REAL TIME WEATHER FORECASTING

A PROJECT REPORT

Submitted By

YASHI JAIN (2000290140142)
PRAKSHA TANDON (2000290140089)
ASHISH NEHRA (2000290140029)

Submitted in partial fulfillment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Ms. Shalika Arora
Associate Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad Uttar Pradesh-201206
(JUNE 2022)

Declaration

We hereby declare that the work presented in this report entitled “REAL TIME WEATHER FORECASTING", was carried out by us. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not our original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources. We affirm that no portion of our work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, we shall be fully responsible and answerable.

Name: - YASHI JAIN
Roll no: - 2000290140142

Name: - PRAKSHA TANDON
Roll no: - 2000290140089

Name: - ASHISH NEHRA
Roll no: - 2000290140029

CERTIFICATE

Certified that **Yashi Jain (2000290140142), Praksha Tandon (2000290140089), Ashish Nehra (2000290140029)** have carried out the project work having "**Real Time Weather Forecasting**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Yashi Jain (2000290140142)

Praksha Tandon (2000290140089)

Ashish Nehra (2000290140029)

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

Date:

Ms. Shalika Arora
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Ajay Shrivastava
Head, Department of Computer Applications KIET Group of Institutions, Ghaziabad

ABSTRACT

Weather forecasting is the attempt by meteorologists to predict the weather conditions at some future time and the weather conditions that may be expected. The climatic condition parameters are based on the temperature, wind, humidity, rainfall, felt temperature , pressure ,day temperature , evening temperature, night temperature, maximum temperature , minimum temperature and size of data set. Here, the parameters temperature and Humidity only are considered for experimental analysis.

The report has been made in attempt to provide user with real time weather forecasting with 8 days in future forecasting. For making this project we have studied various concepts related to the weather app and how different factors affect weather app. We also studied about various API Application Programming Interfaces that can be used to provide weather data in easier manner. The project aims at providing most details like humidity, air speed, and many other features.

Weather warnings are a special kind of short-range forecast carried out for the protection of human life. Weather warnings are issued by the governments throughout the world for all kinds of threatening weather events including tropical storms and tropical cyclones depending upon the location. The forecast may be short-range or Long-range. It is a very interesting and challenging task. This report provides a basic understanding of the purpose and scope of weather forecasts, the basic principles and the general models developed for forecasting.

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **MS. SHALIKA ARORA** for her guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Yashi Jain (2000290140142)

Praksha Tandon (2000290140089)

Ashish Nehra (2000290140029)

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Project Description

Weather forecasting is the application of science and technology to predict the conditions of the atmosphere for a given location and time. People have attempted to predict the weather informally for country and city and formally since the 19th century. Weather forecasts are made by collecting quantitative data about the current state of the atmosphere and using meteorology to project how the atmosphere will change at a given place.

Once calculated manually based mainly upon changes in biometric pressure, current weather conditions, and sky condition or cloud cover, weather forecasting now relies on Computer based model that take many atmospheric factors into account. The inaccuracy of forecasting is due to the chaotic nature of the atmosphere, the massive computational power required to solve the equations that describe the atmosphere, the land, and the ocean, the error involved in measuring the initial conditions, and an incomplete understanding of atmospheric and related processes.

The basis for weather prediction started with the theories of the ancient Greek philosophers and continued with Renaissance scientists. It was followed by the scientific revolution of the 17th and 18th centuries. The theoretical models of 20th- and 21st-century atmospheric scientists and meteorologists helped for the betterment in applications. The so-called synoptic weather map came to be the principal tool of 19th-century meteorologists. This is used today in weather stations and on television weather reports all over the world. All can happen only through a comprehensive weather forecast. Any weather prediction needs a systematic collection of weather record of various places and proper analysis using the data for prediction.

1.2 Project Objective

The main objective behind the project **REAL TIME WEATHER FORECATING** is to determine the atmospheric condition of a any city or country on the basis of the following parameters and some of them are listed below,

- Humidity
- Temperature
- Wind
- Pressure
- Day Temperature

  Also, we can determine the atmospheric condition of a city by taking current location into consideration. In our project we add an extra feature of google map by using that find a direction and state of enter city in the search location box. And in the current location, using map we find the direction, state, pin code of the current location.

1.3 Project Scope

There are several reasons why weather forecasts are important. They would certainly be missed if they were not there. It is a product of science that impacts the lives of many people. The following is a list of various reasons why weather forecasts are important

- Helps people prepare for how to dress (i.e., warm weather, cold weather, windy weather, rainy weather)
- Helps people prepare if they need to take extra gear to prepare for the weather (i.e. umbrella, raincoat, sunscreen)
- Helps people plan outdoor activities (i.e., to see if rain/storms/cold weather will impact outdoor event)

- Helps curious people to know what sort of weather can be expected (i.e., a snow on the way, severe storms)
- Helps businesses plan for transportation hazards that can result from the weather (i.e., fog, snow, ice, storms, clouds as it relates to driving and flying for example)
- Helps people with health-related issues to plan the day (i.e., allergies, asthma, heat stress)
- Helps businesses and people plan for severe weather and other weather hazards (lightning, hail, tornadoes, hurricanes, ice storms)9. Helps farmers and gardeners plan for crop irrigation and protection (irrigation scheduling, freeze protection)

1.4 Hardware/ Software Requirements

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. requirement

Table.1 Hardware Requirements

Processor	Pentium and above
Speed	1.1 GHz
RAM	512 MB (min)
Hard Disk	20GB
Keyboard	Standard Windows Keyboard
Mouse	Two or Three Button Mouse
Monitor	SVGA

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 must be maintained including the interfaces established.

Weather Forecasting Using HTML, CSS, JavaScript

Table 2 Software Requirements

Operating System	Windows/Android
Front End	React JS
IDE	Visual Studio
Application	Browser

1.5 Problem and Existing Technology

The traditional forecast process employed by most NMHSs involves forecasters producing text-based, sensible, weather-element forecast products (e.g. maximum/minimum temperature, cloud cover) using numerical weather prediction (NWP) output as guidance. The process is typically schedule-driven, product-oriented and labour-intensive. Over the last decade, technological advances and scientific breakthroughs have allowed NMHSs' hydrometeorological forecasts and warnings to become much more specific and accurate.

