SKILL DEVELOPMENT FOR MOBILE GAME & APPLICATION

Report on

Weather App BD

Weather application Bangladeshs a iOS mobile application that shows the realtime and accurate weather information through our mobile phone.

Project Developed By–

Md Zahidul Islam Mazumder

S.K. Al–Abid

Arika Afrin Boshra

Abstract

Nowadays we face a huge problem that knowing real weather status instantly in such place we need to know. It is a complex and often challenging skill that involves observing and processing vast amounts of data. Weather systems can range from small, short lived thunderstorms only a few miles in diameter that last a couple of hours to large scale rain and wind up to a thousand miles in a diameter, and lasting for days. So most of the times we cannot get the real weather forecast and face to lot of troubles. We have another problem in weather forecasting.

As there are 23 weather stations in Bangladesh, they all need to give updates on weather situation in those areas to headquarter in Meteorology Department in proper way. But nowadays they use telephone calls for giving updates to department in every other three hours. This is called as a synopsis record. This method is inappropriate, because those data are not secured. So this method need to get update to a proper way. The system will provide them with reports, analyzed data according to their request.

And also we use crowdsourcing application. The crowdsourcing application for weather forecasting is basically more important and useful for people to make decisions depending on the weather condition of an area. The application also builds a trustworthy relationship with them in order to gather accurate data, to maximize the utilization of the application, and to cover the whole island for gaining hundred percent coverage of information.

Table of Content

Page

Table of Contents

List of Figures/ Tables

Page

Chapter 1

Introduction

1.1 Introduction

In this chapter, we hope to provide a brief depiction on the background and motivation about our project, point out the aim and objectives and to present the proposed solution through our project.

1.2 Background and Motivation

It is very important to get educated on the current weather situation of a particular location as preferred since it affects the day to day life of everyone. It is more effective if we can get quickly updated on current weather status of a required location, as it make easy to handle not only our activities, but also our livelihoods too.

A huge problem that we are facing nowadays is inability of knowing real weather status in such places we need to know. So if we need to know current situation on a certain place, it is better to ask from a person who is in that area recently or currently. He is a better source than any prevailing weather information.

We have another problem in weather forecasting as there are weather stations which are island wide need to give updates to the headquarters in a proper way and in a timely manner. Though it is a complex and significant process, telephone calls are currently being used to retrieve updates from the stations in Meteorology Department of Bangladesh. Therefore, weather forecasting is ultimately a three steps process which includes observing, forecasting and communicating. This implies that this method needs to be upgraded into an appropriate approach.

1.3 Aim and Objectives

1.3.1 Aim

The aim of this project is to make a software application that can be downloaded and used in an iOS device to get to know about real- time weather updates in a particular place that we need to know and inform others about the current weather status in our location, and also to make an application for different weather stations of Meteorology Department to submit their weather synopsis to the system for every three hours either using the computer at the station or using the specified mobile application only for them.

1.3.2 Objectives

- Study about weather forecasting applications and systems.
- Study about technologies like web application, iOS app, which can overcome problems.
- Study about mapping.
- Design and develop a system which can properly get updates of the current weather status.
- Evaluate proposed solutions.
- Hand over successful system to the company.

1.4 Proposed Solution

Our solution is to implement weather forecasting system of Meteorological Department and implement a crowd based iOS application where users can submit the weather condition of their current location. The system of Meteorological Department is basically for saturating their current needs on updating weather synopsis accurately and in a timely manner. This synopsis-record contains a code that implies the data of temperature, tendency, dew-point, humidity, rainfall, cloud, wind type, visibility.

Functions of the weather forecasting system of Meteorological Department:

- It allows users from different weather stations to submit synopsis to the system for every three hours either using the computer at the station or using the specified mobile application only for them.
- The system provides weather status of a particular area according to the submitted synopsis using weather maps and weather reports.
- The system analyzes the current weather condition of a particular area with the track records of past thirty years, and generates reports based on those analyzed data.

Users of the weather synopsis submission system:

- They have the basic knowledge on how to submit the synopsis, and they have fully experienced on the encoding techniques used in synopsis.
- The users submit a weather report from a particular weather station every three hours.
- They can submit the synopsis from their mobile phone by using a mobile application developed for them only after verifying their location.

If users want to know of weather condition of any other location, they can search for the specific location. Application will provide all the weather condition of that specific location like is it raining there or not, wind speed, temperature and etc. If needed application will provide a map of Bangladesh visualizing temperature, whether it is raining or not and etc. But users will not be willing to give weather condition of

their current locations every time. If this problem keep happening then we can't generate an accurate weather forecast based on crowed.

To avoid that we propose some kind of a restriction: If a user want to know weather condition of a specific location first he or she has to submit the weather condition of his or her current location. The user does not have to select him or her location as input. Because based on their GPS position their location will be automatically submitted. All the user has to do is select weather it is raining or not. But we cannot rely 100% on user submission as users tend to submit false submissions too. At this point we have to compare user submissions with a reliable source too. So before providing a requested weather update to user we decided to compare all the user submissions within one hour with an online weather forecast provider. Then the result will be much reliable one.

Functions of the crowdsource mobile weather forecasting application:

- The application covers the all areas within the island to facilitate all the users of it within the country.
- The mobile application provides current weather status of an area or a location according to the need of the user.
- It enables users to update the system of the mobile application with the latest weather condition of a particular area.

Users of the crowdsourcing mobile application:

- These users are the general publics who have this application installed mobile phones.
- They can get the current weather condition of a place or an area according to their necessity.
- The users also can update the system by providing the latest status of the place where he/she is at that moment.

Another bump we did hit while going down our path of achieving this is Department of Meteorology of Bangladesh doesn't provide an API for retrieve weather conditions for the third party applications. Meteorology Department doesn't provide hourly base update of weather conditions therefore, we use Yahoo weather API which gives us hourly- based weather update for all the locations in Bangladesh.

All the user inputs within one hour will be stored for generating a weather forecast, and also all the weather parameters from Yahoo weather server will be stored hourly basis.

1.5 Summary

This chapter, Introduction has provided a brief description of our project including several captions including background, motivation, aim and objectives, proposed solution. In the next chapter, we will discuss about prevailing similar applications and point out the specialty, performance and benefits of our proposed application while comparing to them.

Chapter2

2.0 Compare with similar works

2.1 Introduction

There is large number of weather applications recently used in the world such as Bangladesh Weather, Yahoo! Weather, AccuWeather, Weather Live, Weather Bug and so on. [6] They provide their users with weather information, and this chapter will supply a comparison between our approach and them.

2.2 Similar applications for IOS Weather

- AccuWeather

  This free app is designed for iPhone users. It includes something called the “iPhone weather station,” which allows you to receive alerts for warnings about severe weather, and it can even forecast out to 15 days, as opposed to the traditional 10 day limit. To top it all off, it integrates seamlessly with iOS calendar so user can schedule important events around weather.

  Though this AccuWeather application is very useful for iPhone users, the majority in Bangladesh is iOS smart phone users. Therefore Our Weather-Now application is more efficient to be utilized in Bangladesh.

- Weather Live

  Weather Live is also for the iOS system. Focusing on aesthetically pleasing design, this app focuses on being minimal and easy for its users. Allowing a layout that fills the entire screen, you can easily view your current weather conditions, represented by the changing background. It even shows you the current temperature on your main screen, adjustable by visiting the settings. If you’re focused on beauty and design, this could be a great weather app for you.

Though this Weather Live application also is very beneficial for iPhone users, the majority in Bangladesh is iOS smart phone users. Therefore Our Weather-Now application is more resourceful to be employed in Bangladesh.

2.3 Summary

Today, there are similar applications that complete related tasks, using the same approach as our application. Therefore, we have detailed down those applications in this chapter including a comparison and list of special points how our approach differs from them, and the next chapter will explain the technology that we are going to use with reasoning.

Chapter 3

Technology Adapted

3.1 Introduction

This chapter provides specifics and details about the technology that we have adapted to solve the problem through implementing our proposed solution. Furthermore, it will point the reasons and the ways that these techniques and technology are appropriate for the proposed solution.

3.2 Technology for the Solution

We hope to use following frameworks and techniques through the implementation of our proposed solution.

- Node.js
- MongoDB
- XCode
- GPS Technology
- Google Maps

3.3 Node.js

In our weather forecasting application for Meteorology Department highly rely on multi-user interaction which is occurred concurrently for every three hours. Thus, IOS Weather mobile application might have users who update and retrieve same data at the same time which should be arisen with concurrency issues. Therefore, it is efficient if we can eliminate those issues and provide an efficient performance throughout the processes, and to achieve this task, we have selected Node.js. It is an open source, cross-platform run-time environment which is used for developing server-side and networking applications. Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution, and its nature of high scalability allows it to provide service for much larger number of requests than other servers. [9]

3.4 MongoDB

The weather forecasting system frequently involves with information updates and retrievals. This occurs due to the frequent updates by weather station users. Thus the users of IOS Weather application concurrently and recently update and request for weather information. These procedures involve a huge amount of data that flow from and to the database. Therefore, we should ensure that our system will continue its performance by accomplishing these tasks, and we have chosen MongoDB as our database for that purpose. It is an open source document-oriented database. MongoDB can represent rich and hierarchical data structure, it is easily scalable and able to give high performance. [10]

3.5 XCode

The weather forecasting system has a mobile application, and for its implementation, we have chosen XCode. It is an open source and Linux-based operating system for mobile devices. XCode supplies a flexible build system, fast and feature-rich emulator, unified environment where we can develop the application.

3.6 GPS Technology

The crowd-sourcing application of our system requires to verify the location of the user before he/she updates the current weather status of the location where he/she is at that moment. It indicates the need of detecting the location of the user, so we have to use GPS technology. Global Positioning System (GPS) is a network of orbiting satellites that send precise details of their position in the space back to earth. The signals are obtained by GPS receivers, and are used to calculate the exact position and time of a particular location. [12]

3.7 Google Maps

Our solution needs geographical maps for demonstrating the maps to indicate weather information graphically for the users for their better clarification. We use Google maps to accomplish this task. By using the Google Maps API which is provided by Google Maps, it is possible to embed Google Maps site into our application. [13]

3.8 Summary

The technology adapted for implement the system has been described in this chapter, and in the next chapter, we will provide how we adopted this technology for the proposed solution.

Chapter 4

Our Approach

4.1 Introduction

This chapter is focused on how we are going to adopt the technology to implement the solution for our weather forecasting system. There are several technologies that we are using in the solution.

4.2 Approaches to the Solution

The contribution of all the members has been ensured through diving a specific part for each member while all of us are aware about the overall progress of the system. The attention for the similar applications to the mobile application and deep observation of them should be done to achieve the expected performance level of our solution. It is required to establish a user-friendly environment in the weather forecasting system for making users easier to interact with it. This needed to be achieved via understanding the reliability and the perspective of public user.
Especially, we had several field visits to the Meteorological Department with the intention of gain more information about the procedure that is launched while updating a weather synopsis. These reports are updated for every three hours, and they contain weather parameters including tendency, temperature, rainfall, wind type.
Thus, there is a special encoding system which is a universal standard that is used in updating figures about the parameters. Therefore, we should be aware about these techniques.
After gathering all the requirements and the details that are essential for the development, we can design and develop the system by fulfilling all the requirements of all the users. Here, we employ the technologies such as GPS technology, Node.js, XCode and Google Maps for the implementation stage of the applications.
After the designing stage, we have to engage in testing for identifying the errors, drawbacks and mistakes in the developing system.
Then, we should stimulate the environment of the weather station users to get them familiar with the new system, and also to ensure that the application is friendly and fast enough for the purpose. When considering the mobile application, we have to

make it much easier to be used by a user, and to ratify its accomplishment in attaining its goals.

