Using JSON Technology",International Conference of Advance Research and Innovation ,2014.

[8] R. Anacleto, N. Luz, and L. Figueiredo. "Personalized sightseeing tours support using mobile devices" *Human-computer interaction*. Springer Berlin Heidelberg, 2010. 301-304.

[9] A. Patel, "An efficient location information management system (LIMS) for smartphone applications", San Diego State University, 2012.

[10] Haversine Formula http://andrew.hedges.name/experiments/haversine Accessed 6th March 2015.

[11] Power Tutor Website. http://ziyang.eecs.umich.edu/projects/powertutor/. Accessed 4th April 2015.