As computer technology and high-speed dissemination systems evolved (e.g. Internet), National Weather Service (NWS) customers/partners were demanding detailed forecasts in gridded, digital and graphic formats. Traditional NWS text forecast products limit the amount of additional information that can be conveyed to the user community. The concept of digital database forecasting provides the capability to meet customer/partner demands for more accurate, detailed hydrometeorological forecasts. Digital database forecasting also offers one of the most exciting opportunities to integrate PWS forecast dissemination and service delivery, which most effectively serves the user community.

1.6 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.

1.7 Advantages of Weather Forecast

- Farmers can know when to plant or harvest their crops
- People cam choose where and when to take their holidays to take advantages of good weather
- Surfers known when large waves are expected
- Regions can be evacuated if hurricanes or floods are expected
- Aircraft and shipping rely heavily on accurate weather forecasting

1.8 Disadvantages of Weather Forecast

- Weather is extremely difficult to forecast correctly
- It is expensive to monitor-so many variables from so many sources
- The computers needed to perform the millions of calculations necessary are expensive
- The weather forecasters get blamed if the weather is different from the forecast

CHAPTER 2

FEASIBILITY STUDY

1. Feasibility Study

Feasibility study is used to check the visibility of the project under the consideration of the theoretical project and its various types of feasibility are conducted below are important which are represented are as follows: -

1.1. Technical Feasibility

Technical Feasibility determines whether the work for the project can be done with the existing equipment's software technology and available personals technical feasibility is concerned with the specified equipment and software that will satisfy the user requirement this project is feasible on technical remark also as the proposed system is beneficiary in term of having a soundproof system with new technical equipment's install on the system that proposed system can run on any machine supporting window and work on the best software and hardware that had been used while designing the system so it would be visible in all technical term of feasibility.

1.2 Economic Feasibility

Proposed System requires development tools and software such as visual studio code and python package which are free of cost and available on internet. For developing proposed system, we need various resources such as computers systems, internet connection for e-help, recommended disk space, and memory speed as mention in technical requirement. By looking at all these expenses and comparing with proposed system, we have many benefits from proposed system such are: As existing system is manual, where data may not accurate, up to date, and available on time. But proposed system will be computerized, so we can overcome all limitations of existing system. Also with this new system insertion, deletion, and modification of various data will be easier to handle. This system will reduce the paperwork. And quality of data will be improved. So keeping all above mentioned benefits and comparing with various expenditures of resources, we conclude that proposed system is economical feasible

1.3. Operational Feasibility

The new solution is possible in all sense but operationally is not the new system demands the explosion of at least 15 people from the company it creates an environment of joblessness and fear among the employees it can lead to indefinite Strike in the company also, so the management must take corrective action prior in advance to start further proceeding.

1.4. Behavioral Feasibility

Behavioral Feasibility is the measure of how the society is looking towards our project, what is the reaction of people who are going to use this in upcoming future.

It includes how strong the reaction of user will be towards the development of new system that involves device use in their daily life for connecting with faculties at college

1.5. Modules Description

In this project we have Two modules

1) Data gathering and pre - processing.
2) Applying Algorithm for prediction.

Explanation:

1) In this module we first gather the data(dataset) for our prediction model. Data comes in all forms, most of it being very messy and unstructured. They rarely come ready to use. Datasets, large and small, come with a variety of issues- invalid fields, missing and additional values, and values that are in forms different from the one we require. In order to bring it to workable or structured form, we need to “clean” our data, and make it ready to use. Some common cleaning includes parsing, converting to one-hot, removing unnecessary data, etc.

In our case, our data has some days where some factors weren’t recorded. And the rainfall in cm was marked as T if there was trace precipitation. Our algorithm requires numbers, so we can’t work with alphabets popping up in our data. so we need to clean the data before applying it on our model.

2) Once the data is cleaned, In this module that cleaned data can be used as an input to our Linear regression model. Linear regression is a linear approach to form a relationship between a dependent variable and many independent explanatory variables. This is done by plotting a line that fits our scatter plot the best, ie, with the least errors. This gives value predictions, ie, how much, by substituting the independent values in the line equation.

We will use Scikit-learn’s linear regression model to train our dataset. Once the model is trained, we can give our own inputs for the various columns such as temperature, dew point, pressure, etc. to predict the weather based on these attributes.