4.3 Summary

The technologies that we are used to implement our system have been designated in this chapter as the way of adopting them for the solution for accomplishing its task, and the next chapter is focused on the analysis and design perspective in order to provide details about the interaction among the modules within the system.

Chapter 5

Analysis and Design

5.1 Introduction

This chapter comprises the particulars about the design of our solution for the weather forecasting system. Therefore, this includes several diagrams to depict the top level design of the proposed solution, and the interaction between each and every components of it.

5.2 Top Level Design of the System

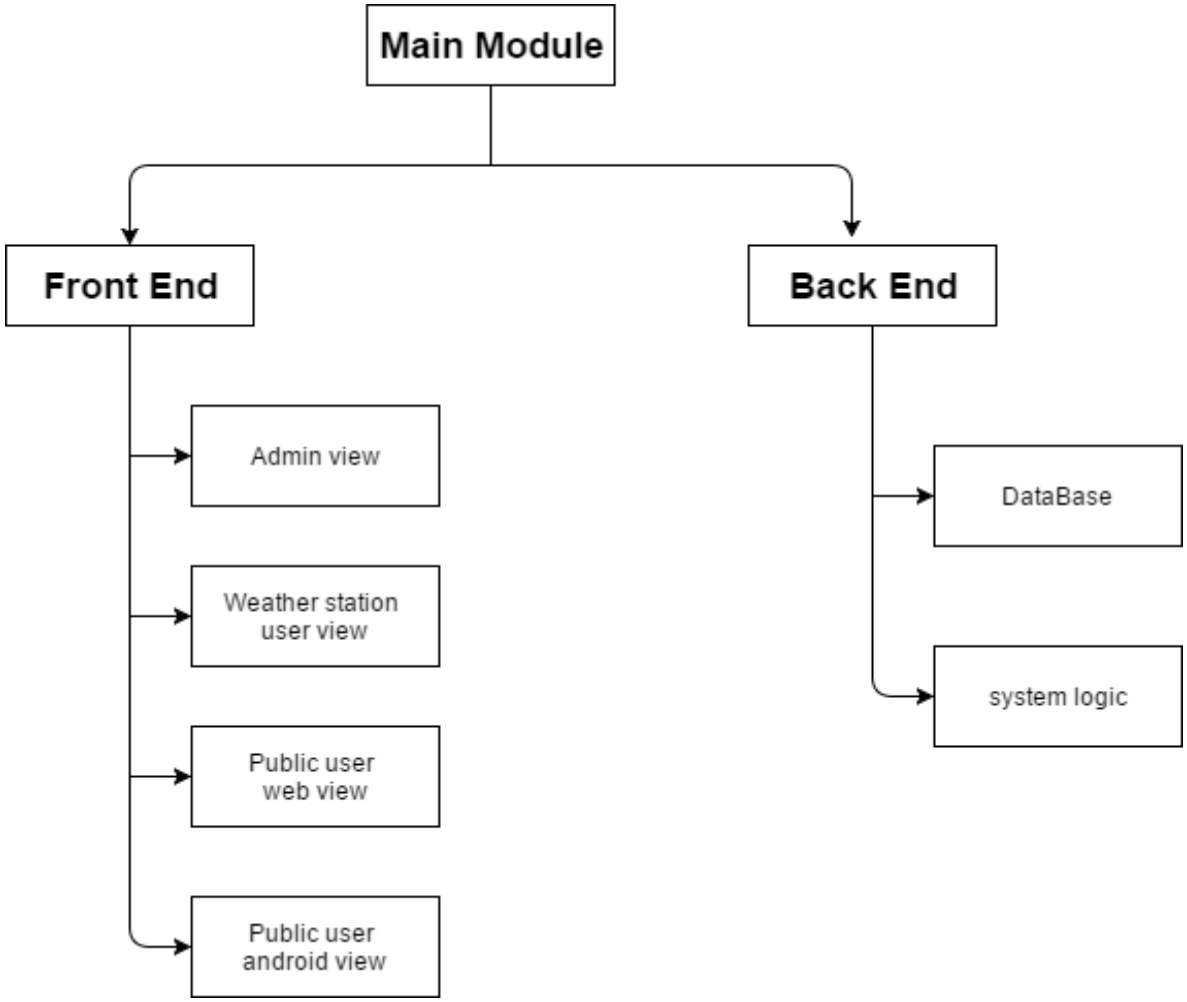

Figure 5.1: Top Level View

This top level view diagram indicates the basic modules of our solution, including user views, database and system logic. Admin View is for the administrator of the Meteorological Department who grants the permission to the user and reviews the

submitted weather reports by all the weather station users. Weather station users are able to update the weather synopsis. App view of the public users allow retrieving required weather information if and only if they have submitted the current location's weather status.

5.3 System Diagram

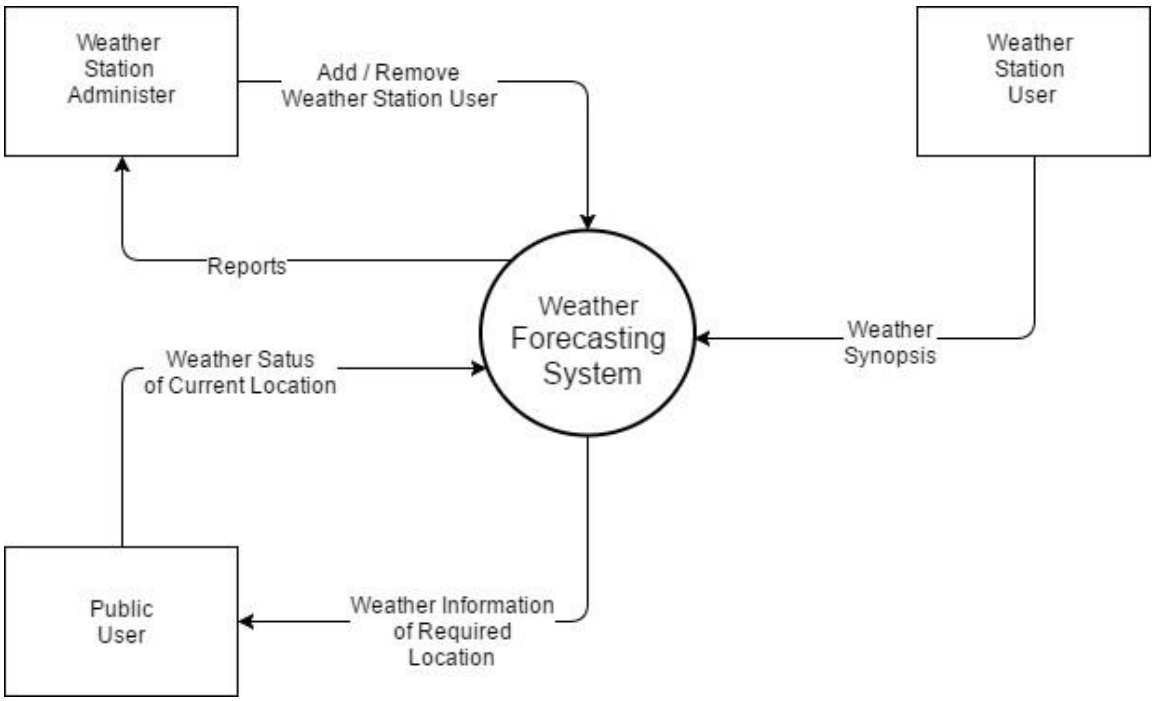

Figure 5.2: System Diagram

The System diagram provides the overall indication of the boundaries and the scope of our system, and it depicts main data flows of the system.

5.4 Use Case Diagrams

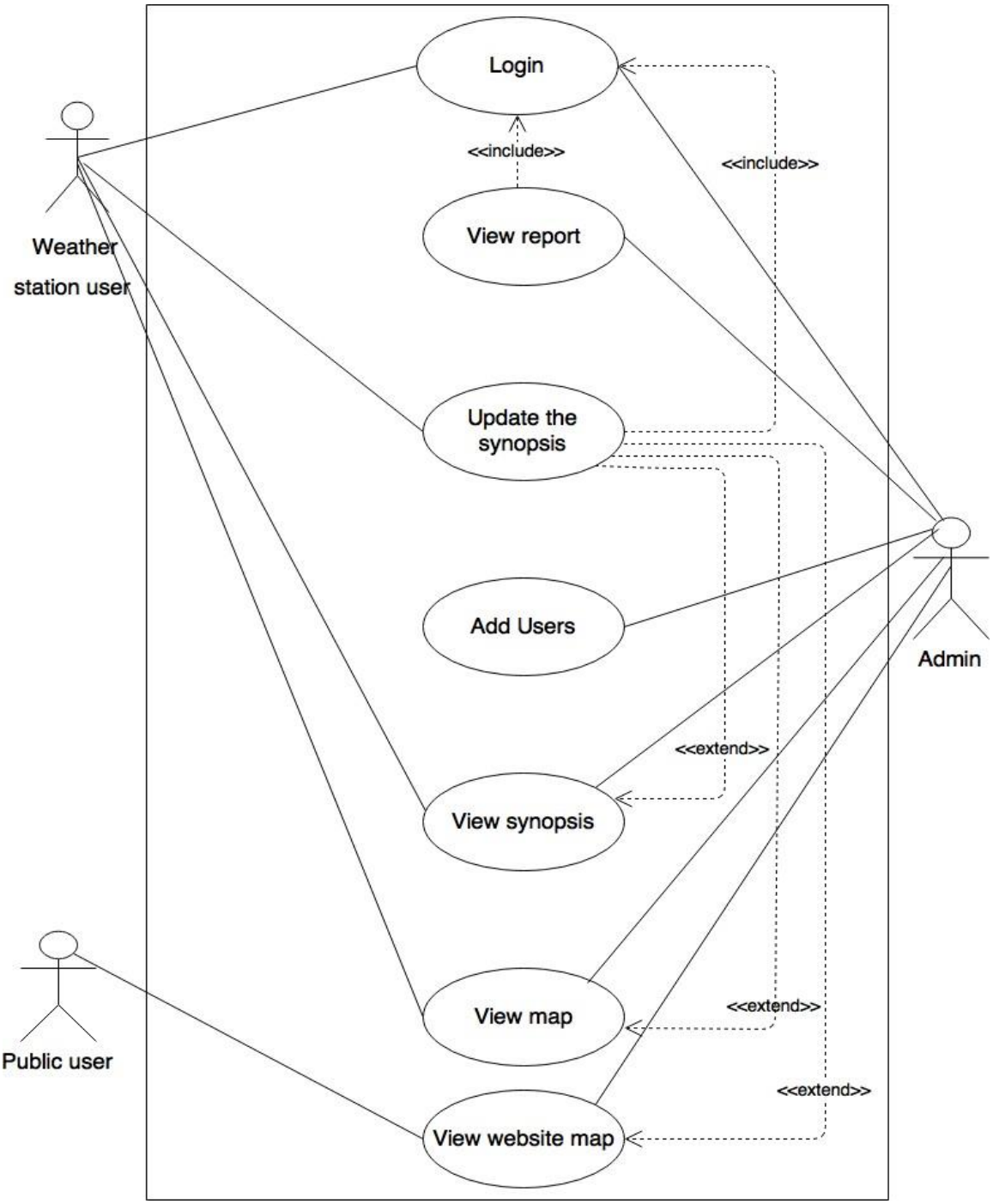

Figure 5.3: Use Case for Weather Stations

Figure 5.3 shows the interaction between its users that are admin and the weather station user, and the application in the viewpoint of each user of the weather station system. The following diagram shows this interaction according to the system and the public user.

