# A Project Report

On

**Weather Tracking System**

***Submitted in partial fulfillment of***
***the requirement for the award of***
***the degree of***

**Bachelor of Technology**

GALGOTIAS UNIVERSITY
(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

**Under TheSupervision**
Dr. Michel Raj Tf
**Designation**

Submitted By

Vaibhav Srivastav 18021011547 / 18SCSE1010313

Shashank Malviya 18021011334/ 18SCSE1010084

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERINGGALGOTIAS UNIVERSITY, GREATER NOIDA INDIA**

## SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
## GALGOTIAS UNIVERSITY, GREATER NOIDA

## CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **"Weather Tracking System "** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021**, under the supervision of **Mr. Michel Raj TF, Assistant Professor**, **Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

18SCSE1010084 – Shashank Malviya
18SCSE1010313 – Vaibhav Srivastava

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Mr.Michel Raj Tf, Assistant Professor)

# CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **18SCSE1010084 – SHASHANK MALVIYA, 18SCSE1010313 – VAIBHAV SRIVASTAV** has been held on ____________________________________________________________________and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

**Signature of Examiner(s)** **Signature of Supervisor(s)**

**Signature of Project Coordinator** **Signature of Dean**

Date: Place:

## ABSTRACT

Weather forecasting is the application of science and technology to predict the state of the atmosphere for a given location. Ancient weather forecasting methods usually relied on observed patterns of events, also termed pattern recognition. For example, it mightbe observed that if the sunset was particularly red, the following day often brought fair weather. However, not all of these predictions prove reliable. Here this system will predict weather based on parameters such as temperature, humidity and wind. This system is a web application with effective graphical user interface. User will login to the system using his user ID and password. User will enter current temperature; humidityand wind, System will take this parameter and will predict weather from previous data in database. The role of the admin is to add previous weather data in database, so that system will calculate weather based on these data. Weather forecasting system takes parameters such as temperature, humidity, and wind and will forecast weather based on previous record therefore this prediction will prove reliable. This system can be used in Air Traffic, Marine, Agriculture, Forestry, Military, and Navy etc.

# CONTENTS

# LIST OF FIGURES

# CHAPTER 1

## INTRODUCTION

### 1.1 DOMAIN INTRODUCTION

Weather forecasting is the application of science and technology to predict the state of the atmosphere for a given location. Ancient weather forecasting methods usually relied on observed patterns of events, also termed pattern recognition. For example, it might be observed that if the sunset was particularly red, the following day often brought fair weather. However, not all of these predictions prove reliable. Here this system will predict weather based on parameters such as temperature, humidity and wind. This system is a web application with effective graphical user interface. User will login to the system using his user ID and password. User will enter current temperature; humidity and wind, System will take this parameter and will predict weather from previous data in database. The role of the admin is to add previous weather data in database, so that system will calculate weather based on these data.

Weather forecasting system takes parameters such as temperature, humidity, and wind and will forecast weather based on previous record therefore this prediction will prove reliable. This system can be used in Air Traffic, Marine, Agriculture, Forestry, Military, and Navy etc. Forecasting the temperature and rain on a particular day and date is the main aim of this paper..Our paper is aimed to provide real time weather forecast service at finest granularity level with recommendations. We grab user’s location (longitude, latitude) using GPS data service whenever user requests for our services. Our system will process the users query and will mine the data from our repository to draw appropriate results. Users will be provided with recommendations also and that is the key facility of our service. Personalized forecast is generated for each individual user based on their location.

## 1.2 PROBLEM DEFINITION

System will take current parameters entered by the user and will compare the parameters with the data in database and will predict the Weather. The output is shown in the maps and also a bar graph that is plotted to show the predicted results. Advantages are User can easily find out Weather condition by using this system. The primary advantage of forecasting is that it provides the business with valuable information that the business can use to make decisions about the future of the organization. Disadvantages are Weather forecast by the system is not very accurate. Previous data is required by the system to forecast weather.

## 1.3 OBJECTIVES

Forecasting the temperature and rain on a particular day and date is the main aim of this paper. Our paper is aimed to provide real time weather forecast service at finest granularity level with recommendations. We grab user's location (longitude, latitude) using GPS data service whenever user requests for our services. Our system will process the users query and will mine the data from our repository to draw appropriate results. Users will be provided with recommendations also and that is the key facility of our service. Personalized forecast is generated for each individual user based on their location.

## 1.4 SCOPE OF THE PROJECT

The project mainly focuses on forecasting weather conditions using historical data. This can be done by extracting knowledge from this given data by using techniques such as association, pattern recognition, nearest neighbor etc.

Disaster Mitigation: Predicting storms, floods, droughts

Helping those sectors which are most dependent on whether such as agriculture, aviation also depends on weather conditions.

# CHAPTER 2

## LITERATURE SURVEY

Established literature on customer churn uses various data mining technologies, such as neural networks, clustering, decision tree, regression , support vector machine , and ensemble of hybrid methods to provide more accurate predictions. Regression is the most commonly adopted technique, probably because of its high reported accuracy and interpretability for understanding key drivers, as well as for providing information to set up retention actions.

One of the most common techniques for dealing with rarity is sampling. Methods that adopt the sampling technique alter the distribution of training examples and generate balanced training set(s) for building churn prediction model. However, a recent study on the class imbalance issue in Weather prediction reveals that advanced sampling technique does not increase predictive performance.

Although the weighted Random Forests technique is suggested tree ensembles, such as Random Forests, are often criticized for being hard to interpret i.e., it is difficult to identify risk factors which can be addressed by the retention process to prevent a customer

From leaving, thus they are not the preferred methods in this study. Other research explores the power of new features for churn prediction, such as social network and text information of customer complaints which is beyond the scope of this paper.

The established literature only uses boosting as a general method to boost accuracy, and few researchers have ever tried to take advantage of the weight assigned by boosting algorithms. The weight also provides important information, specifically, outliers.

## 2.1 TECHNOLOGY

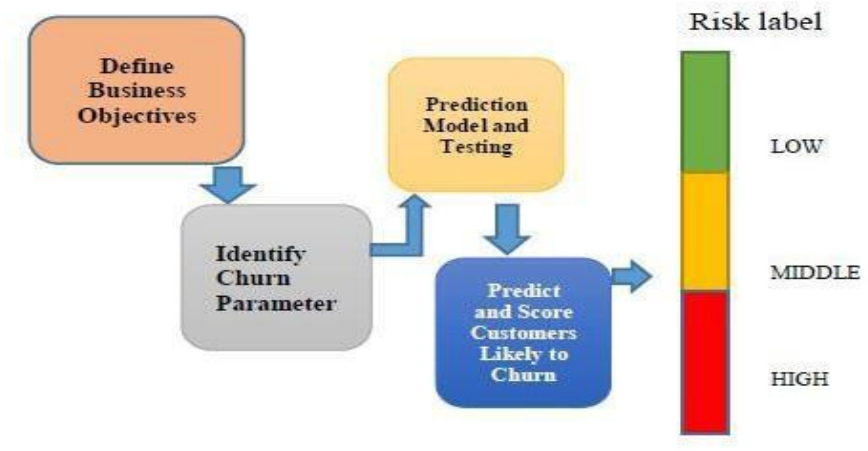

Define Business Objectives
Identify Churn Parameter
Prediction Model and Testing
Predict and Score Customers Likely to Churn
Risk label
LOW
MIDDLE
HIGH

Fig no.2.1: Technology Diagram

This section explains how the Ensemble based Classifiers and well-known Base Classifiers were used in the Churn Prediction Model.