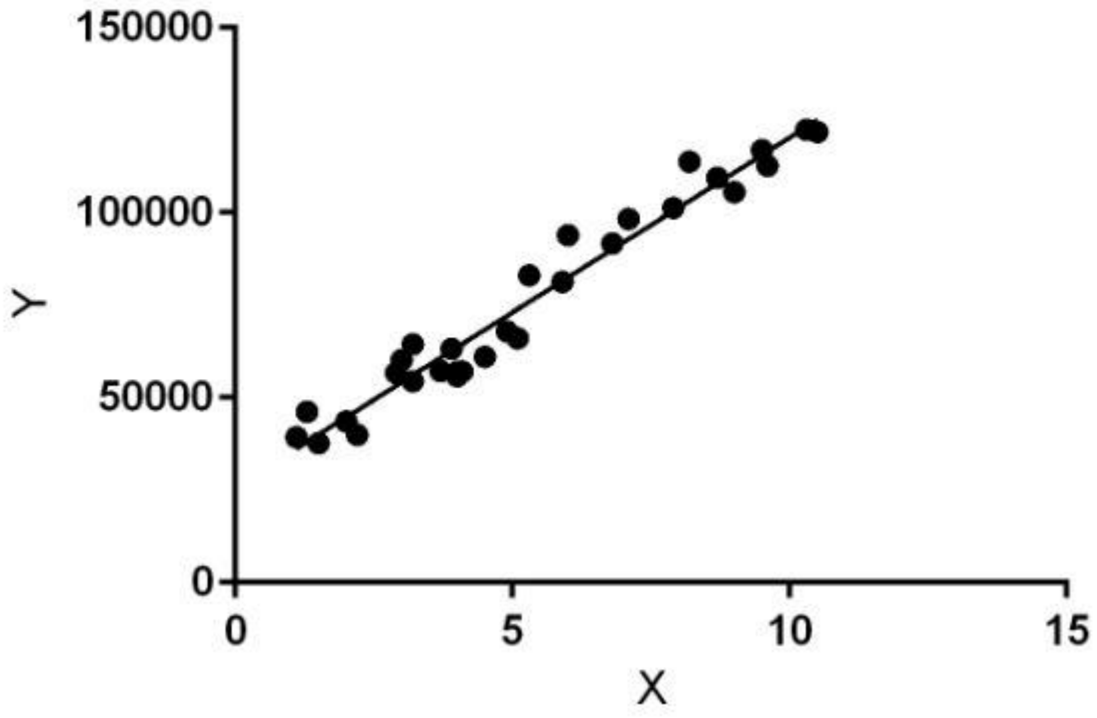
Module Outcomes

1) By the end of the first module the fully cleaned and useful data is available for the apply the algorithm for the prediction.

2) By the end of the second module the actual prediction will be happen the outcome is the amount of rainfall in inches based upon the user’s input.

Algorithm:

Linear Regression is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.
In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression:

$$y=mx+c$$

Where,

y is the response variable.

x is the predictor variable.

m and **c** are constants which are called the coefficients.

CHAPTER 3

DESIGN

1 Database Design

1.1 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

While a use case itself might drill into a lot of detail about every possibility, a use-case diagram can help provide a higher-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system”. They provide the simplified and graphical representation of what the system must do.

Purpose of Use Case Diagrams

The purpose of use case diagram is to capture the dynamic aspect of a system. However, this definition is too generic to describe the purpose, as other four diagrams (activity, sequence, collaboration, and Statechart) also have the same purpose. We will look into some specific purpose, which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analysed to gather its functionalities, use cases are prepared and actors are identified.

When the initial task is complete, use case diagrams are modelled to present the outside view.

In brief, the purposes of use case diagrams can be said to be as follows −

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements are actors.

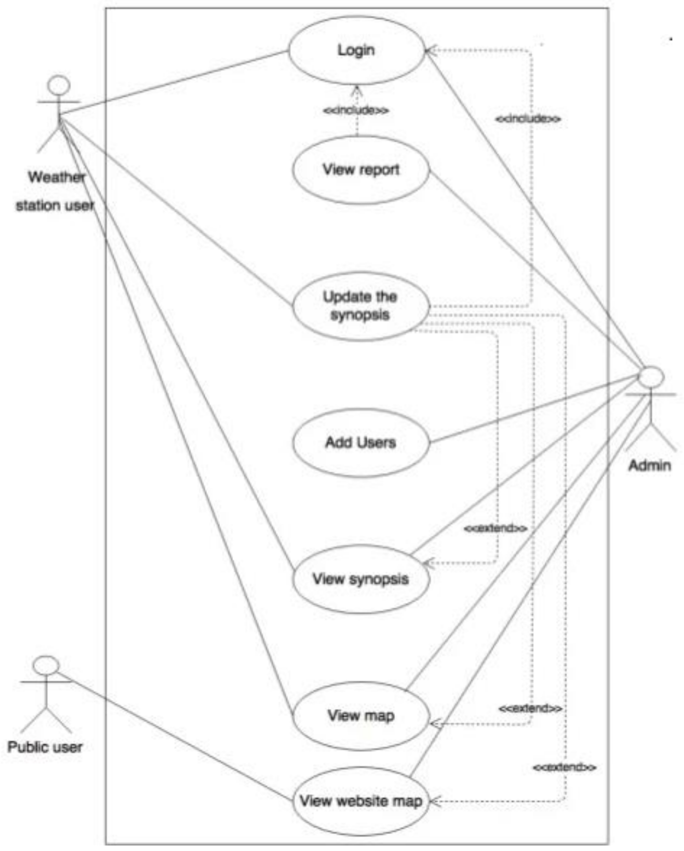


Fig.1. Use Case for weather forecasting system

1.2. ER Diagrams

ER Diagram stands for Entity Relationship Diagram, also known as ERD, is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes, and relationships.