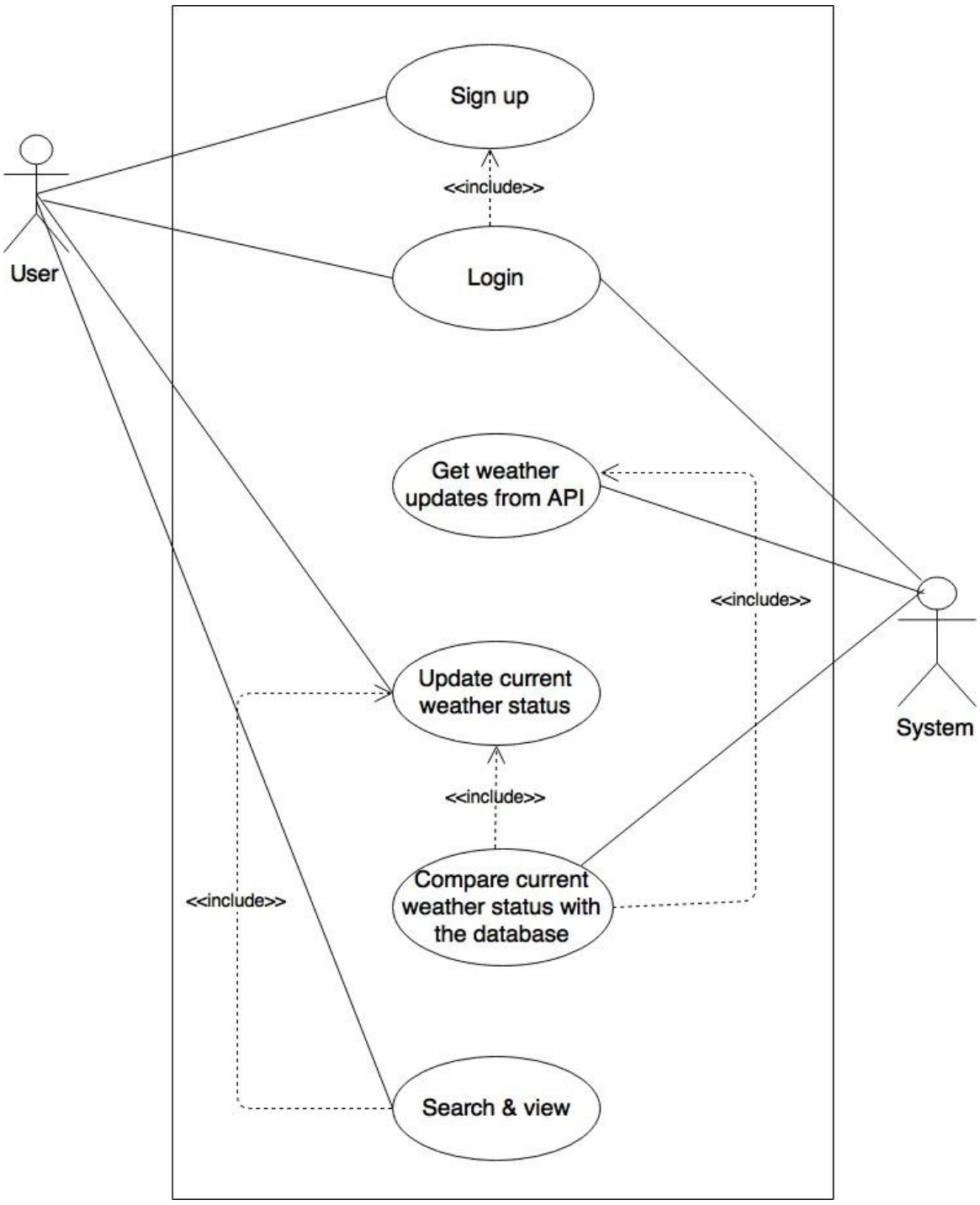

Figure 5.4: Use Case for Crowdsourcing Application

5.5 Activity Diagrams

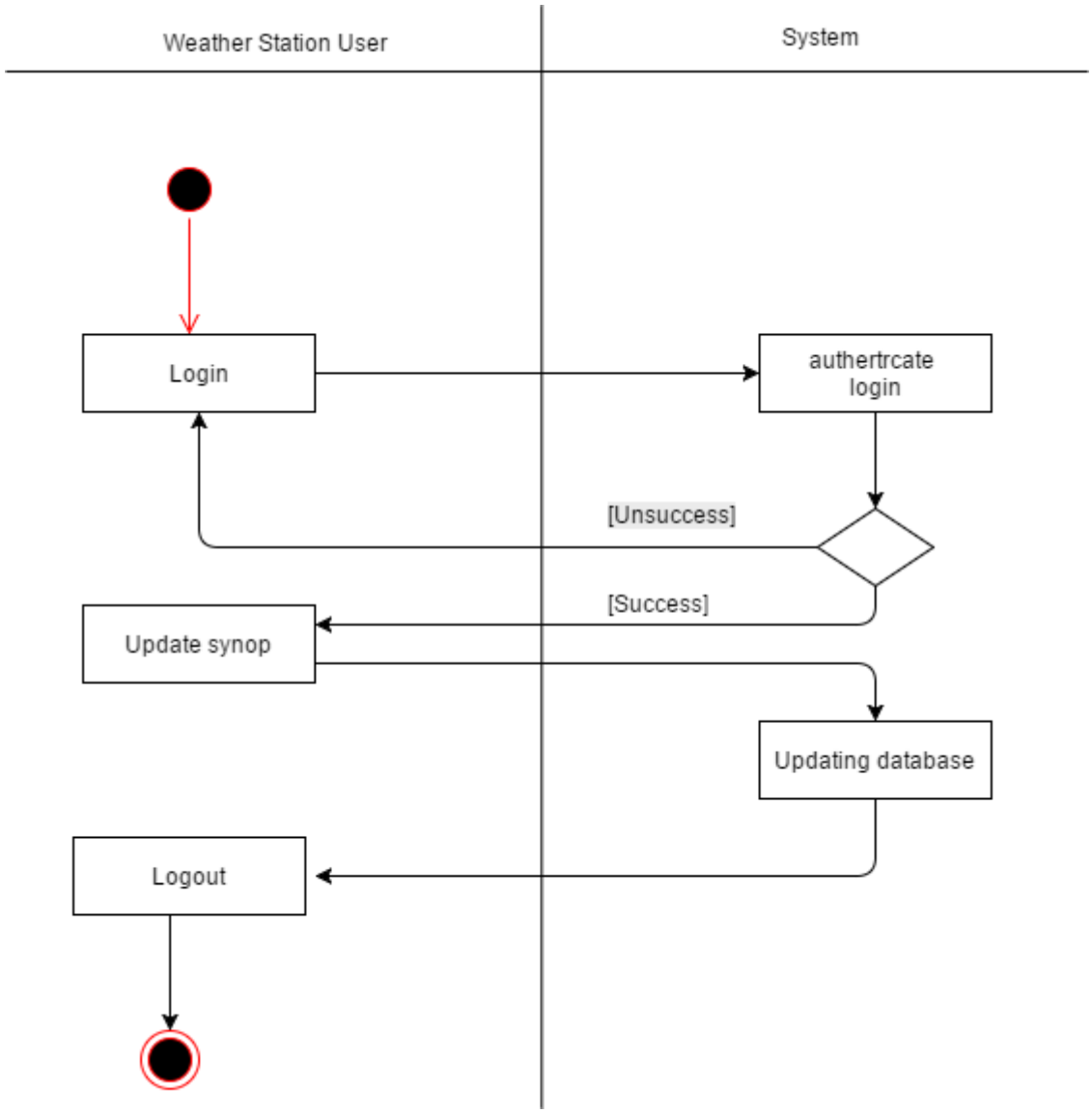

Figure 5.5: Activity Diagram for Weather Station User

The above activity diagram depicts the flow of the process of updating the synopsis by a particular weather station user. This comprises a login mechanism initially which is followed by the updating the weather synopsis. A user must log in to the system for updating to confirm the accuracy and reliability of information.

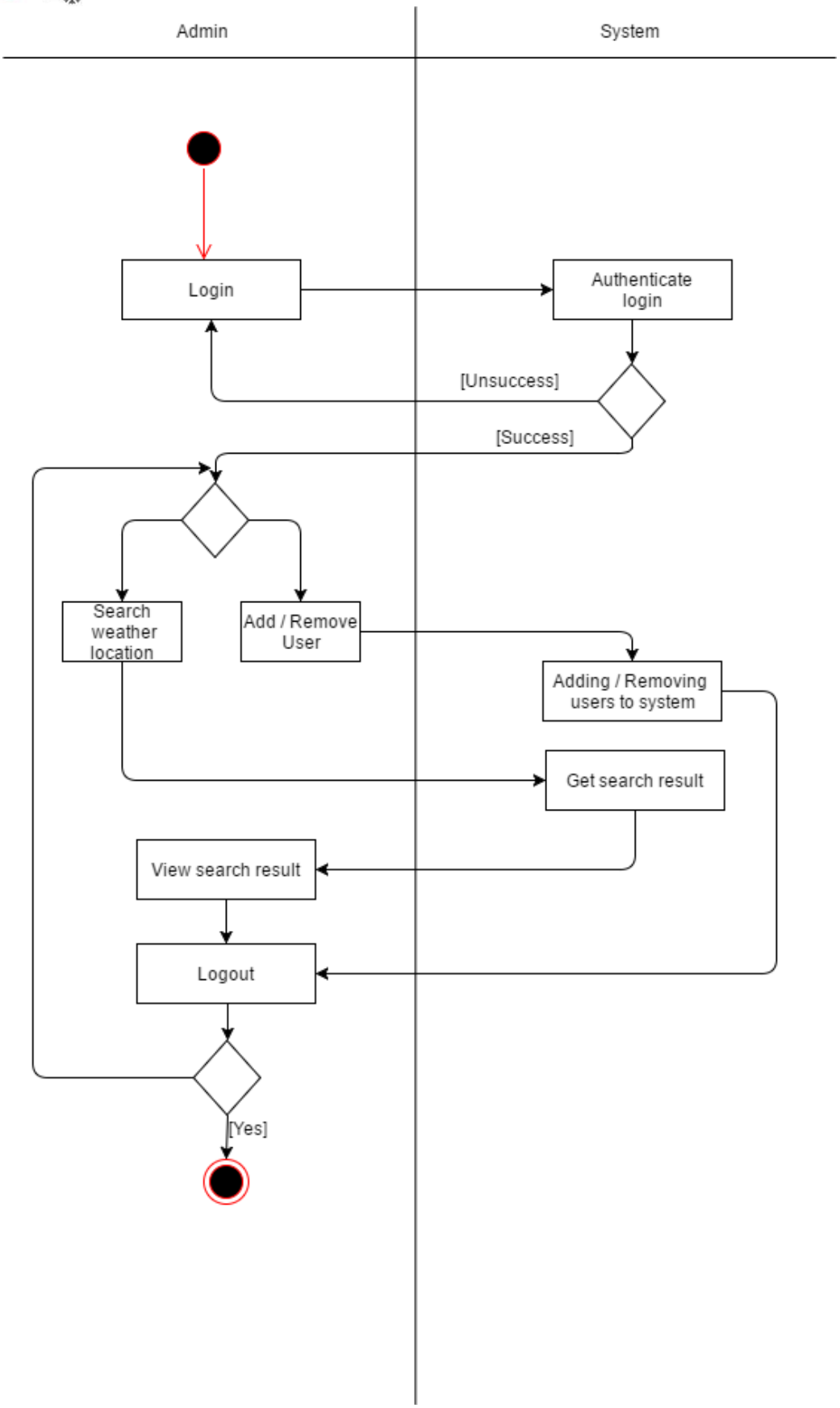

Figure 5.6: Activity Diagram for Administrator of Weather Station System

The above activity diagram is about the administrator's activity flow, and firstly, he/she should login to the system by confirming their identity. Then, administrator can remove or add weather station users to the system, and to view updated synopsis by each user at every prevailing weather station.