**A. Decision Tree**

Decision Tree was developed to overcome the drawbacks of ID3 algorithm. Utilizes the benefits of greedy approach and uses a series of rules for classification. Although this approach gives a high classification accuracy rate it fails to respond to noisy data. Gain is the main metric used in the decision tree to decide the root node attribute.

**B. Support Vector Machine**

SVM algorithm was proposed by **Boser, Guyon, and Vapnik.** It was very well used for both classification and regression problem. SVM maps all the data points to a higher dimensional plane to make the data points linear separable. The plane which divides data points is known as hyper plane. It can be used for small dataset to give an optimal solution. SVM cannot be more effective for noisy data. SVM model tries to find out the churn and non-churn customer. In order to divide the dataset into churner and non-churner group, first it will take all the data points in n*dimensional* plane and divide the data points into churner and
non-churner group based on maximum marginal hyper plane.

Based on the maximum marginal hyper plane it will divide the data points into churner and non-churner group. Here *n* represents the number of predictor variable associated with the dataset.

***C. Boosting***

Boosting Ensemble technique is designed in such a way that it will maintain a weight for each training tuple. After a classifier is learned from the training tuple, weights are updated for the subsequent classifier. The final Boosted Classifier combines the vote of each individual classifier for prediction to improve the performance of the classifier. In similar to SVM, this model is also not suited for noisy data. The key idea for the customer Churn Prediction using Boosting algorithm is to train a series of classifier simultaneously and keep updating the model accuracy for improving the performance of the classifier.

***D. Random Forest***

Random forest (**Breiman 2001)** works based on the random subspace method. The designed strategy used in Random Forest is divide and conquer. It forms number of Decision Trees and each Decision Tree is trained by selecting any random subset of attribute from the whole predictor attribute set. Each tree will grow up to maximum extent based on the attribute present in the subset. Then after, based on average or weighted average method, the final Decision Tree will be constructed for the prediction of the test dataset. Random forest runs efficiently in large dataset. It can handle thousands of input variables without variable deletion. It also handle the missing values inside the dataset for training the model. It  is  difficult to handle the unbalanced dataset by using Random Forest.

## 2.2 EXISTING SYSTEM

Preparation of All India weather bulletins that includes forecast & warning with graphics along with text up to 3 days for 36 sub-divisions and outlook for subsequent 4 days. It has been issued four times in a day and disseminated to various users like Door Darshan, All India Radio, Press Information Bureau, National Disaster Management Authority, National Disaster Management, Ministry of Home affairs, Ministry of Agriculture etc. This is also post in IMD Website for public in general.

a. Preparation of 5 days forecast of different cities at All India Level, to preparation of this forecast current synoptic observations, satellite imageries and numerical Weather Prediction model outputs taken into consideration.

b. Preparation of Press Releases/Reports on severe weather events. During monsoon season, weekly press release with detail weather of the week and forecast & warning for one week.

c. Monitoring of southwest and northeast monsoon onset, advance, performance and its withdrawal.

d. Co-ordination with sub offices of IMD regarding forecast & warning through video/audio conferences.

e. Collection of meteorological parameters like temperature, rainfall, snowfall, humidity etc. of major cities for internal use and for media briefing etc.

f. To provide the operational forecasting training to national and international forecasters. Also impart internship training to various colleges and university students. In addition also provide briefing to school/college/university students, disaster managers and other various national and international delegates.

## 2.3 PROPOSED SYSTEM

User will enter current temperature; humidity and wind, System will take this parameter and will predict weather from previous data in database. The role of the admin is to add previous weather data in database, so that system will calculate weather based on these data. Weather forecasting system takes parameters such as temperature, humidity, and wind and will forecast weather based on previous record therefore this prediction will prove reliable.

## 2.4 METHODOLOGY

The process we have briefed in earlier section can be depicted pictorially and which is self-explanatory.

We can divide our process in two modules namely:

1) Weather Mining

2) Recommendation

B. Weather Mining:

Data collection: We have collected weather data from WORLD DATA CENTER for climate,Hamburg. We have decided to use NWS API for data collection in future.

Data formatting and cleaning: We have converted our data from .NC (netcdf) format to .CSV (comma-separated values) format because WEKA supports .CSV format.

Clustering: Using WEKA, we have performed clustering on weather data to draw inferences.

Recommendation: We have planned to use recommendation algorithm as user to location collaborative algorithm similar to user to item collaborative algorithm. This algorithm uses user location (N*M) metrics.

## MODULES

**Login: -** Admin will login to the system using ID and Password.

**User Subscription: -** User can access the system using his user ID and password. User can register himself by filling registration form.

**Data Collection:** - Admin will add previous weather data in database. Which is used for clustering the data based on which the prediction is done

**Weather Forecast: -** In this module, system will take current parameters entered by the user and will compare the parameters with the data in database and will predict the weather.

**Visualization of predicted result**: This gives a posturized representation of the predicted weather in the form of maps and graphs.

## 2.5. PRODUCT PERCEPTIVE

User will login to the system using his user ID and password. User will choose a location for which the weather must predicted. System will take this parameter and will predict weather from previous data in database.

The project mainly focuses on forecasting weather conditions using historical data (Dynamic data).This can be done by extracting knowledge from this given data by using techniques such as association, pattern recognition, nearest neighbor etc.

Disaster Mitigation: Predicting storms, floods, droughts helping those sectors which are most dependent on weather such as agriculture, etc.

## 2.6. DESIGN DESCRIPTION

**Admin**: Is a super user who has unlimited privileges. He/she can add a place, delete a place, view a place, view predicted place. He/she has control over what an average user can physically see on their screens therefore admins can also make changes to the website layouts.

**User**: Is an account type who will have limited privileges. He/she can search for a place, check the weather, and view the place for which we want the weather prediction.

**Weather Forecasting system:** This is a advanced system that uses various machine learning techniques and the WEKA tool (which will be mainly used for data mining) to predict the weather for the next day.

## 2.7. DESIGN APPROACH

**Model-View-Controller**

MVC is an application design model comprised of three interconnected parts. They include the

Model (data),

View (user interface).

Controller (processes that handle input).

-Commonly used for developing modern user interfaces.

- It is provides the fundamental pieces for designing a programs for web applications

-It works well with object-oriented programming, since the different models, views,and controllers can be treated as objects and reused within an application.

**Advantages Of MVC**

- Faster Web Application Development Process: .
- MVC Web Application Supports Asynchronous Technique
- Offers The Multiple Views:
- Ideal for developing large size web application: ...
- MVC Model Returns The Data Without The Need of Formatting.

**Disadvantage of MVC**

- Difficulty of using MVC with modern user interface
- Knowledge on multiple technologies is required.
- Developer have knowledge of client side code and html code
- Need multiple programmers.

## RISKS :

## Design Constraints, Assumptions and Dependencies

The working of the application has a limitation to take inputs as few previous days data with the current day and predict the next days climate.

**User constraints :_**Should have to be a registered user. Access to redirect and allows to use the location that is available in out server only. Internet connection to use the application.

**Usage Limitations :** The location can be chosen from the list of places the application is bound to. The specifications to the range of each location is set my the administrator not the user.

# CHAPTER 3

## REQUIREMENT ANALYSIS

### 3.1 FUNCTIONAL REQUREMENTS

In software engineering, a functional requirement defines a function of the software system or its components. A function is described as a set of inputs, the behavior and outputs. Functional requirements maybe calculations, technical details, data manipulation and processing and other specific functionality that define what a system must accomplish. A functionalrequirement defines a function of a software system or its components. It captures the intended behavior of the system. This behavior maybe expressed in terms of services, tasks or functions that the system has to perform.

### 3.2 Non Functional Requirements

Non Functional Requirements specify the criteria that can be used to judge the operation of a system, rather than specific behaviors. They are the metrics that are considered to measure the performance of the developed system. The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates.