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

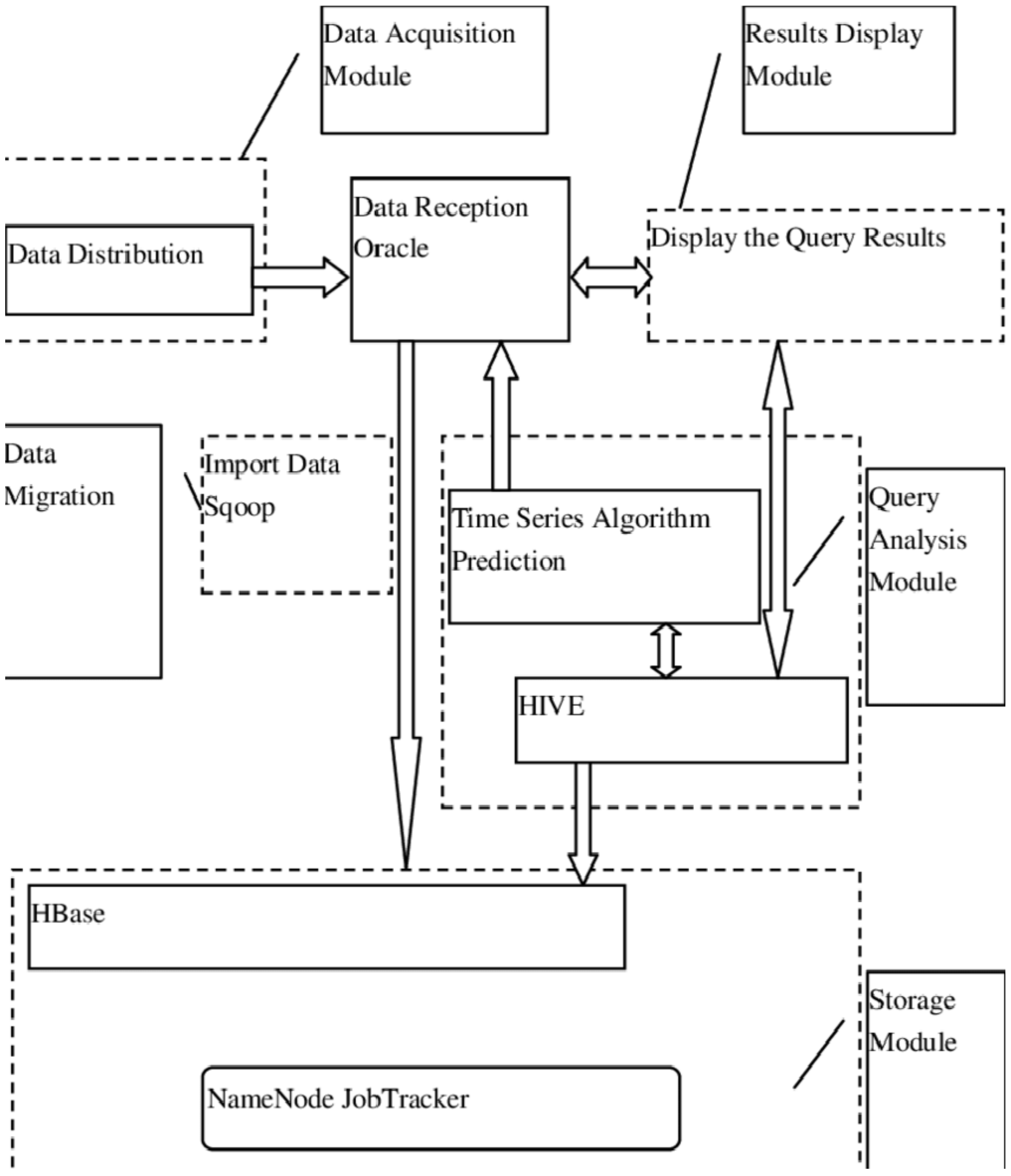


Fig.2. ER diagram for weather forecasting system

1.3. DFD Diagram

DFD is the abbreviation for **Data Flow Diagram**. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

Components of DFD

The Data Flow Diagram has 4 components:

- **Process**
  Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence
- **Dataflow**
  Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents material along with information that is being moved. Material shifts are modelled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi-directional.
- **Warehouse**
  The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The

data warehouse can be viewed independent of its implementation. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updating.

- **Terminator**

  The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity. Modelled systems also communicate with terminator.

Rules for creating DFD

- The name of the entity should be easy and understandable without any extra assistance(like comments).
- The processes should be numbered or put in ordered list to be referred easily.
- The DFD should maintain consistency across all the DFD levels.
- A single DFD can have maximum processes up to 9 and minimum 3 processes.

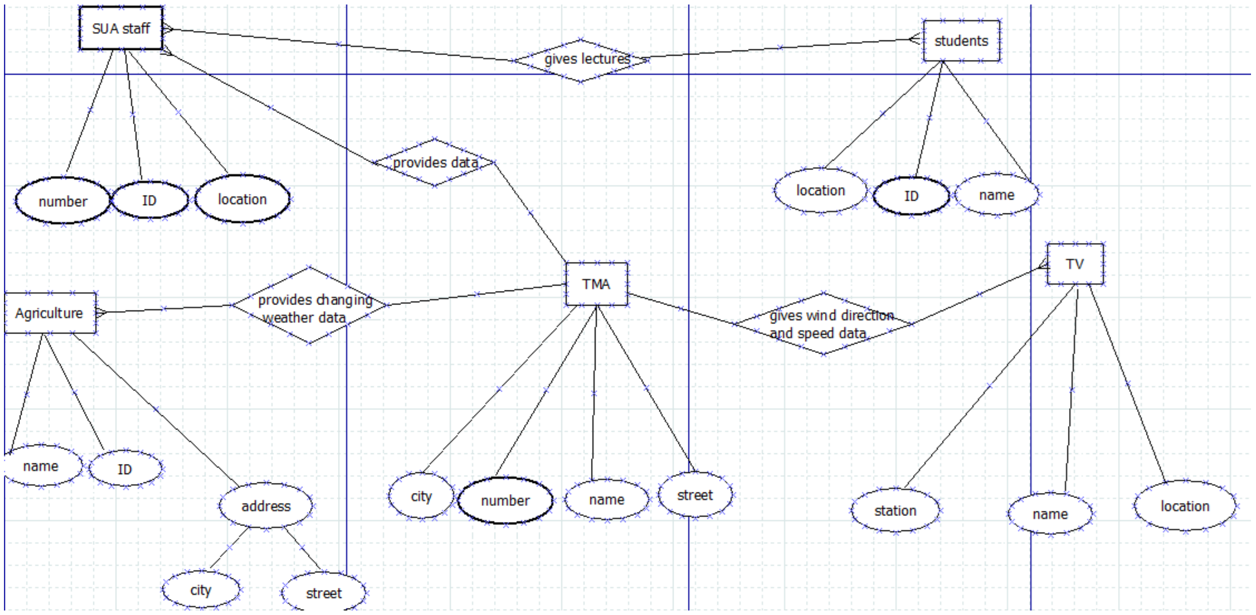


Fig.3. DFD for weather forecasting system

1.4 Sequence Diagram

A sequence diagram or system sequence diagram (SSD) shows object interactions arranged in time sequence in the field of software engineering. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. For a particular scenario of a use case, the diagrams show the events that external actors generate, their order, and possible inter-system events. All systems are treated as a black box; the diagram places emphasis on events that cross the system boundary from actors to systems. A system sequence diagram should be done for the main success scenario of the use case, and frequent or complex alternative scenarios.

i. To model high-level interaction among active objects within a system.