The following activity diagram illustrates the process involved when a public user is going to acquire necessary weather information. A public user should first sign in to the application, and in every time he/she would want to use the application, the user have to login in the beginning. Then, the user must update the weather state of his/her current location. After this update only they are allowed to acquire information as their needs.

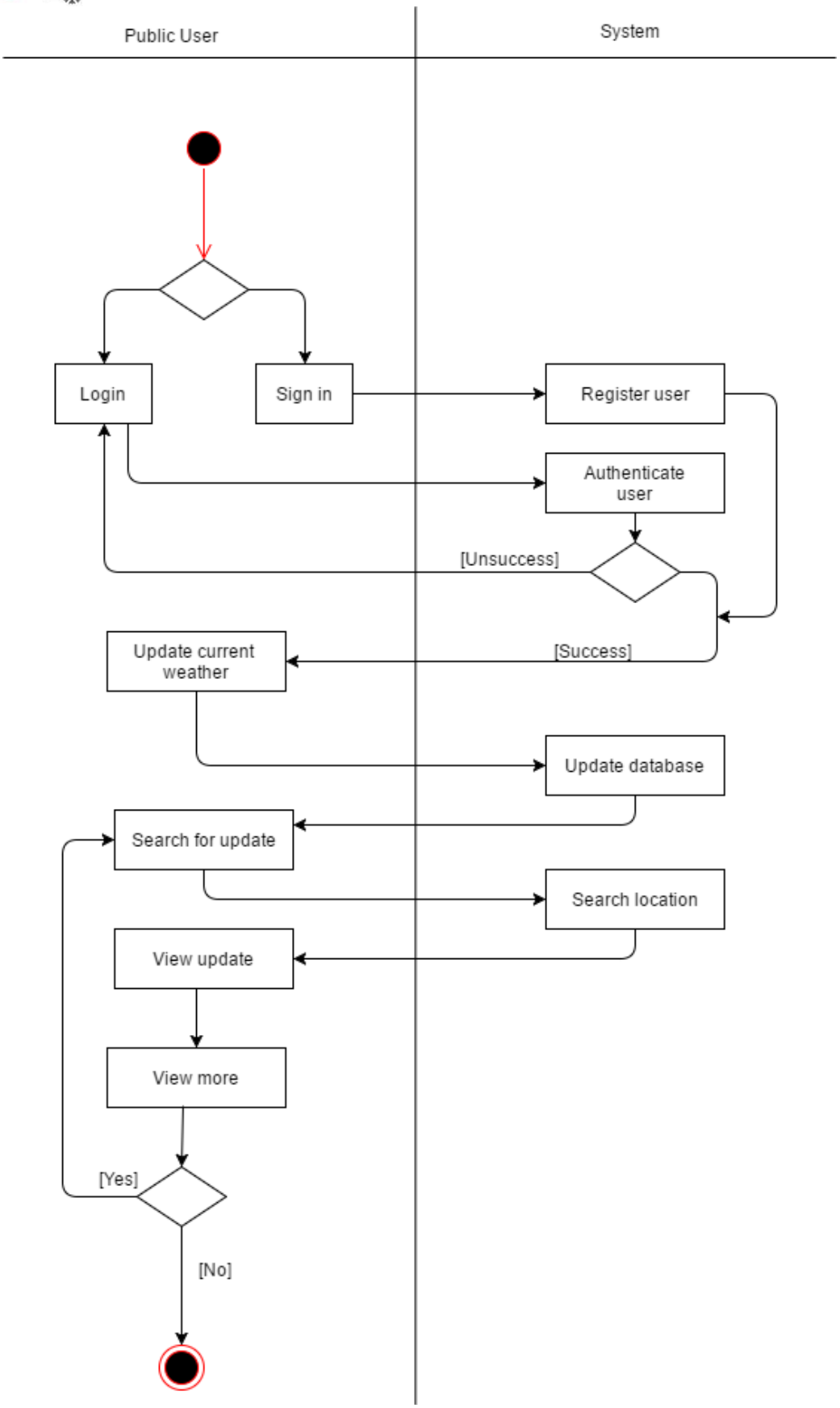

Figure 5.7: Activity Diagram for Public User

5.6 Entity-Relationship Diagram (E-R)

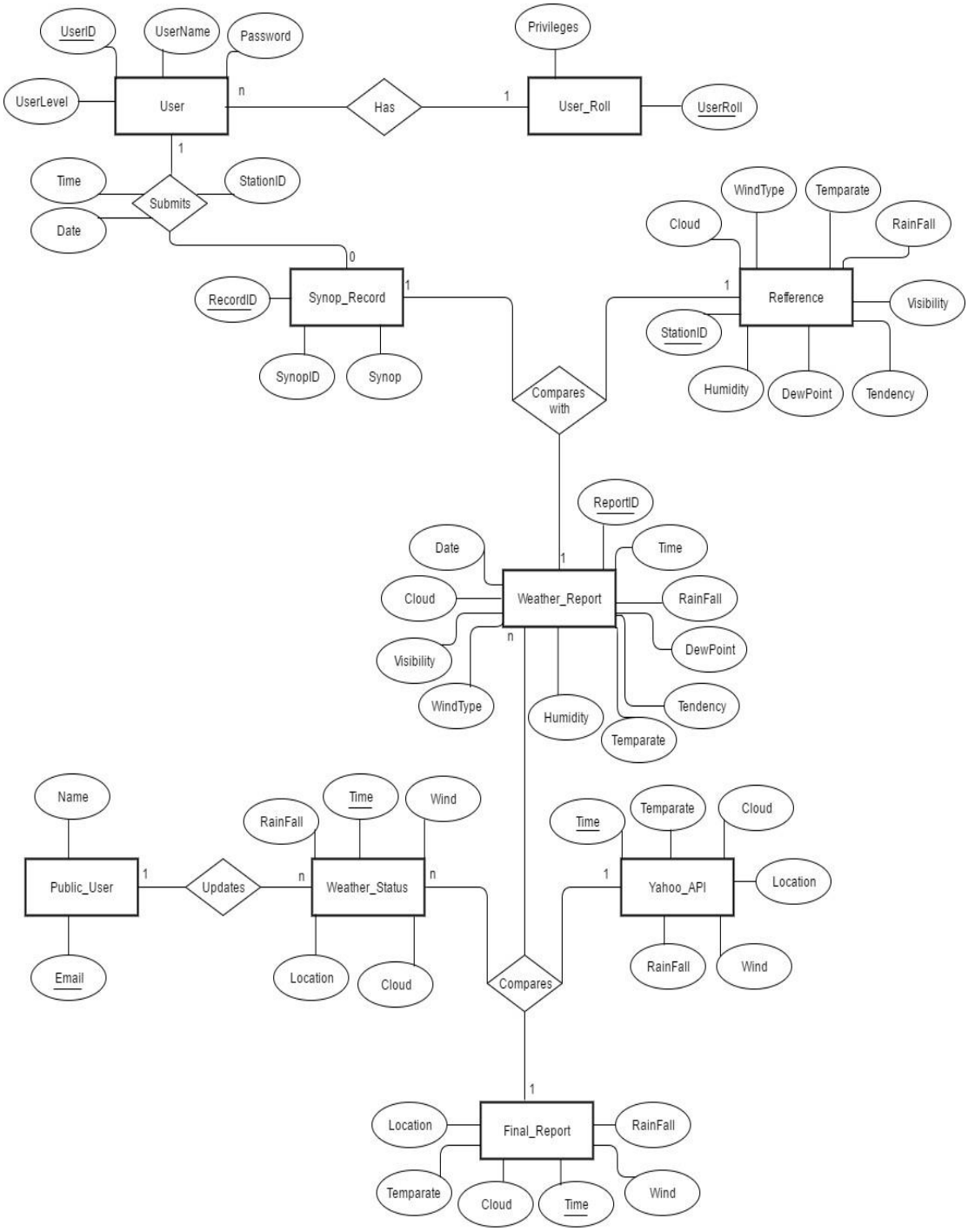

Figure 5.8: E-R Diagram

The Entity-Relationship diagram indicated the entire relationships among prevailing entities in the weather forecasting system.