Three key considerations involved in the feasibility analysis are

- **Usability:** Simple is the key here. The system must be simple that people like to use it, but not so complex that people avoid using it. The user must be familiar with the user interfaces and should not have problems in migrating to a new system with a new environment. The menus, buttons and dialog boxes should be named in a manner that they provide clear understanding of the functionality.
- **Reliability:** The system should be trustworthy and reliable in providing the functionalities.

- **Performance:** The system is going to be used by many people simultaneously. Performance becomes a major concern. The system should not succumb when many users would be using it simultaneously. It should allow fast accessibility to all of its users.
- **Scalability:** The system should be scalable enough to add new functionalities at a later stage. There should be a common channel, which can accommodate the new functionalities.
- **Maintainability:** The system monitoring and maintenance should be simple and objective in its approach.
- **Portability:** The system should be easily portable to another system.
- **Reusability:** The system should be divided into such modules that it could be used as a part of another system without requiring much of work.
- **Security**: Security is a major concern .This system must not allow unauthorized users to access the information of other users.

## 3.3 DOMAIN AND UI REQUIREMENTS

- **Admin Login: -** Admin will login to the system using Admin ID and Password.
- **Add previous weather data: -** Admin will add previous weather data in database.
- **User Login: -** User can access the system using his user ID and password.
- **User Registration: -** User can register himself by filling registration form.
- **Input Parameters: -** In this module user will enter parameters such as temperature, humidity and wind.
- **Weather Forecast: -** In this module, system will take current parameters entered by the user and will compare the parameters with the data in database and will predict the weather.

## 3.4 HARDWARE AND SOFTWARE REQUIREMENTS

### 3.4. HARDWARE REQUIREMENTS:

The hardware requirements listed below are almost in a significantly higher level which represents the ideal situations to run the system. Following are the system hardware requirements used :

| | | |
|---|---|---|
| Processor | - | Pentium –III |
| Speed | - | 1.1 GHz |
| RAM | - | 256 MB (min) |
| Hard Disk | - | 20 GB |

### 3.5. SOFTWARE REQUIREMENTS:

A major element in building a system is a section of compatible software since the software in the market is experiencing in geometric progression. Selected software should be acceptable by the firm and the user as well as it should be feasible for the system. This document gives the detailed description of the software requirements specification. The study of requirement specification is focused specially on the functioning of the system. It allows the analyst to understand the system, functions to be carried out and the performance level which has to be maintained including the interfaces established.

Operating System -

WindowsApplication Server -

Tomcat

Front End - HTML, Java, Jsp

Scripts - JavaScript.

Server side Script - Java Server

Pages.Database - Mysql

Database Connectivity - JDBC.

## 3.6. DATA REQUIREMENTS:

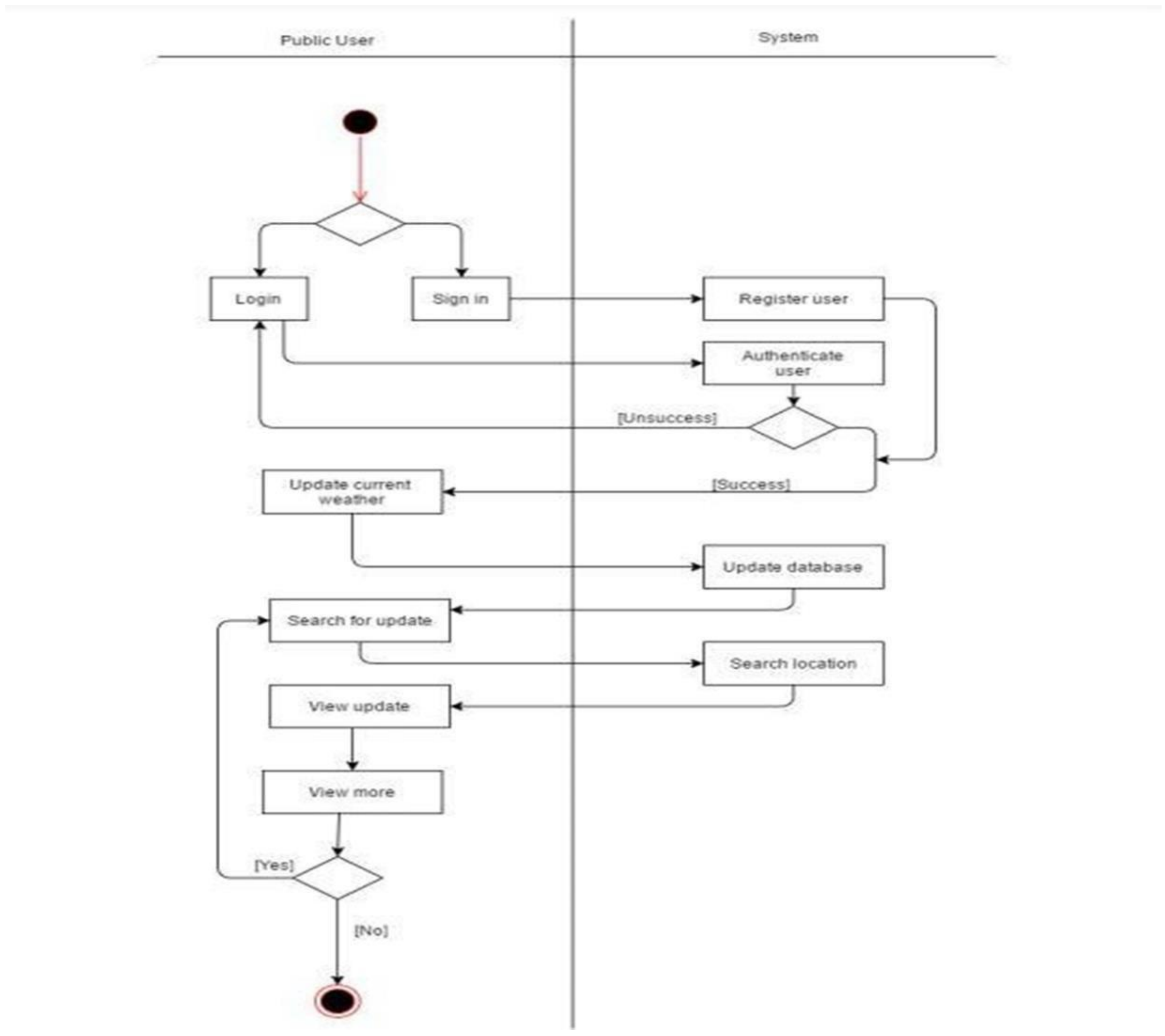
**Data collection**: We have collected weather data from WORLD DATA CENTER for climate, Hamburg. We have decided to use NWS API for data collection in future.

**Data formatting and cleaning:** We have converted our data from .NC (netcdf) format to .CSV (comma-separated values) format because WEKA supports .CSV formats.