ii. To model interaction among objects inside a collaboration realizing a use case.

iii. It either models generic interactions or some certain instances of interaction. Sequence Diagram Notations –

i. Actors – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UMLdiagram

ii. Lifelines – A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically, each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram.

iii. Messages – Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram. iv. Guards – To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

Uses of sequence diagrams –

i. Used to model and visualize the logic behind a sophisticated function, operation, or procedure.

ii. They are also used to show details of UML use case diagrams.

iii. Used to understand the detailed functionality of current or future systems.

iv. Visualize how messages and tasks move between objects or components in a system

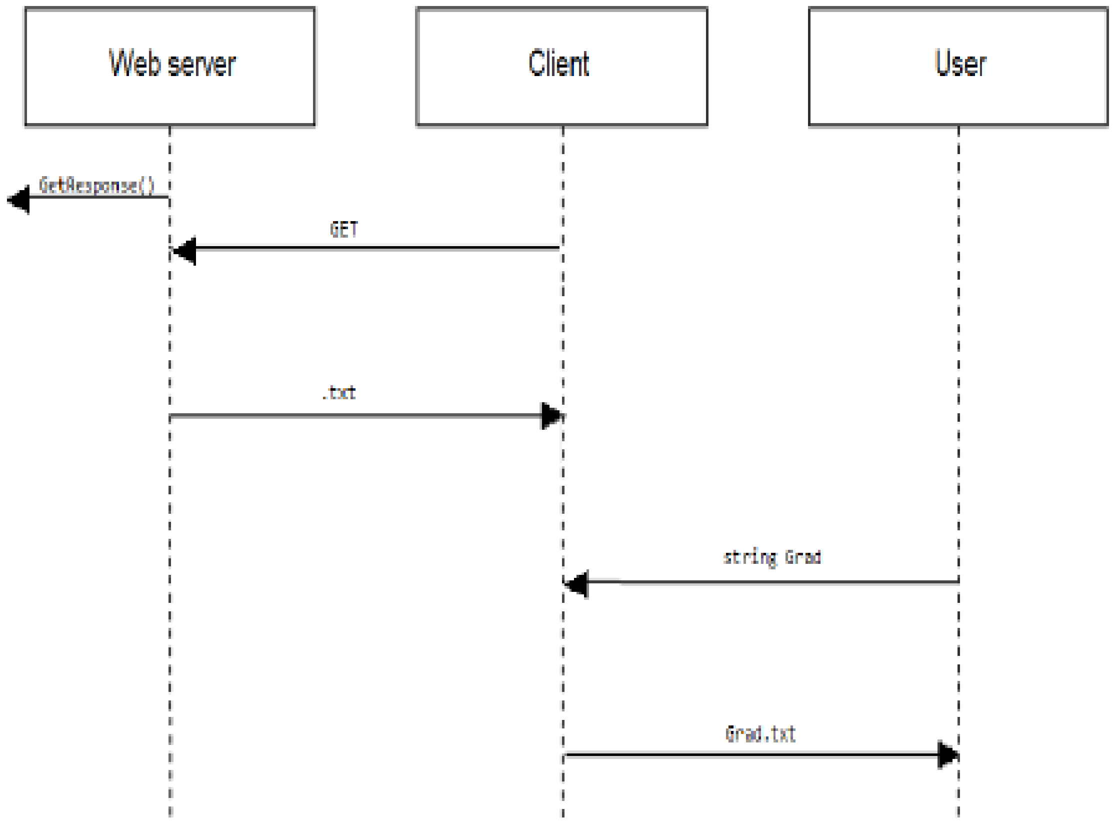


Fig.4. Sequence diagram for weather forecasting system

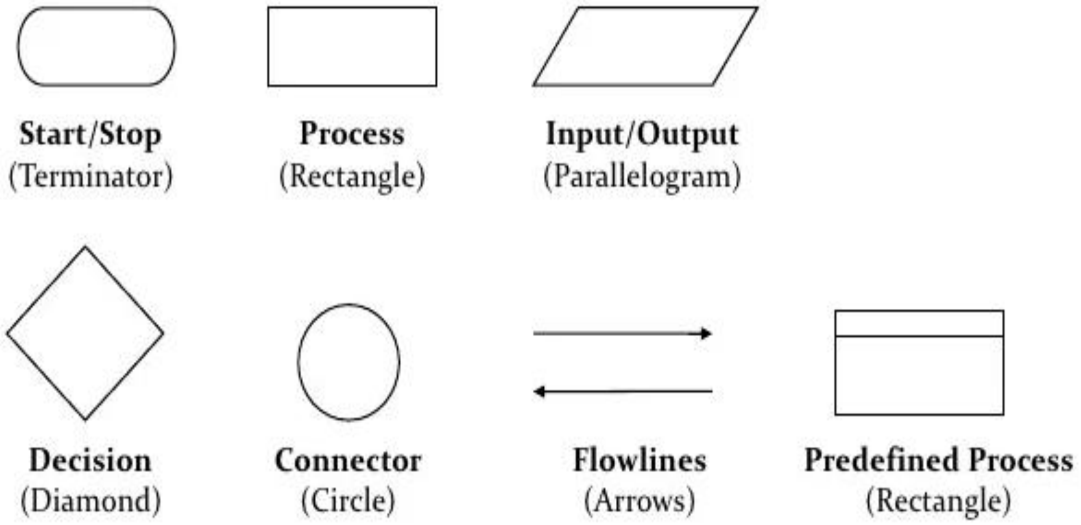
1.5 Flowchart Diagram

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams. Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence. They can range from simple, hand-drawn charts to comprehensive computer-drawn diagrams depicting multiple steps and routes. If we consider all the various forms of flowcharts, they are one of the most common diagrams on the planet, used by both technical and non-technical people in numerous fields. Flowcharts are sometimes called by more specialized names such as , Process Map, Functional Flowchart, Business Process Mapping, Business Process Modeling and Notation (BPMN), or Process Flow Diagram (PFD). They are related to other popular diagrams, such as Data Flow Diagrams (DFDs) and Unified Modeling Language (UML) Activity Diagrams.