5.7 Sequence Diagram

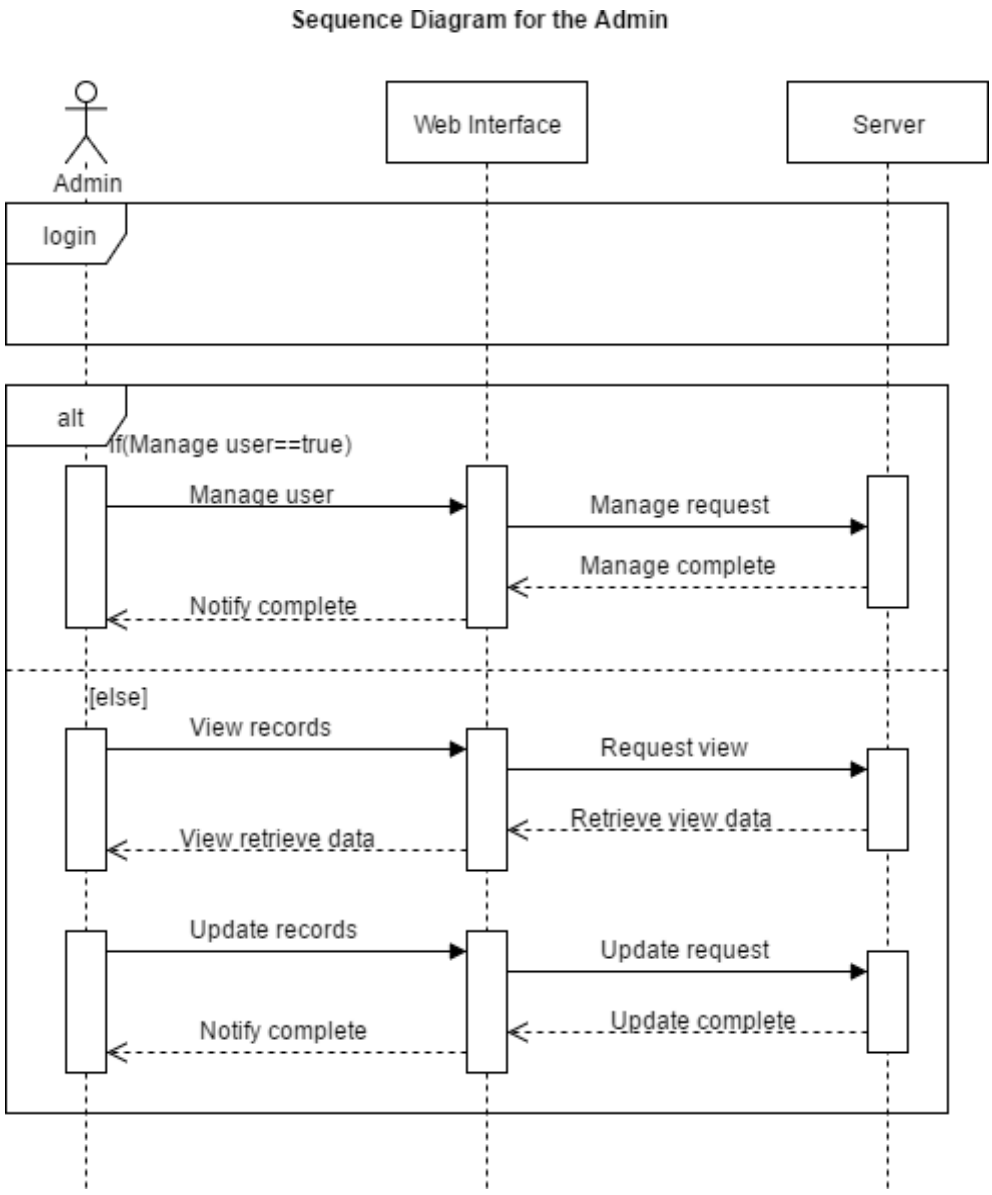

Figure 5.9: Sequence diagram for the Admin

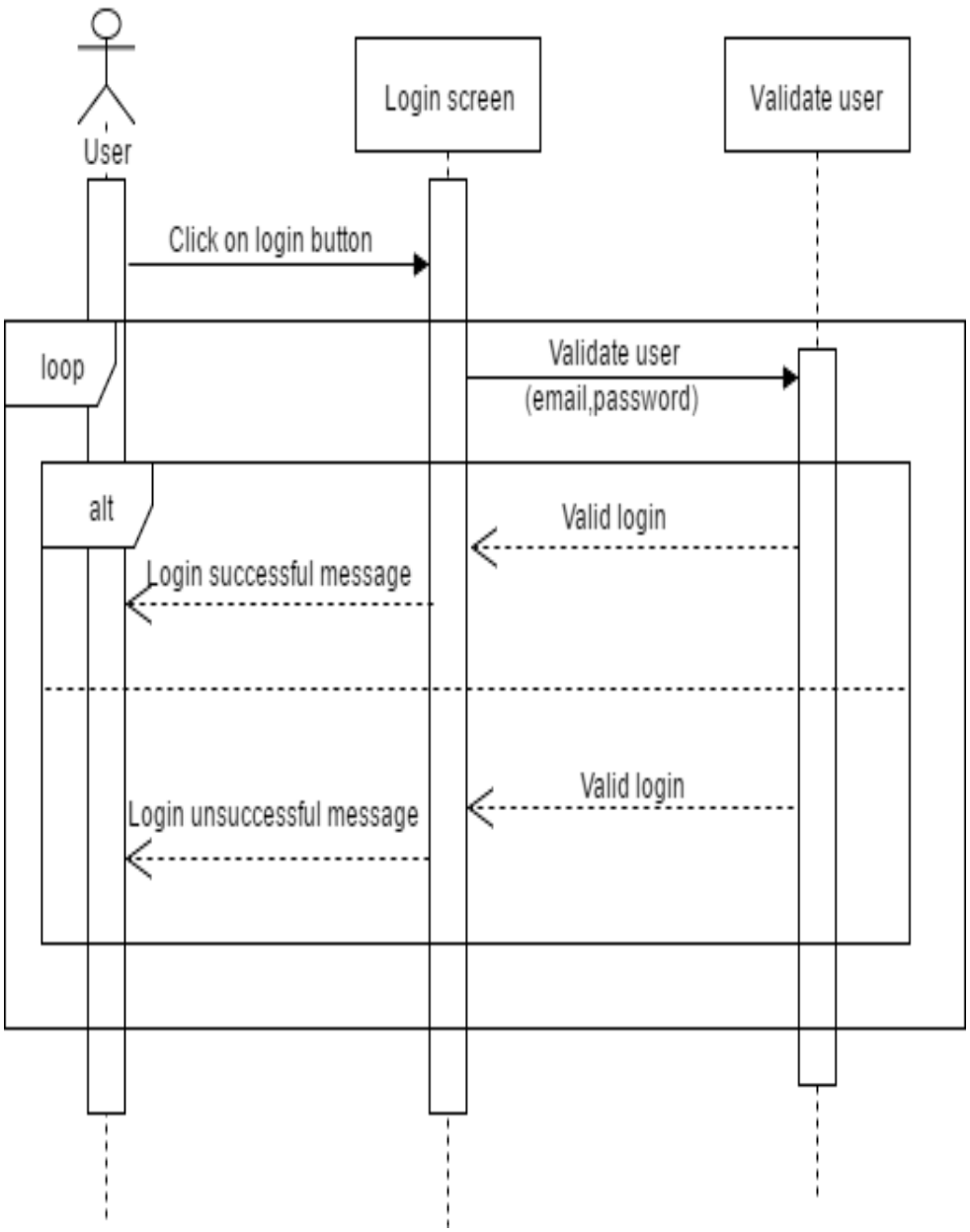

Figure 5.10: Sequence diagram for the Login

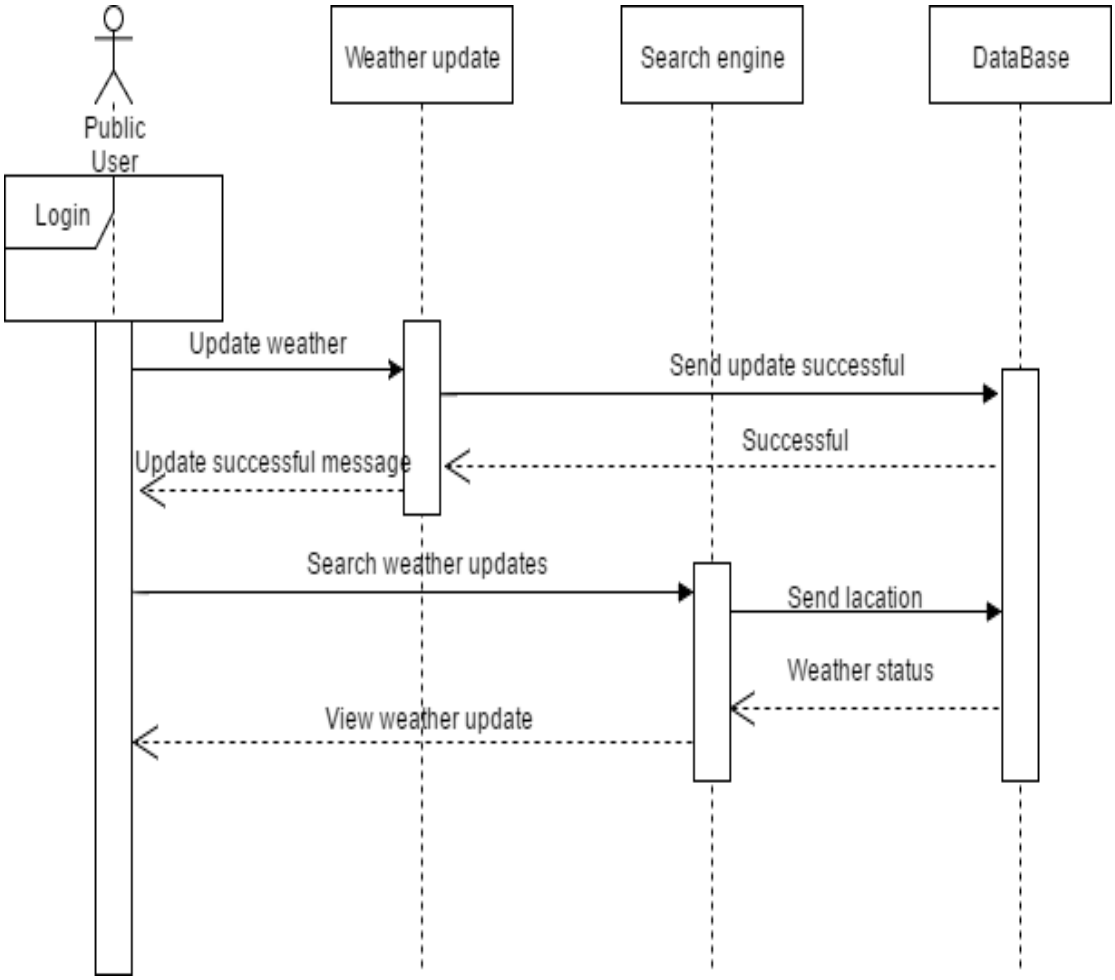

Figure 5.11: Sequence diagram for the Public User

Sequence Diagram for the weather station user

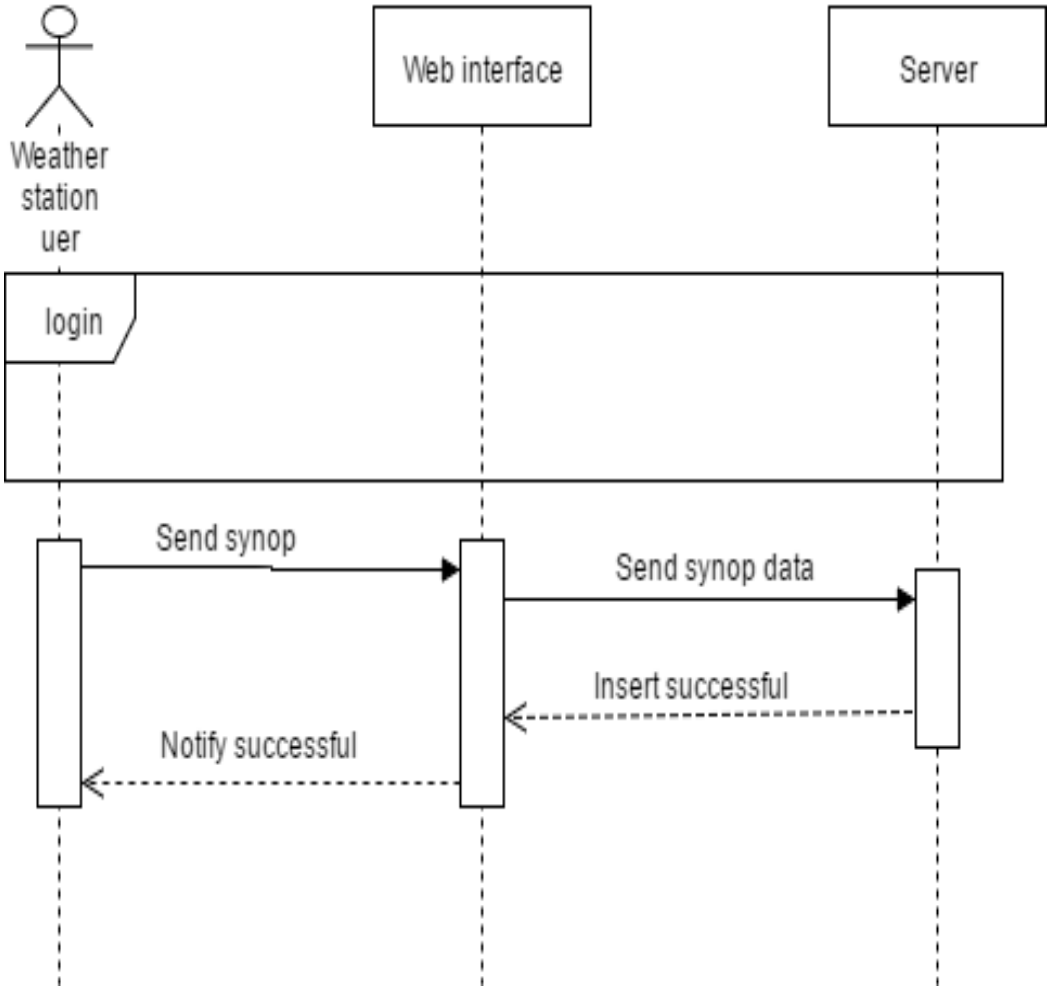

Figure 5.12: Sequence diagram for the Weather Station User

5.9 Summary

In this chapter, we have provided the design of the proposed system using various kind of diagrams in order to verify the fundamental idea, modules and components of the system which is going to be implemented. The next chapter will convey the implementation details of the system referring to this analysis and design factors.