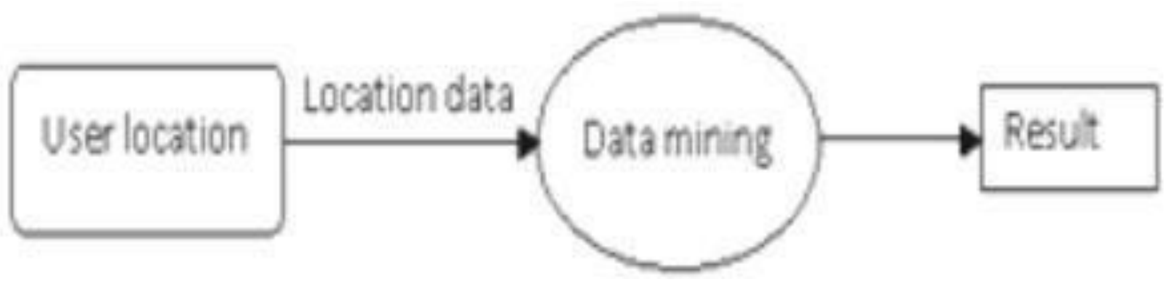
# CHAPTER 4

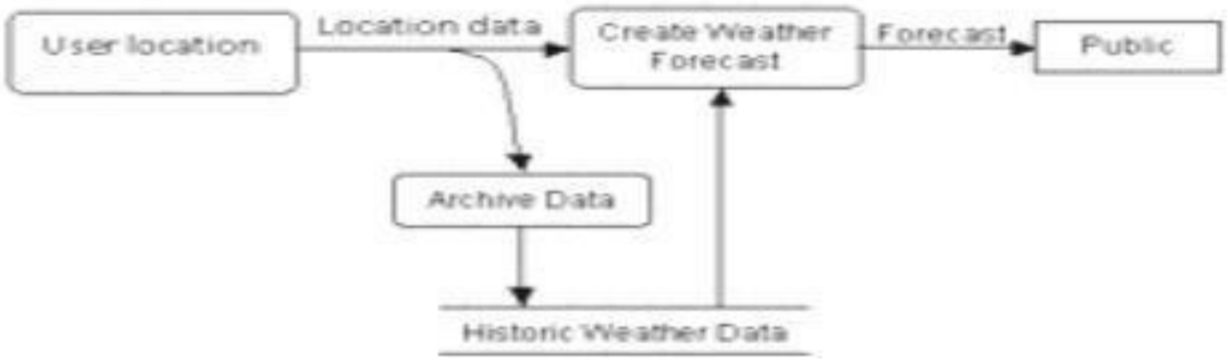
## DESIGN

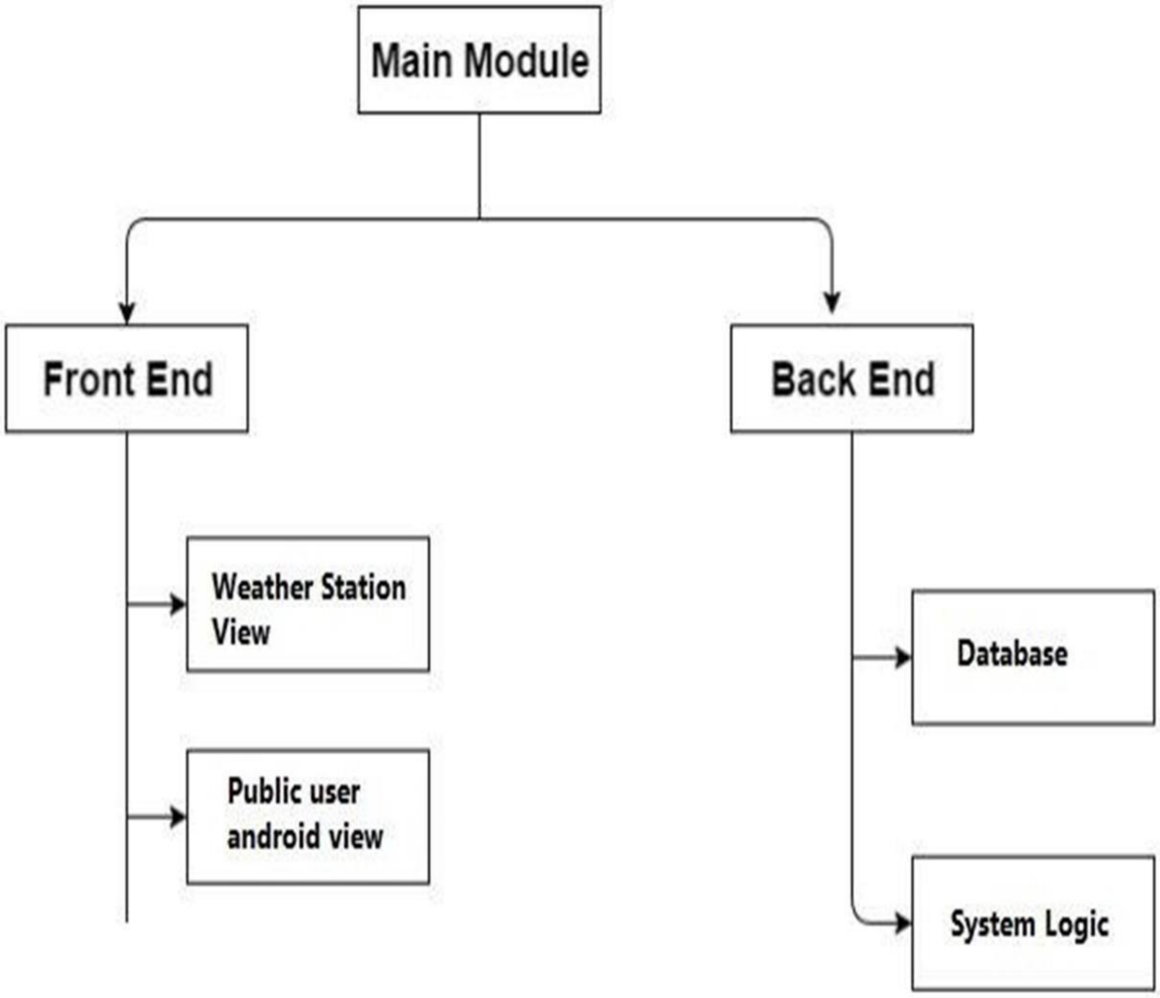
### 4.1 DESIGN GOALS



## 4.2 DATA FLOW DIAGRAM/ ACTIVITY DIAGRAM




User location
Location data
Create Weather Forecast
Forecast
Public
Archive Data
Historic Weather Data


Main Module
Front End
Back End
Weather Station View
Public user android view
Database
System Logic

A **data flow diagram (DFD)** is a graphical representation of the flow of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. This context-level DFD is then exploded to show more detail of the system being modelled.

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system. The DFD is simple graphicalformalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data and the output data generated by the system.

A DFD model uses a very limited number of primitive symbols to represent the functionsperformed by a system and the data flow among the functions. The main reason why the DFD technique is so popular is probably because of the fact that DFD is a very simple formalism- It is simple to understand and use. Starting with the set of high level functions that a systemperforms, a DFD model hierarchically represents various sub functions. In fact, any hierarchical model is simple to understand. The human mind is such that it can easily understand any hierarchical model of a system because in a hierarchical model, starting with a very simple and abstract model of system, different details of a system are slowly introduced through the

Different hierarchies. The data flow diagramming technique also follows a simple set of intuitive concepts and rules.

DFD is an elegant modeling technique that turns out to be useful not only to represent the results of a structured analysis of a software problem but also for several other applications such as showing the flow of documents or items in an organization. A data-flow diagram (DFD)is a graphical representation of the "flow" of data through an information system. DFDs canalso be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

**Level 0:** A context-level or level 0 data flow diagram shows the interaction between the system and external agents which act as data sources and data sinks. On the context diagram (also known as the Level 0 DFD) the system's interactions with the outside world are modeled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization

**Level 1:** The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, Level 1 Data Flow diagram shows an in depth explanation of overall process of the data flow.