Flowchart Annotations

Here are some of the common flowchart symbols-

- Start
- Stop
- Input/Output
- Decision
- Connectors
- Flowlines
- Pre-defined Process



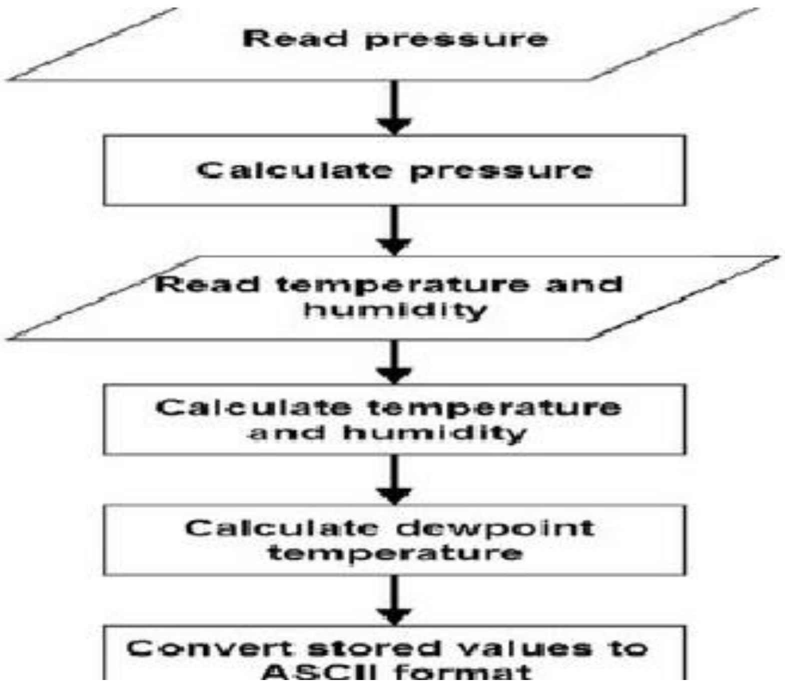
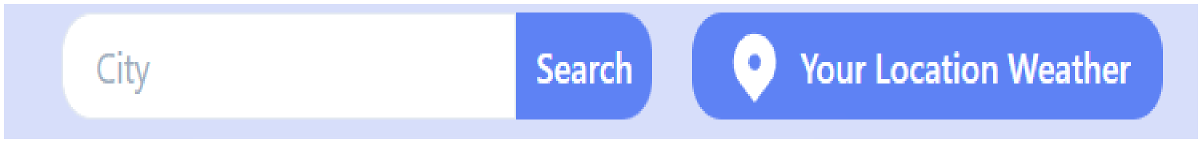

Read pressure
Calculate pressure
Read temperature and humidity
Calculate temperature and humidity
Calculate dewpoint temperature
Convert stored values to ASCII format

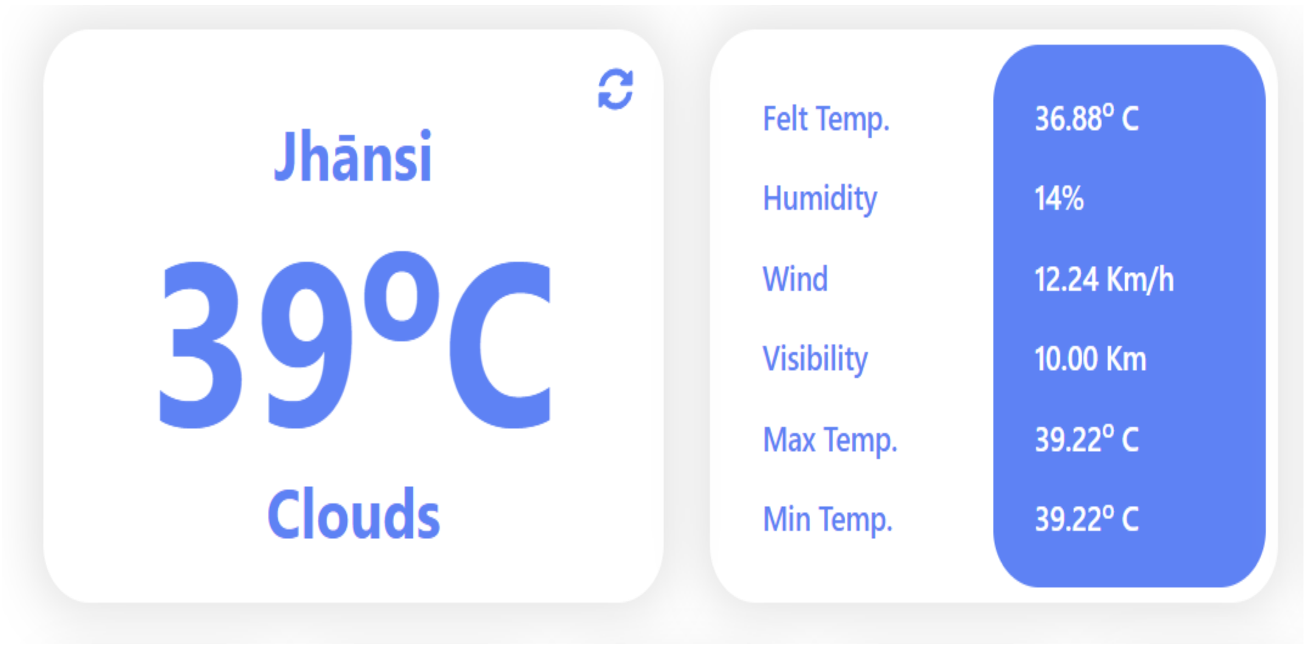
Fig.5. Flowchart for weather forecasting system