Chapter 6

Implementation

6.1 Introduction

This chapter is described the way that how we are going to implement our proposed solution. Thus, in here our main attention focuses to the code techniques and progress of the implementing system.

6.2 Implementation

After selecting and confirming the project, we went several times to meet our clients to gather their requirements. As there are two clients we had to spend considerable time period to gather information. We had some sessions with the Meteorology Department to gather all their requirements that they have got at this moment. After gathering all the information, we decided to develop three main modules. They are:

- Weather Station Administrator Interface
- Weather Station User Interface
- Public User Interface

6.3 API

Dark Sky API — Overview

The Dark Sky API allows you to look up the weather anywhere on the globe, returning (where available):

- Current weather conditions
- Minute-by-minute forecasts out to one hour
- Hour-by-hour and day-by-day forecasts out to seven days
- Hour-by-hour and day-by-day observations going back decades
- Severe weather alerts in the US, Canada, European Union member nations, and Israel

Weather API Call Types

We provide two types of API requests to retrieve the weather anywhere in the world:

- The Forecast Request returns the current weather forecast for the next week.

- The Time Machine Request returns the observed or forecast weather conditions for a date in the past or future.

  Weather Conditions

  The Dark Sky API offers a full collection of meteorological conditions in 39 different languages, including:

- Apparent (feels-like) temperature
- Atmospheric pressure
- Cloud cover
- Dew point
- Humidity
- Liquid precipitation rate
- Moon phase
- Nearest storm distance
- Nearest storm direction
- Ozone
- Precipitation type
- Snowfall
- Sun rise/set
- Temperature
- Text summaries
- UV index
- Wind gust
- Wind speed
- Wind direction

  Both forecast and time machine requests return the same weather conditions, in the same convenient JSON format. You can parse the response directly, or use one several community-contributed libraries to interact with our API in the programming language of your choice.

Pricing and Attribution

Pricing is dead-simple.

- The first 1,000 API requests you make every day are free of charge.
- Every API request over the free daily limit costs $0.0001.

You are required to display the message "Powered by Dark Sky" that links to https://darksky.net/poweredby/ somewhere prominent in your app or service. If attribution isn't for you, email sales@darksky.net for details about white labeling.

Further details can be found in the terms of service.

If you have any questions, please check the FAQ and, if you can't find what you're looking for there, send us an email.

API Request Types

Forecast Request

```
https://api.darksky.net/forecast/[key]/[latitude],[longitude]
```

A Forecast Request returns the current weather conditions, a minute-by-minute forecast for the next hour (where available), an hour-by-hour forecast for the next 48 hours, and a day-by-day forecast for the next week.

Example Request

```
GET https://api.darksky.net/forecast/0123456789abcdef9876543210fedcba/42.3601,-71.0589

  {
    "latitude": 42.3601,
    "longitude": -71.0589,
    "timezone": "America/New_York",
    "currently": {
      "time": 1509993277,
      "summary": "Drizzle",
      "icon": "rain",
      "nearestStormDistance": 0,
      "precipIntensity": 0.0089,
      "precipIntensityError": 0.0046,
      "precipProbability": 0.9,
      "precipType": "rain",
      "temperature": 66.1,
      "apparentTemperature": 66.31,
      "dewPoint": 60.77,
      "humidity": 0.83,
      "pressure": 1010.34,
      "windSpeed": 5.59,
      "windGust": 12.03,
      "windBearing": 246,
      "cloudCover": 0.7,
      "uvIndex": 1,
      "visibility": 9.84,
      "ozone": 267.44
    },
    "minutely": {
```



```
            "summary": "Light rain stopping in 13 min., starting again 30 min. later.",
            "icon": "rain",
            "data": [{
               "time": 1509993240,
               "precipIntensity": 0.007,
               "precipIntensityError": 0.004,
               "precipProbability": 0.84,
               "precipType": "rain"
            },
          ...
          ]
        },
        "hourly": {
            "summary": "Rain starting later this afternoon, continuing until this evening.",
            "icon": "rain",
            "data": [{
               "time": 1509991200,
               "summary": "Mostly Cloudy",
               "icon": "partly-cloudy-day",
               "precipIntensity": 0.0007,
               "precipProbability": 0.1,
               "precipType": "rain",
               "temperature": 65.76,
               "apparentTemperature": 66.01,
               "dewPoint": 60.99,
               "humidity": 0.85,
               "pressure": 1010.57,
               "windSpeed": 4.23,
               "windGust": 9.52,
               "windBearing": 230,
               "cloudCover": 0.62,
               "uvIndex": 1,
               "visibility": 9.32,
               "ozone": 268.95
            },
          ...
          ]
        },
       "daily": {
            "summary": "Mixed precipitation throughout the week, with temperatures falling to
39°F on Saturday.",
            "icon": "rain",
            "data": [{
               "time": 1509944400,
               "summary": "Rain starting in the afternoon, continuing until evening.",
               "icon": "rain",
               "sunriseTime": 1509967519,
               "sunsetTime": 1510003982,
               "moonPhase": 0.59,
               "precipIntensity": 0.0088,
               "precipIntensityMax": 0.0725,
```



```
            "precipIntensityMaxTime": 1510002000,
            "precipProbability": 0.73,
            "precipType": "rain",
            "temperatureHigh": 66.35,
            "temperatureHighTime": 1509994800,
            "temperatureLow": 41.28,
            "temperatureLowTime": 1510056000,
            "apparentTemperatureHigh": 66.53,
            "apparentTemperatureHighTime": 1509994800,
            "apparentTemperatureLow": 35.74,
            "apparentTemperatureLowTime": 1510056000,
            "dewPoint": 57.66,
            "humidity": 0.86,
            "pressure": 1012.93,
            "windSpeed": 3.22,
            "windGust": 26.32,
            "windGustTime": 1510023600,
            "windBearing": 270,
            "cloudCover": 0.8,
            "uvIndex": 2,
            "uvIndexTime": 1509987600,
            "visibility": 10,
            "ozone": 269.45,
            "temperatureMin": 52.08,
            "temperatureMinTime": 1510027200,
            "temperatureMax": 66.35,
            "temperatureMaxTime": 1509994800,
            "apparentTemperatureMin": 52.08,
            "apparentTemperatureMinTime": 1510027200,
            "apparentTemperatureMax": 66.53,
            "apparentTemperatureMaxTime": 1509994800
          },
        ...
        ]
      },
      "alerts": [
      {
        "title": "Flood Watch for Mason, WA",
        "time": 1509993360,
        "expires": 1510036680,
        "description": "...FLOOD WATCH REMAINS IN EFFECT THROUGH LATE MONDAY NIGHT...\nTHE FLOOD WATCH CONTINUES FOR\n* A PORTION OF NORTHWEST WASHINGTON...INCLUDING THE FOLLOWING\nCOUNTY...MASON.\n* THROUGH LATE FRIDAY NIGHT\n* A STRONG WARM FRONT WILL BRING HEAVY RAIN TO THE OLYMPICS\nTONIGHT THROUGH THURSDAY NIGHT. THE HEAVY RAIN WILL PUSH THE\nSKOKOMISH RIVER ABOVE FLOOD STAGE TODAY...AND MAJOR FLOODING IS\nPOSSIBLE.\n* A FLOOD WARNING IS IN EFFECT FOR THE SKOKOMISH RIVER. THE FLOOD\nWATCH REMAINS IN EFFECT FOR MASON COUNTY FOR THE POSSIBILITY OF\nAREAL FLOODING ASSOCIATED WITH A MAJOR FLOOD.\n",
```



```
          "uri":
"http://alerts.weather.gov/cap/wwacapget.php?x=WA1255E4DB8494.FloodWatch.1255E4DCE
35CWA.SEWFFASEW.38e78ec64613478bb70fc6ed9c87f6e6"
        },
        ...
        ],
        {
          "flags": {
            "units": "us",
            ...
          }
        }
```


Request Parameters

Required parameters slot directly into the request URL. Optional parameters should be specified as HTTP query parameters.

- **key** required
  Your Dark Sky secret key. (Your secret key must be kept secret; in particular, do not embed it in JavaScript source code that you transmit to clients.)
- **latitude** required
  The latitude of a location (in decimal degrees). Positive is north, negative is south.
- **longitude** required
  The longitude of a location (in decimal degrees). Positive is east, negative is west.
- **exclude=[blocks]** optional

  Exclude some number of data blocks from the API response. This is useful for reducing latency and saving cache space. The value `blocks` should be a comma-delimeted list (without spaces) of any of the following:

- `currently`
- `minutely`
- `hourly`
- `daily`
- `alerts`
- `flags`
- **extend=hourly** optional
  When present, return hour-by-hour data for the next 168 hours, instead of the next 48. When using this option, we strongly recommend enabling HTTP compression.
- **lang=[language]** optional

Return `summary` properties in the desired language. (Note that units in the summary will be set according to the `units` parameter, so be sure to set both parameters appropriately.) `language` may be:

- `ar` : Arabic
- `az` : Azerbaijani
- `be` : Belarusian
- `bg` : Bulgarian
- `bn` : Bengali
- `bs` : Bosnian
- `ca` : Catalan
- `cs` : Czech
- `da` : Danish
- `de` : German
- `el` : Greek
- `en` : English (which is the default)
- `eo` : Esperanto
- `es` : Spanish
- `et` : Estonian
- `fi` : Finnish
- `fr` : French
- `he` : Hebrew
- `hi` : Hindi
- `hr` : Croatian
- `hu` : Hungarian
- `id` : Indonesian
- `is` : Icelandic
- `it` : Italian
- `ja` : Japanese
- `ka` : Georgian
- `kn` : Kannada
- `ko` : Korean
- `kw` : Cornish
- `lv` : Latvian
- `ml` : Malayam
- `mr` : Marathi
- `nb` : Norwegian Bokmål

- `nl` : Dutch
- `no` : Norwegian Bokmål (alias for `nb`)
- `pa` : Punjabi
- `pl` : Polish
- `pt` : Portuguese
- `ro` : Romanian
- `ru` : Russian
- `sk` : Slovak
- `sl` : Slovenian
- `sr` : Serbian
- `sv` : Swedish
- `ta` : Tamil
- `te` : Telugu
- `tet` : Tetum
- `tr` : Turkish
- `uk` : Ukrainian
- `ur` : Urdu
- `x-pig-latin` : Igpay Atinlay
- `zh` : simplified Chinese
- `zh-tw` : traditional Chinese

If you require a language not listed here, please consider contributing to our API translation module on Github.