**Level 2:** Complete details of every function performed by Admin. Every step is explained in detail.

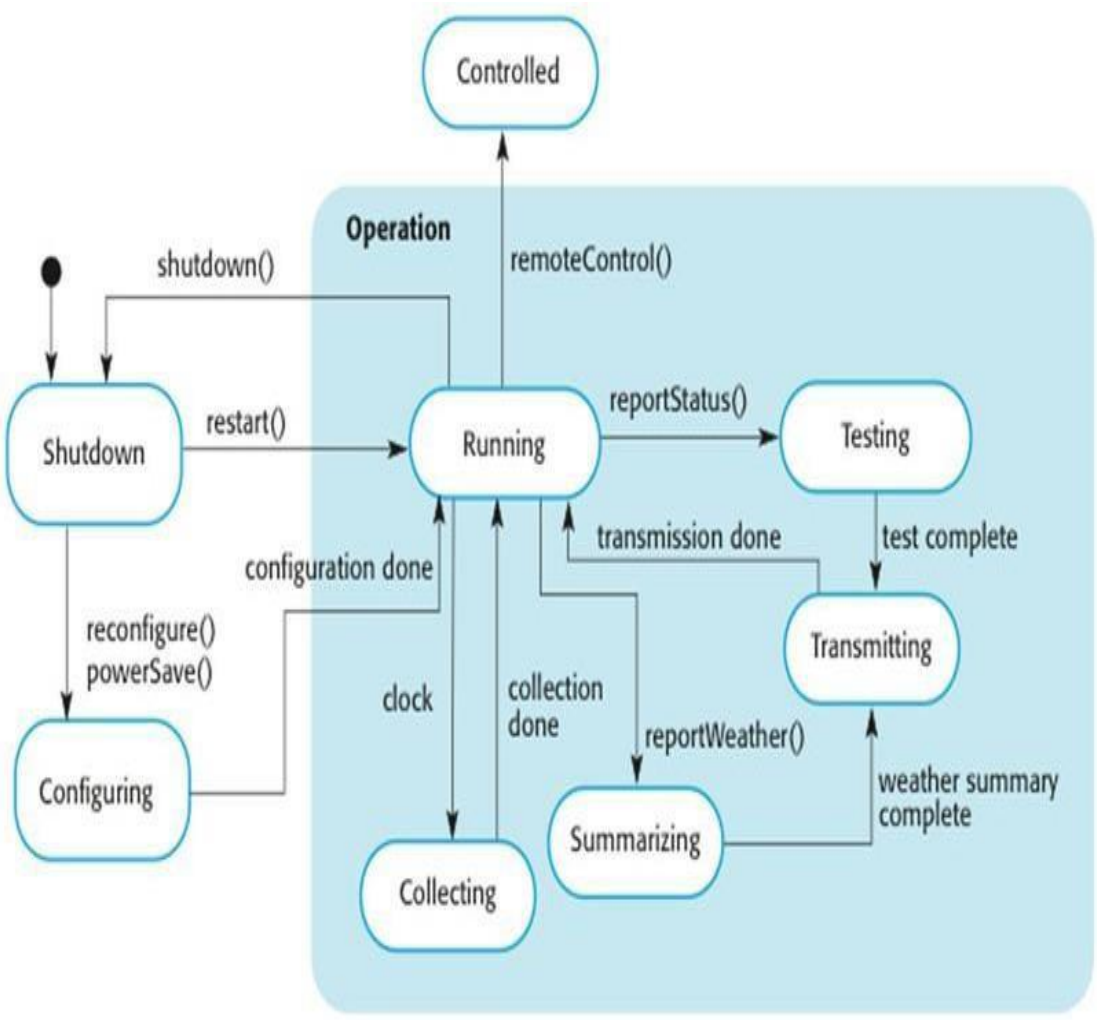
The four components of a data flow diagram (DFD) are:

External Entities/Terminators are outside of the system being modelled. Terminators represent where information comes from and where it goes. In designing a system, we have no idea about what these terminators do or how they do it. Processes modify the inputs in the process of generating the outputs

Data Stores represent a place in the process where data comes to rest. A DFD does not say anything about the relative timing of the processes, so a data store might be a place to accumulate data over a year for the annual accounting process.

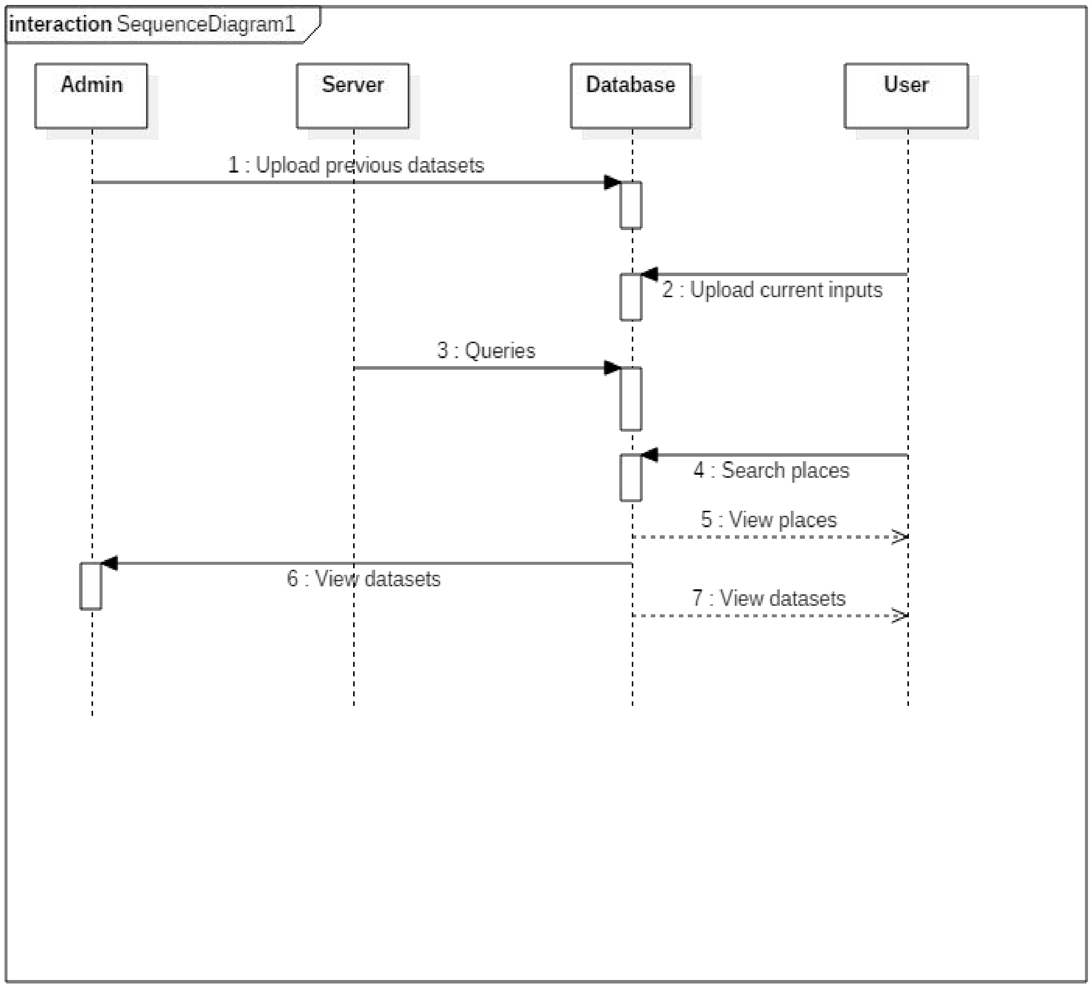
Data Flows shows how data moves between terminators, processes, and data stores (those that cross the system boundary are known as IO or Input Output Descriptions).