CHAPTER 4

LAYOUT

4.1 Input/Output Form Screenshots




Jhānsi

39°C

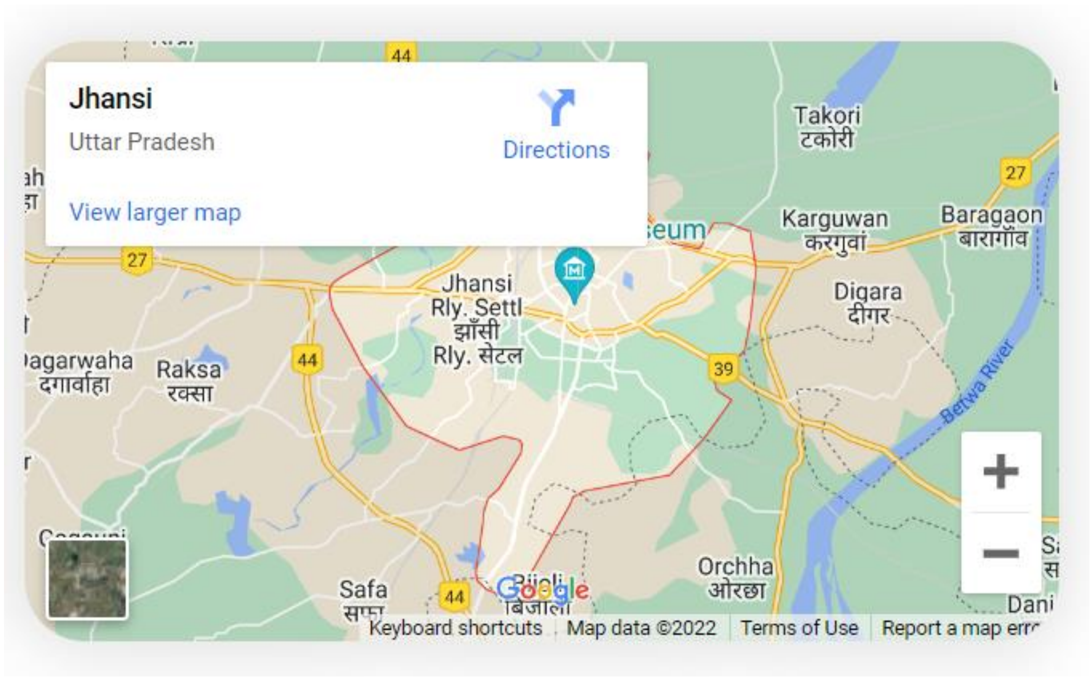
Clouds

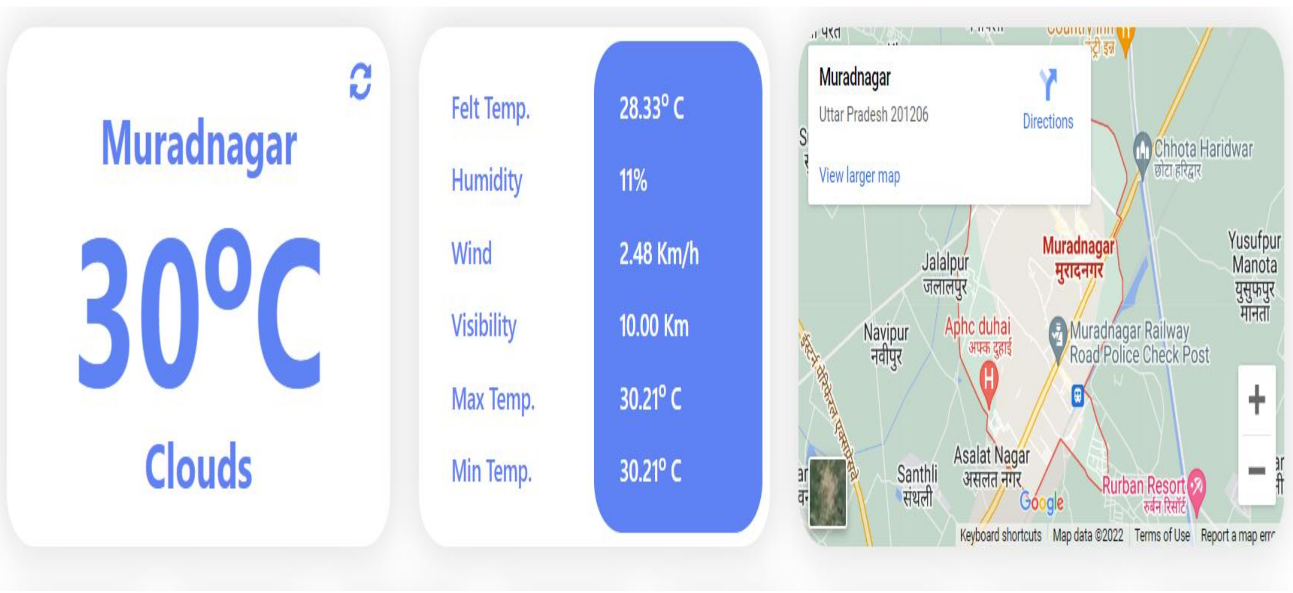
Felt Temp.	36.88° C
Humidity	14%
Wind	12.24 Km/h
Visibility	10.00 Km
Max Temp.	39.22° C
Min Temp.	39.22° C

Google Maps is a Web-based service that provides detailed information about geographical regions and sites around the world. In addition to conventional road maps, Google Maps offers aerial and satellite views of many places. In some cities, Google Maps offers street views comprising photographs taken from vehicles.

Google Maps offers several services as part of the larger Web application, as follows.

- A route planner offers directions for drivers, bikers, walkers, and users of public transportation who want to take a trip from one specific location to another.
- The Google Maps application program interface (API) makes it possible for website administrators to embed Google Maps into a proprietary site such as a real estate guide or community service page.
- Google Maps for Mobile offers a location service for motorists that utilizes the Global Positioning System (GPS) location of the mobile device (if available) along with data from wireless and cellular networks.
- Google Street View enables users to view and navigate through horizontal and vertical panoramic street level images of various cities around the world.
- Supplemental services offer images of the moon, Mars, and the heavens for hobby astronomers.