- **units=[units]** optional

Return weather conditions in the requested units. `[units]` should be one of the following:

- `auto` : automatically select units based on geographic location
- `ca` : same as `si`, except that `windSpeed` and `windGust` are in kilometers per hour
- `uk2` : same as `si`, except that `nearestStormDistance` and `visibility` are in miles, and `windSpeed` and `windGust` in miles per hour
- `us` : Imperial units (the default)
- `si` : SI units

SI units are as follows:

- `summary`: Any summaries containing temperature or snow accumulation units will have their values in degrees Celsius or in centimeters (respectively).
- `nearestStormDistance`: Kilometers.
- `precipIntensity`: Millimeters per hour.
- `precipIntensityMax`: Millimeters per hour.
- `precipAccumulation`: Centimeters.
- `temperature`: Degrees Celsius.
- `temperatureMin`: Degrees Celsius.
- `temperatureMax`: Degrees Celsius.
- `apparentTemperature`: Degrees Celsius.
- `dewPoint`: Degrees Celsius.
- `windSpeed`: Meters per second.
- `windGust`: Meters per second.
- `pressure`: Hectopascals.
- `visibility`: Kilometers.

Time Machine Request

```
https://api.darksky.net/forecast/[key]/[latitude],[longitude],[time]
```

A Time Machine Request returns the observed (in the past) or forecasted (in the future) hour-by-hour weather and daily weather conditions for a particular date. A Time Machine request is identical in structure to a Forecast Request, except:

- The `currently` data point will refer to the time provided, rather than the current time.
- The `minutely` data block will be omitted, unless you are requesting a time within an hour of the present.
- The `hourly` data block will contain data points starting at midnight (local time) of the day requested, and continuing until midnight (local time) of the following day.
- The `daily` data block will contain a single data point referring to the requested date.
- The `alerts` data block will be omitted.

Example Request

```
GET https://api.darksky.net/forecast/0123456789abcdef9876543210fedcba/42.3601,-71.0589,255657600?exclude=currently,flags

{
  "latitude": 42.3601,
  "longitude": -71.0589,
  "timezone": "America/New_York",
  "hourly": {
```



```
  "summary": "Snow (6–9 in.) and windy starting in the afternoon.",
  "icon": "snow",
  "data": [
    {
      "time": 255589200,
      "summary": "Mostly Cloudy",
      "icon": "partly-cloudy-night",
      "precipIntensity": 0,
      "precipProbability": 0,
      "temperature": 22.8,
      "apparentTemperature": 16.46,
      "dewPoint": 15.51,
      "humidity": 0.73,
      "pressure": 1026.78,
      "windSpeed": 4.83,
      "windBearing": 354,
      "cloudCover": 0.78,
      "uvIndex": 0,
      "visibility": 9.62
    },
    ...
  ]
 },
 "daily": {
  "data": [
    {
      "time": 255589200,
      "summary": "Snow (9–14 in.) and windy starting in the afternoon.",
      "icon": "snow",
      "sunriseTime": 255613996,
      "sunsetTime": 255650764,
      "moonPhase": 0.97,
      "precipIntensity": 0.0354,
      "precipIntensityMax": 0.1731,
      "precipIntensityMaxTime": 255657600,
      "precipProbability": 1,
      "precipAccumulation": 7.337,
      "precipType": "snow",
      "temperatureHigh": 31.84,
      "temperatureHighTime": 255632400,
      "temperatureLow": 28.63,
      "temperatureLowTime": 255697200,
      "apparentTemperatureHigh": 20.47,
      "apparentTemperatureHighTime": 255625200,
      "apparentTemperatureLow": 13.03,
      "apparentTemperatureLowTime": 255697200,
      "dewPoint": 24.72,
      "humidity": 0.86,
      "pressure": 1016.41,
      "windSpeed": 22.93,
      "windBearing": 56,
```



```
            "cloudCover": 0.95,
            "uvIndex": 1,
            "uvIndexTime": 255621600,
            "visibility": 4.83,
            "temperatureMin": 22.72,
            "temperatureMinTime": 255596400,
            "temperatureMax": 32.04,
            "temperatureMaxTime": 255672000,
            "apparentTemperatureMin": 11.13,
            "apparentTemperatureMinTime": 255650400,
            "apparentTemperatureMax": 20.47,
            "apparentTemperatureMaxTime": 255625200
          }
        ]
      },
      "offset": -5
    }
```


Request Parameters

Required parameters slot directly into the request URL. Optional parameters should be specified as HTTP query parameters.

- **key** required
  Your Dark Sky secret key. (Your secret key must be kept secret; in particular, do not embed it in JavaScript source code that you transmit to clients.)
- **latitude** required
  The latitude of a location (in decimal degrees). Positive is north, negative is south.
- **longitude** required
  The longitude of a location (in decimal degrees). Positive is east, negative is west.
- **time** required
  Either be a UNIX time (that is, seconds since midnight GMT on 1 Jan 1970) or a string formatted as follows: `[YYYY]-[MM]-[DD]T[HH]:[MM]:[SS][timezone]`. `timezone` should either be omitted (to refer to local time for the location being requested), `Z` (referring to GMT time), or `+[HH][MM]` or `-[HH][MM]` for an offset from GMT in hours and minutes. The timezone is only used for determining the time of the request; the response will always be relative to the local time zone.
- **exclude=[blocks]** optional

  Exclude some number of data blocks from the API response. This is useful for reducing latency and saving cache space. The value `blocks` should be a comma-delimeted list (without spaces) of any of the following:

- `currently`

- minutely
- hourly
- daily
- alerts
- flags
- **lang=[language]** optional

  Return summary properties in the desired language. (Note that units in the summary will be set according to the units parameter, so be sure to set both parameters appropriately.) language may be:

- ar : Arabic
- az : Azerbaijani
- be : Belarusian
- bg : Bulgarian
- bn : Bengali
- bs : Bosnian
- ca : Catalan
- cs : Czech
- da : Danish
- de : German
- el : Greek
- en : English (which is the default)
- eo : Esperanto
- es : Spanish
- et : Estonian
- fi : Finnish
- fr : French
- he : Hebrew
- hi : Hindi
- hr : Croatian
- hu : Hungarian
- id : Indonesian
- is : Icelandic
- it : Italian
- ja : Japanese
- ka : Georgian

- kn : Kannada
- ko : Korean
- kw : Cornish
- lv : Latvian
- ml : Malayam
- mr : Marathi
- nb : Norwegian Bokmål
- nl : Dutch
- no : Norwegian Bokmål (alias for nb)
- pa : Punjabi
- pl : Polish
- pt : Portuguese
- ro : Romanian
- ru : Russian
- sk : Slovak
- sl : Slovenian
- sr : Serbian
- sv : Swedish
- ta : Tamil
- te : Telugu
- tet : Tetum
- tr : Turkish
- uk : Ukrainian
- ur : Urdu
- x-pig-latin : Igpay Atinlay
- zh : simplified Chinese
- zh-tw : traditional Chinese

If you require a language not listed here, please consider contributing to our API translation module on Github.

- **units=[units]** optional

Return weather conditions in the requested units. [units] should be one of the following:

- auto : automatically select units based on geographic location
- ca : same as si , except that windSpeed is in kilometers per hour

- `uk2`: same as `si`, except that `nearestStormDistance` and `visibility` are in miles and `windSpeed` is in miles per hour
- `us`: Imperial units (the default)
- `si`: SI units

SI units are as follows:

- `summary`: Any summaries containing temperature or snow accumulation units will have their values in degrees Celsius or in centimeters (respectively).
- `nearestStormDistance`: Kilometers.
- `precipIntensity`: Millimeters per hour.
- `precipIntensityMax`: Millimeters per hour.
- `precipAccumulation`: Centimeters.
- `temperature`: Degrees Celsius.
- `temperatureMin`: Degrees Celsius.
- `temperatureMax`: Degrees Celsius.
- `apparentTemperature`: Degrees Celsius.
- `dewPoint`: Degrees Celsius.
- `windSpeed`: Meters per second.
- `pressure`: Hectopascals.
- `visibility`: Kilometers.

Response Format

API responses consist of a UTF-8-encoded, JSON-formatted object with the following properties:

- **latitude** required
  The requested latitude.
- **longitude** required
  The requested longitude.
- **timezone** (e.g. `America/New_York`) required
  The IANA timezone name for the requested location. This is used for text summaries and for determining when `hourly` and `daily` data block objects begin.
- **offset** deprecated
  The current timezone offset in hours. (Use of this property will almost certainly result in Daylight Saving Time bugs. Please use `timezone`, instead.)
- **currently** optional
  A data point containing the current weather conditions at the requested location.

- **minutely** optional

  A data block containing the weather conditions minute-by-minute for the next hour.
- **hourly** optional

  A data block containing the weather conditions hour-by-hour for the next two days.
- **daily** optional

  A data block containing the weather conditions day-by-day for the next week.
- **alerts** optional

  An alerts array, which, if present, contains any severe weather alerts pertinent to the requested location.
- **flags** optional

  A flags object containing miscellaneous metadata about the request.

Data Point Object

A data point object contains various properties, each representing the average (unless otherwise specified) of a particular weather phenomenon occurring during a period of time: an instant in the case of `currently`, a minute for `minutely`, an hour for `hourly`, and a day for `daily`. These properties are:

- **apparentTemperature** optional, only on `hourly` and `currently`

  The apparent (or “feels like”) temperature in degrees Fahrenheit.
- **apparentTemperatureHigh** optional, only on `daily`

  The daytime high apparent temperature.
- **apparentTemperatureHighTime** optional, only on `daily`

  The UNIX time representing when the daytime high apparent temperature occurs.
- **apparentTemperatureLow** optional, only on `daily`

  The overnight low apparent temperature.
- **apparentTemperatureLowTime** optional, only on `daily`

  The UNIX time representing when the overnight low apparent temperature occurs.
- **apparentTemperatureMax** optional, only on `daily`

  The maximum apparent temperature during a given date.
- **apparentTemperatureMaxTime** optional, only on `daily`

  The UNIX time representing when the maximum apparent temperature during a given date occurs.
- **apparentTemperatureMin** optional, only on `daily`

  The minimum apparent temperature during a given date.
- **apparentTemperatureMinTime** optional, only on `daily`

  The UNIX time representing when the minimum apparent temperature during a given date occurs.
- **cloudCover** optional

The percentage of sky occluded by clouds, between `0` and `1`, inclusive.

- **dewPoint** optional

  The dew point in degrees Fahrenheit.

- **humidity** optional

  The relative humidity, between `0` and `1`, inclusive.

- **icon** optional

  A machine-readable text summary of this data point, suitable for selecting an icon for display. If defined, this property will have one of the following values: `clear-day`, `clear-night`, `rain`, `snow`, `sleet`, `wind`, `fog`, `cloudy`, `partly-cloudy-day`, or `partly-cloudy-night`. (Developers should ensure that a sensible default is defined, as additional values, such as `hail`, `thunderstorm`, or `tornado`, may be defined in the future.)

- **moonPhase** optional, only on `daily`

  The fractional part of the lunation number during the given day: a value of `0` corresponds to a new moon, `0.25` to a first quarter moon, `0.5` to a full moon, and `0.75` to a last quarter moon. (The ranges in between these represent waxing crescent, waxing gibbous, waning gibbous, and waning crescent moons, respectively.)

- **nearestStormBearing** optional, only on `currently`

  The approximate direction of the nearest storm in degrees, with true north at 0° and progressing clockwise. (If `nearestStormDistance` is zero, then this value will not be defined.)

- **nearestStormDistance** optional, only on `currently`

  The approximate distance to the nearest storm in miles. (A storm distance of `0` doesn't necessarily refer to a storm at the requested location, but rather a storm in the vicinity of that location.)

- **ozone** optional

  The columnar density of total atmospheric ozone at the given time in Dobson units.

- **precipAccumulation** optional, only on `hourly`, `currently` and `daily`

  The amount of snowfall accumulation expected to occur (over the hour or day, respectively), in inches. (If no snowfall is expected, this property will not be defined.)