## 4.3 STATE MACHINE UML DIAGRAM

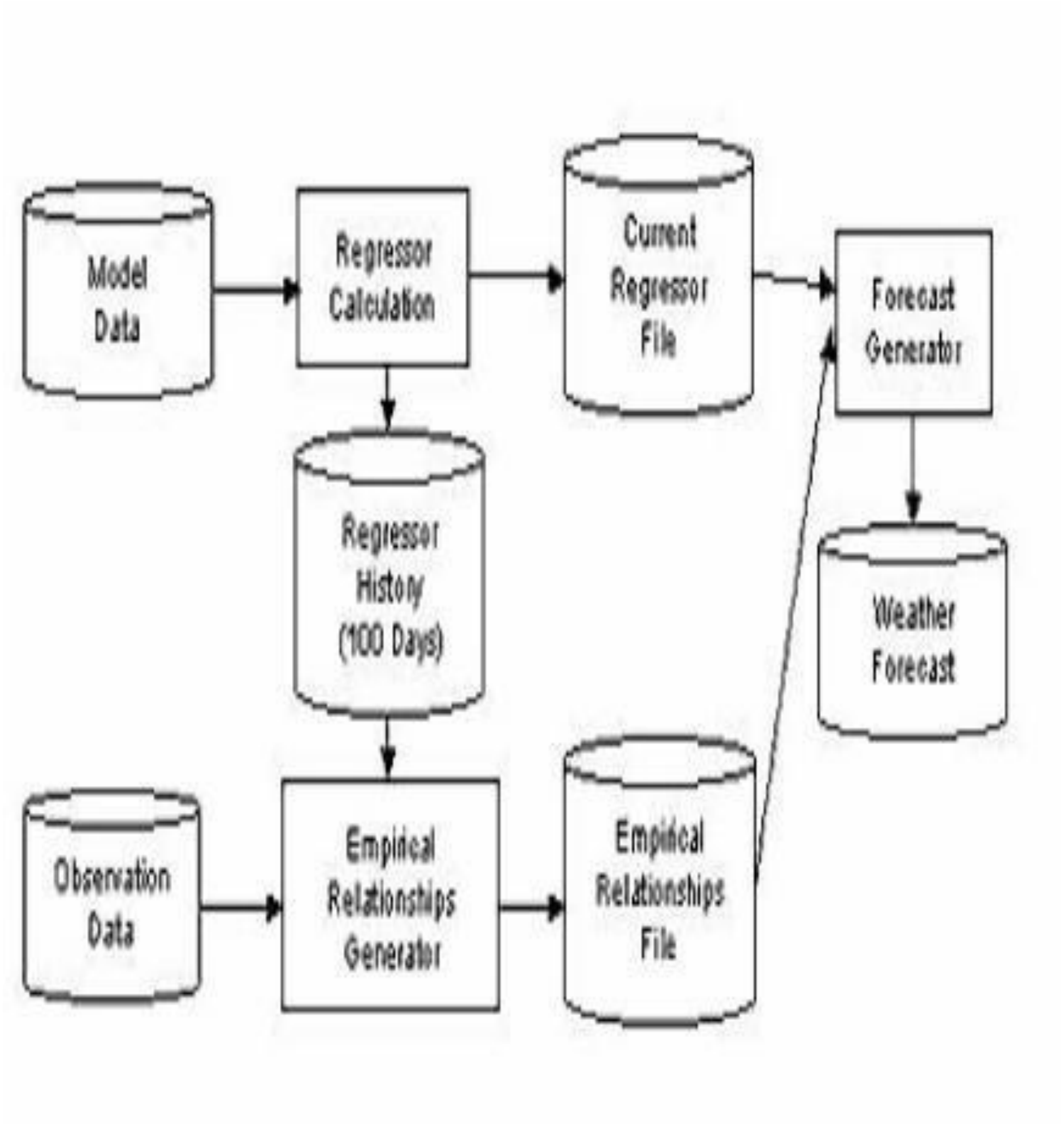


## 4.4 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Chart. Sequence diagrams are sometimes called event diagrams, event scenarios and the sequence diagram for this project is given in figure



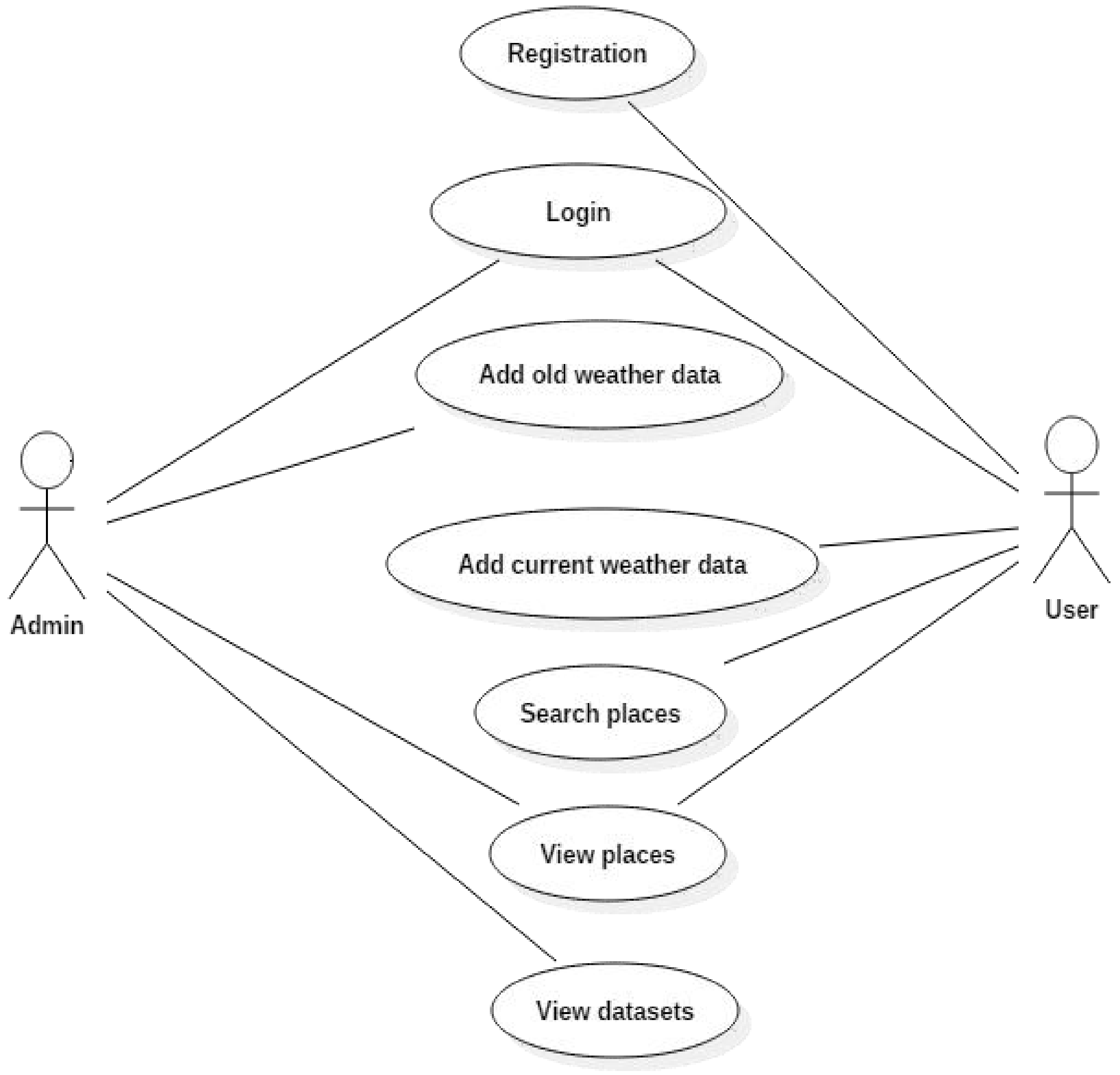
## 4.5 INTERACTION OVERVIEW DIAGRAM



## 4.6 USE CASE DIAGRAM

A use case diagram is a type of behavioral diagram created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. Use case diagram provides us the information about how that users and use cases are related

with the system.



## 4.7 ALGORITHM/PSEUDOCODE

**Data collection and processing**

```
def extract_weather_data(url, api_key, target_date, days):
    records = []
    for _ in range(days):
        request = BASE_URL.format(API_KEY,
        target_date.strftime('%Y%m%d'))response = requests.get(request)
        if response.status_code == 200:
            data = response.json()['history']['dailysummary'][0]
            records.append(DailySummary(
                date=target_date, meantempm=data['meantempm'],
                meandewptm=data['meandewptm'],
                meanpressurem=data['meanpressurem'],
                maxhumidity=data['maxhumidity'],
                minhumidity=data['minhumidity'],
                maxtempm=data['maxtempm'],
                mintempm=data['mintempm'],
                maxdewptm=data['maxdewptm'],
                mindewptm=data['mindewptm'],
                maxpressurem=data['maxpressurem'],
                minpressurem=data['minpressurem'],
                precipm=data['precipm']))
            time.sleep(6)
            target_date += timedelta(days=1)
        return records
if not station_id:
            raise Exception("'station_id' is required.")
        if 'recordId' in params and 'current' in params:
            raise Exception("Cannot have both 'current' and 'recordId'")
        if 'start' in params:
            start = params['start']
            self.parse_param_timestamp(start)if
            len(start) < 19:
                start =
            '{}T00:00:00Z'.format(start[:10])elif
            len(params['start']) < 20:
                start = start.replace(' ', 'T')
                start = '{}Z'.format(start)
            params['start'] = start
        if 'end' in params:
            end = params['end']
            self.parse_param_timestamp(end)if
            len(end) < 19:
                end =
            '{}T23:59:59Z'.format(end[:10])elif
```

```
len(params['end']) < 20:
        end = end.replace(' ', 'T')
        end =
    '{}Z'.format(end)
    params['end'] = end

request_uri = "/stations/{stationId}/observations".format(
    stationId=station_id)

if len(params) > 0:
    if 'recordId' in params:
        return self.make_get_request(
            '{old_request_uri}/{recordId}'.format(
                old_request_uri=request_uri,
                recordId=params['recordId']),
            end_point=self.DEFAULT_END_POI
    NT)if 'current' in params:
        return self.make_get_request(
            '{old_request_uri}/current'.format(
                old_request_uri=request_uri),
            end_point=self.DEFAULT_END_POINT)

    if len(params) > 1:
        request_uri = '{old_request_uri}?{query_string}'.format(
            old_request_uri=request_uri,
                query_string=urlencode(params))


            observations = self.make_get_request(
              request_uri,
          end_point=self.DEFAULT_END_POINT)if 'features'
          not in observations:
              raise Exception(observations)
          return observations['features']


        return self.make_get_request(
"/stations/{stationId}/observations".format(stationId=station_id),
          end_point=self.DEFAULT_END_POINT)


    def products(self, id):
        functional tests @todo(paulokuong) later on.
  Args:
          id (str): product id.
        Returns:
          json: json response from api.
"""
        return self.make_get_request(
"/products/{productId}".format(productId=id),
          end_point=self.DEFAULT_END_POINT)
    def products_types(self, **params),
      Response in this method should not be modified.
        In this way, we can keep track of changes made by NOAA
        throughfunctional tests @todo(paulokuong) later on.
   Args:
          type_id (str): an id of a valid product type
          locations (boolean[optional]): True to get a list of
              locations that have issues products for a type.
          location_id (str): location id.
        Returns:
          json: json response from api.
        if 'type_id' in params and 'locations' not in params:return
          self.make_get_request(
"/products/types/{type_id}".format(type_id=params['type_id']),
              end_point=self.DEFAULT_END_POINT)
        elif 'locations' in params:
          if 'type_id' not in params:
              raise Exception('Error: Missing type id (type_id=None)')
          if 'location_id' in params:
              return self.make_get_request(
                ('/products/types/{type_id}/locations/'
                '{location_id}').format(
                  type_id=params['type_id'],
                  location_id=params['location_id']),
               end_point=self.DEFAULT_END_POINT)
         else:
            return self.make_get_request(
"/products/types/{type_id}/locations".format(
                  type_id=params['type_id']),
               end_point=self.DEFAULT_END_POINT)
      return self.make_get_request(
"/products/types",
         end_point=self.DEFAULT_END_POIN
 T)def products_locations(self, **params)
      functional tests @todo(paulokuong) later on.
Args:
         location_id (str): location id.
      Returns:
         json: json response from api.
      if 'location_id' in params:
         return self.make_get_request(
"/products/locations/{locationId}/types".format(
               locationId=params['location_id']),
            end_point=self.DEFAULT_END_POINT)
      return self.make_get_request(
"/products/locations",
      end_point=self.DEFAULT_END_POIN
 T)def offices(self, office_id):
    Response in this method should not be modified.
    In this way, we can keep track of changes made by NOAA
    throughfunctional tests @todo(paulokuong) later on.
Args:
      office_id (str): office id.
    Returns:
      json: json response from api.
    return self.make_get_request("/offices/{office_id}".format(
      office_id=office_id), end_point=self.DEFAULT_END_POINT)
 def zones(self, type, zone_id, forecast=False):
    functional tests @todo(paulokuong) later on.
 Args:
      type (str): a valid zone type (list forthcoming).
      zone_id (str): a zone id (list forthcoming).
      forecast (bool): True to show forecast data of the zone.
    Returns:
      json: json response from api.
    if forecast:
      return self.make_get_request(
"/zones/{type}/{zone_id}/forecast".format(
          type=type, zone_id=zone_id),
        end_point=self.DEFAULT_END_POINT)
    return self.make_get_request("/zones/{type}/{zone_id}".format( type=type,
      zone_id=zone_id), end_point=self.DEFAULT_END_POINT)
  def alerts(self, **params):
    Response in this method should not be modified.
    In this way, we can keep track of changes made by NOAA
    throughfunctional tests @todo(paulokuong) later on.
Args:
      alert_id (str): alert id.
      active (int): Active alerts (1 or 0).
      start (str): Start time (ISO8601DateTime).
      end (str): End time (ISO8601DateTime).
      status (str): Event status (alert, update, cancel).
      type (str): Event type (list forthcoming).
      zone_type (str): Zone type (land or marine).
      point (str): Point (latitude,longitude).
      region (str): Region code (list forthcoming).
      state (str): State/marine code (list forthcoming).
      zone (str): Zone Id (forecast or county, list forthcoming).
      urgency (str): Urgency (expected, immediate).
      severity (str): Severity (minor, moderate, severe).
      certainty (str): Certainty (likely, observed).
      limit (int) Limit (an integer).
```

```
        Returns:

            json: json response from api.

       if 'alert_id' in params:

            return self.make_get_request(

"/alerts/{alert_id}".format(alert_id=params['alert_id']),

                end_point=self.DEFAULT_END_POINT

        )return self.make_get_request(

"/alerts?{query_string}".format(query_string=urlencode(params)),

            end_point=self.DEFAULT_END_POINT)

  def active_alerts(self, count=False, **params):

        functional tests @todo(paulokuong) later on.

   Args:

            count (bool): True to hit /alerts/active/count. zone_id

            (str): a valid zone, see list in counts endpoint.area

            (str): a valid area, see list in counts endpoint. region

            (str): a valid region, see list in counts endpoint

        Returns:

            json: json response from api.

        if count:

            return self.make_get_request(

"/alerts/count",

                end_point=self.DEFAULT_END_POI

        NT)if 'zone_id' in params:
            return self.make_get_request(
"/alerts/active/zone/{zoneId}".format(
                    zoneId=params['zone_id']),
                end_point=self.DEFAULT_END_POINT)
        if 'area' in params:
            return self.make_get_request(
"/alerts/active/area/{area}".format(
                    area=params['area']),
                end_point=self.DEFAULT_END_POINT
                )
        if 'region' in params:
            return self.make_get_request( "/alerts/active/region/{region}".format(
                    region=params['region']),
                end_point=self.DEFAULT_END_POINT
```

# CHAPTER

# IMPLEMENTATION

## FUNCTIONALITY

We can divide our process in two modules namely:

1) Weather Mining

2) Recommendation

B. Weather Mining:

Data collection: We have collected weather data from WORLD DATA CENTER for climate, Hamburg. We have decided to use NWS API for data collection in future.