- **precipIntensity** optional

  The intensity (in inches of liquid water per hour) of precipitation occurring at the given time. This value is conditional on probability (that is, assuming any precipitation occurs at all).

- **precipIntensityError** optional

  The standard deviation of the distribution of `precipIntensity`. (We only return this property when the full distribution, and not merely the expected mean, can be estimated with accuracy.)

- **precipIntensityMax** optional, only on `daily`

  The maximum value of `precipIntensity` during a given day.

- **precipIntensityMaxTime** optional, only on `daily`

  The UNIX time of when `precipIntensityMax` occurs during a given day.

- **precipProbability** optional
  The probability of precipitation occurring, between `0` and `1`, inclusive.
- **precipType** optional
  The type of precipitation occurring at the given time. If defined, this property will have one of the following values: `"rain"`, `"snow"`, or `"sleet"` (which refers to each of freezing rain, ice pellets, and "wintery mix"). (If `precipIntensity` is zero, then this property will not be defined. Additionally, due to the lack of data in our sources, historical `precipType` information is usually estimated, rather than observed.)
- **pressure** optional
  The sea-level air pressure in millibars.
- **summary** optional
  A human-readable text summary of this data point. (This property has millions of possible values, so don't use it for automated purposes: use the `icon` property, instead!)
- **sunriseTime** optional, only on `daily`
  The UNIX time of when the sun will rise during a given day.
- **sunsetTime** optional, only on `daily`
  The UNIX time of when the sun will set during a given day.
- **temperature** optional, only on `hourly` and `currently`
  The air temperature in degrees Fahrenheit.
- **temperatureHigh** optional, only on `daily`
  The daytime high temperature.
- **temperatureHighTime** optional, only on `daily`
  The UNIX time representing when the daytime high temperature occurs.
- **temperatureLow** optional, only on `daily`
  The overnight low temperature.
- **temperatureLowTime** optional, only on `daily`
  The UNIX time representing when the overnight low temperature occurs.
- **temperatureMax** optional, only on `daily`
  The maximum temperature during a given date.
- **temperatureMaxTime** optional, only on `daily`
  The UNIX time representing when the maximum temperature during a given date occurs.
- **temperatureMin** optional, only on `daily`
  The minimum temperature during a given date.
- **temperatureMinTime** optional, only on `daily`
  The UNIX time representing when the minimum temperature during a given date occurs.
- **time** required
  The UNIX time at which this data point begins. `minutely` data point are always aligned to the top of the minute, `hourly` data point objects to the top of the hour, `daily` data point objects to

midnight of the day, and `currently` data point object to the point of time provided all according to the local time zone.

- **uvIndex** optional
  The UV index.
- **uvIndexTime** optional, only on `daily`
  The UNIX time of when the maximum `uvIndex` occurs during a given day.
- **visibility** optional
  The average visibility in miles, capped at 10 miles.
- **windBearing** optional
  The direction that the wind is coming from in degrees, with true north at 0° and progressing clockwise. (If `windSpeed` is zero, then this value will not be defined.)
- **windGust** optional
  The wind gust speed in miles per hour.
- **windGustTime** optional, only on `daily`
  The time at which the maximum wind gust speed occurs during the day.
- **windSpeed** optional
  The wind speed in miles per hour.

Data Block Object

A data block object represents the various weather phenomena occurring over a period of time. Such objects contain the following properties:

- **data** required
  An array of data points, ordered by time, which together describe the weather conditions at the requested location over time.
- **summary** optional
  A human-readable summary of this data block.
- **icon** optional
  A machine-readable text summary of this data block. (May take on the same values as the `icon` property of data points.)

Alerts Array

The alerts array contains objects representing the severe weather warnings issued for the requested location by a governmental authority (please see our data sources page for a list of sources). Objects in the alerts array contain the following properties:

- **description** required
  A detailed description of the alert.
- **expires** required
  The UNIX time at which the alert will expire.

- **regions** required
  An array of strings representing the names of the regions covered by this weather alert.
- **severity** required
  The severity of the weather alert. Will take one of the following values: `"advisory"` (an individual should be aware of potentially severe weather), `"watch"` (an individual should prepare for potentially severe weather), or `"warning"` (an individual should take immediate action to protect themselves and others from potentially severe weather).
- **time** required
  The UNIX time at which the alert was issued.
- **title** required
  A brief description of the alert.
- **uri** required
  An HTTP(S) URI that one may refer to for detailed information about the alert.

Flags Object

The flags object contains various metadata information related to the request. This object may optionally contain any of the following properties:

- **darksky-unavailable** optional
  The presence of this property indicates that the Dark Sky data source supports the given location, but a temporary error (such as a radar station being down for maintenance) has made the data unavailable.
- **nearest-station** required
  The distance to the nearest weather station that contributed data to this response. Note, however, that many other stations may have also been used; this value is primarily for debugging purposes. This property's value is in miles (if US units are selected) or kilometers (if SI units are selected).
- **sources** required
  This property contains an array of IDs for each data source utilized in servicing this request.
- **units** required
  Indicates the units which were used for the data in this request.

Response Headers

The API will set the following HTTP response headers to values useful to developers:

- **Cache-Control** optional
  Set to a conservative value for data caching purposes based on the data present in the response body.
- **Expires** deprecated
  Set to a conservative value for data caching purposes based on the data present in the response

body.

- **X-Forecast-API-Calls** optional
  The number of API requests made by the given API key for today.
- **X-Response-Time** optional
  The server-side response time of the request.

Notes

- Never make any assumptions about the presence of data or lengths of arrays. For example, a lack of data in our data sources may cause data to be missing; or Daylight Savings Time may cause a day to consist of 23 or 25 hours (instead of the usual 24); or, at high latitudes, a given day may not have a sunrise or sunset. Always check for the presence of data before trying to use it.
- Overnight low temperatures usually occur around dawn, and so will usually occur on a different date from the daytime high temperatures. (This is the primary difference between `temperatureLow` and `temperatureMin`.)
- Summaries on the `hourly` data block actually only cover up to a maximum of 24 hours, rather than the full time period in the data block. We found that covering the full 48 hours could be, in a number of circumstances, far too wordy to be practical.
- Summaries and icons on `daily` data point objects actually cover the period from 4AM to 4AM, rather than the stated time period of midnight to midnight. We found that the summaries so generated were less awkward.
- The Forecast Data API supports HTTP compression. We heartily recommend using it, as it will make responses much smaller over the wire. To enable it, simply add an `Accept-Encoding: gzip` header to your request. (Most HTTP client libraries wrap this functionality for you, please consult your library's documentation for details.)

6.4 Summary

This chapter has described our implementation of the project. We have number of user interfaces to create and we have a huge database to maintain. Therefore our implementation is still in progress. In next chapter the summary of the entire scenario that we have talked so far is included.

Chapter 7

Discussion

7.1. Introduction

In previous chapters, we have described the problem that we have, the solution that we gave for this problem, how we are going to analyze the problem and to create an overall design for that. In this chapter, we are going to give a quick summary on the things what we have discussed so far.

7.2. Discussion

It is beneficial if we can know about the current weather condition of a particular location as desired since it might help our day to day life. It is more effective if we can get quickly updated on current weather status of a required location, as it make easy to handle our activities.

Therefore, we thought of implementing a system for weather forecasting in order to facilitate these needs, and especially, a system for Meteorological Department. The department requires a system to get all the updates done by each weather station for every three hours by a specific user who logs into the system. The solution for general public need is solved through a crowdsource application for mobile phones. The user can update their current location's weather status and also get required updates from it. This application is supported by Yahoo Weather API, and information from weather stations.

Though there are many similar applications for this, they are not enriched with the latest data, but this application is frequently updates by its users. Thus, it provides not only information regarding the main cities of the country, but also other regional locations' too.

We use several technological approaches for the design of the proposed solution. GPS technology for detecting the user's location, Node.js, AngularJS, MongoDB for application development, and Google Maps for demonstrating weather information graphically.

Then, we have to ensure and evaluate the performance of the system by testing, and implement the system. Especially, the maintenance of the system is much important fact. Throughout the process, we should make sure that the system is enriched with

reliability, accuracy, fast as possible, feasibility and user-friendly.

7.3 Summary

This chapter has provided an overall description about all the things of our solution including background, proposed solution, similar approaches, technology adapted, our approach, design and implementation.

Chapter 8

References

[1] http://www.physics.uwo.ca/~whocking/p103/instrum.html, *introduction to meteorology-tools for Science*

[2] http://www.indiawaterportal.org/articles/measurement-weather-parameters-data-collection-and-analysis-presentation-acwadam, *measurements of weather parameters*

[3] http://www.meteor.wisc.edu/~hopkins/aos100/sfc-anl.htm

[4] http://heavy.com/tech/2015/04/top-5-best-free-weather-channel-app-apps-for- iphone-ipad-iOS/

[5] https://angularjs.org/

[6] http://www.codeproject.com/Articles/1037052/Introduction-to-MongoDB

[7] https://nodejs.org

[8] https://www.raywenderlich.com

[9] http://www.mio.com/technology-what-is-gps.htm

[10] https://developers.google.com

Appendix A

Crowdsourcing

The process of obtaining needed information, input, services, ideas, or content by soliciting contributions from a large group of people, especially from an online community. Furthermore crowdsourcing is the practice of engaging a 'crowd' or group for a common goal often innovation, problem solving, or efficiency. It is powered by new technologies, social media and web. Crowdsourcing can take place on many different levels and across various industries.

However, crowdsourcing also brings with it quite a few disadvantages.

- The users may not be able to connect the Internet as soon as they expect.
- The users may be unable to verify the location and information due to weak network connections.
- The low degree of reliability on information retrieved from the users.
- The inability to find users from all over the country to gain full coverage, and to provide with required information.
- The difficulties in supplying computer and Internet facilities to all weather stations, and in repairing, fixing them.
- The problem of not having sufficient amount of staff for stations

Weather Synopsis

SYNOP (surface synoptic observations) is a numerical code used for reporting weather observations made by manned and automated weather stations. It is a description of weather patterns which is affected a large area of an island, state, region or a continent.

Appendix B

External Interface Requirements

(1) User Interfaces

- IOS Interface

The IOS interface of IOS Weather is design for the users who needs to know current weather status of any place.

Features of the iOS interface of IOS Weather will include a login page, searching for weather updates, update user's current weather updates and show searched weather results.

Visual design of the interface will be make in a way that any user without prior experience of the interface will be able to fulfill his information requirements with a little effort.

(2) Hardware interfaces

Since neither the mobile application nor the web portal have any designated hardware, it does not have any direct hardware interfaces. The physical GPS is managed by the GPS application in the mobile phone.

(3) Software Interfaces

The IOS Weather app is to be developed under the IOS operating systems.

The mobile application communicates with the GPS in order to get geographical information about where the user is located and the visual representation of it,

- Incoming and outgoing items

* Incoming data consist of updates from the server regarding to the weather and search results.

* Outgoing data consist of current weather update sent by user, new login information and synopsis.

(4) Communication protocols and interfaces

Communication will occur in occasional, short bursts between a user's phone and the server in the following situations:

- Whenever a user creates a new login in iOS phone
- Whenever user send current weather updates to the server
- Whenever server send search results to the user

Communication will occur between web interface and the server between following situations:

- Whenever officers update synopsis.
- Whenever users search for weather updates.
- Whenever admins add/remove user details

Screenshots

Figure- Splash Screen.

Figure- Today Screen.

Figure- Tomorrow Screen.

Figure- Seven Days Screen.

Figure- Popup Details Screen.