Data formatting and cleaning: We have converted our data from .NC (netcdf) format to .CSV (comma-separated values) format because WEKA supports .CSV format.

Clustering: Using WEKA, we have performed clustering on weather data to draw inferences.

Recommendation: We have planned to use recommendation algorithm as user to location collaborative algorithm similar to user to item collaborative algorithm. This algorithm uses user location (N*M) metrics.

Visualization: To generate visualization for user, we have used NOAA weather and climate tool kit.

For the part of the implementation, on which your project focused most, which algorithms you implemented or used and if any modifications were needed to those algorithms or if you did some initial pre-processing, discuss here For the other phases of data mining, discuss brief. E.g., if you focused most on visualization, you can talk about: which data (Example: downloaded from some website put the URL here; did some survey, then talk about how you did the survey etc.) collection approach was used in the project?

*C. Recommendation*

Extract the location of the user. Extract the destination of the user and then recommend the best path according to the conditions.

# CHAPTER

# TESTING

## 6.1. TEST STRATEGY

The test for predictive accuracy has been used widely and adapted for economic forecasts, but has not seen much activity in weather forecast verification. The technique is applied to both simulated verification sets as well as  weather data at eight stations in Utah, and a loss function based on dynamic time warping (DTW) is used. Results of the simulation experiment show that the DTW  technique can be useful if timing errors are the concern.

The project mainly focuses on forecasting weather conditions using historical  data. This can be done by extracting knowledge from this given data by using techniques such as association, pattern recognition, nearest neighbor etc.

The testing is completely based on the how accurate the weather conditions are retrieved. The other parameters that the testing is considered over functional testing

**Admin**

- To check for proper locations to be added with their latitude and longitude.
- To check if there is unique generated id for each location
- To check the database as and when the place is added or deleted
- To check weather the previous predicted places are shown Etc.

**User**

- check for the user and password to be valid.
- check the location should be selected from the drop down that is updated by the admin.
- check user location once loaded should show the current weather.
- check predicted weather should be graphically representation.

***Recommendation***

- Extract the location of the user
- Extract the destination of the user
- And then recommend the best path according to the conditions.

**Test Tools Used**

**Forecasting a Continuum of Environmental Threats (FACETs)**

A Tool to proposed next-generation severe weather watch and warning framework that is modern, flexible, and designed to communicate clear and simple hazardous weather information to serve the public.

## RISK IDENTIFICATION AND CONTIGENCY PLANNING

This section identifies the risks that are associated during the testing

| **Risk #** | **Risk** | **Nature of Impact** | **Contingent Plan** |
|---|---|---|---|
| **1** | Mapping of day to day data | No output generated | **Proper data to be checked and updated on the current weather field** |
| **2** | Data Collection with specific parameters | Unexpected termination | **Creating a new dataset with the required parameters** |

## 6.2. CCEPTANCE CRITERIA

This application must serve the purpose by predicting the next day weather which should be shown as a graph with has to be generated with the increase in humidity anddegree of the location requested by the user.

**Test Case List**

This section shall clearly define the test cases that are planned for testing. The following information shall be mentioned: -

| TestCase Number | Test Case | Required Output |
|---|---|---|
| 1 | Enters the latitude and longitudeof each place | Successful update of the table place |
| 2 | Generation of User ID. | Unique id generated for each user |
| 3 | Auto generation of Place ID. | Each place has its own id . |
| 4 | The predict details page for previous weather Condition. | Previous weather data updated |
| 5 | View place on the places table | Check for the added new place |
| 6 | Predict for next day | Should provide a graph with humidity and degree generated |

| | | |
|---|---|---|
| 7 | Predict check for previous days | Should match almost the values that has been stored on the data set |

## Results and Discussions

This system is a mobile app application with an effective graphical user interface that is capable of predicting weather based on parameters such as temperature, pressure and rain. The user enters the temperature, pressure and rain and must enter the next day to get the accurate prediction. The location and current weather is mapped and shown in the Google maps using an API and also a bar graph is plotted to show the predicted results.

Two graphs are humidity and temperature tomorrow. The left side graph shows the humidity and the right side graph shows the temperature. The user can understand the graph by seeing the difference of increase and decrease in temperature and humidity compared to previous day.

# CHAPTER 8

## CONCLUSION

Traditionally, weather forecasting has always been performed by physically simulating the atmosphere as a fluid and then the current state of the atmosphere would be sampled. In the previous system the future state of the atmosphere is computed by solving numerical equations of thermodynamics. But this model is sometimes unstable under disturbances and uncertainties while measuring the initial conditions of the atmosphere. This leads to an incomplete understanding of the atmospheric processes, so it restricts weather prediction.

Our proposed solution of using Machine learning for weather predicting is relatively robust to most atmospheric disturbances when compared to traditional methods. Another advantage of using machine learning is that it is not dependent on the physical laws of atmospheric processes. In the long run weather prediction using Machine Learning has a lot of advantages and thus it should be used globally.

# CHAPTER 9

## FURTHER ENHANCEMENTS

Weather forecasts are becoming more detailed, more accurate and are providing the information needed to make sound decisions to protect life and property. Technological advances, such as apps, are making weather information more accessible and immediately alerting those in harm’s way.

The current version of Weather Prediction that we have developed is still premature. This implies that there are still many limitations that can be resolved and improved. One of the biggest limitation right now is, that the location has to be chosen from the list of places the application is bound to. This can be improved if we use web scraping tools to automatically get the weather data, for various locations, from the internet and then input it into the database. Another enhancement that can be done is the automatic validation of longitude and latitude coordinates. Another improvement that can be done is to beautify the UI to make it more appealing to the younger generation.

Future enhancements will make our Weather Prediction more flexible, user friendly and thus it will be more appealing to a wide range of audience.

# REFERENCES

1. A. Payne and P. Frow, “A strategic framework for customer relationship management,” *J. Marketing*, vol. 69, no. 4, pp. 167–176, Oct. 2005.
2. L. D. Xu, “Enterprise systems: State-of-the-art and future trends,” *IEEE Trans. Ind. Inf.*, vol. 7, no. 4, pp. 630–640, Nov. 2011.
3. A. Berson, K. Thearling, and S. Smith, *Building Data Mining Applications for CRM*. New York: McGraw-Hill, 1999.
4. K. Coussement and D. V. Poel, “Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques,” *Expert Syst. Appl.*, vol. 34, no. 1, pp. 313–327, Jan. 2008.
5. W. Verbeke, K. Dejaeger, D. Martens, J. Hur, and B. Baesens, “New insights into churn prediction in the telecommunication sector: A profit driven data mining approach,” *Eur.J. Oper. Res.*, vol. 218, no. 1, pp. 211–229, Apr. 2012.