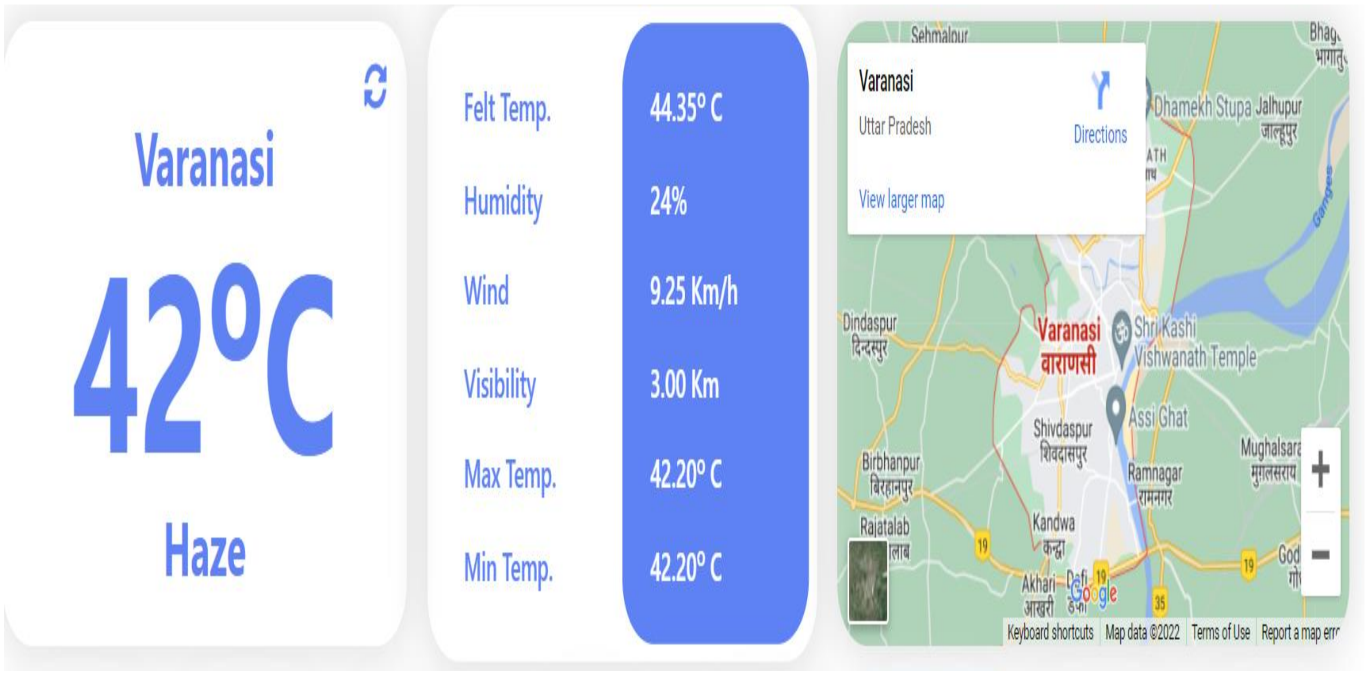



Muradnagar

30°C

Clouds

Felt Temp.	28.33° C
Humidity	11%
Wind	2.48 Km/h
Visibility	10.00 Km
Max Temp.	30.21° C
Min Temp.	30.21° C

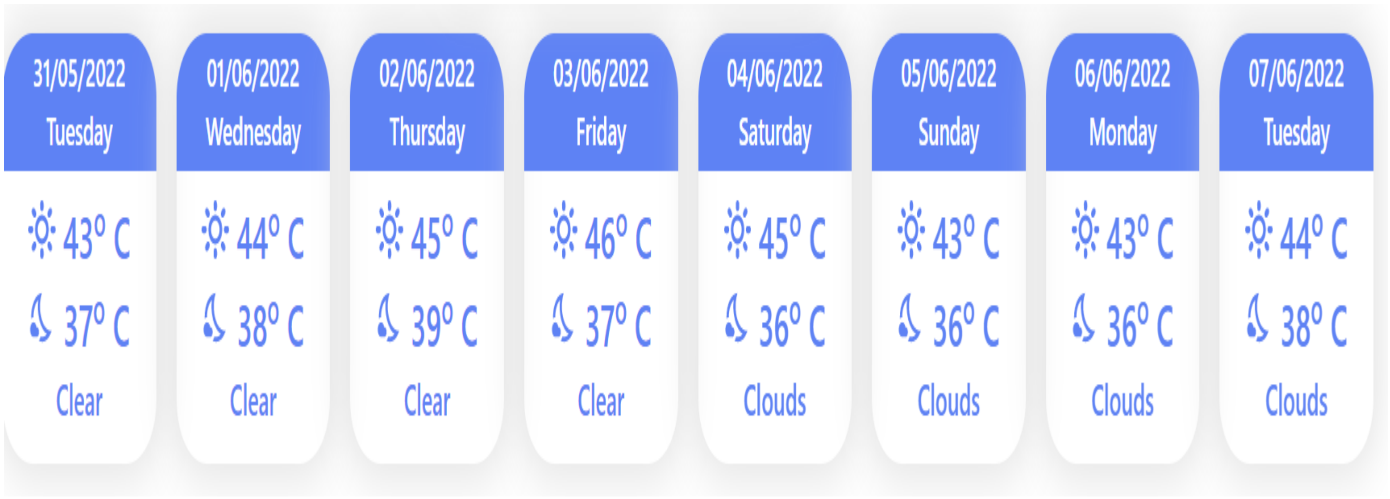


Varanasi

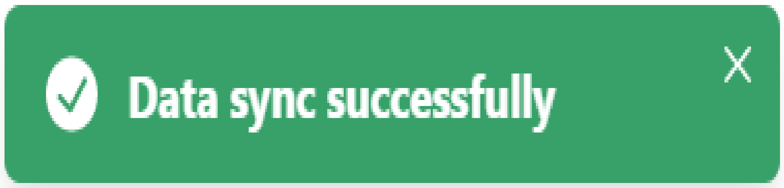
42°C

Haze

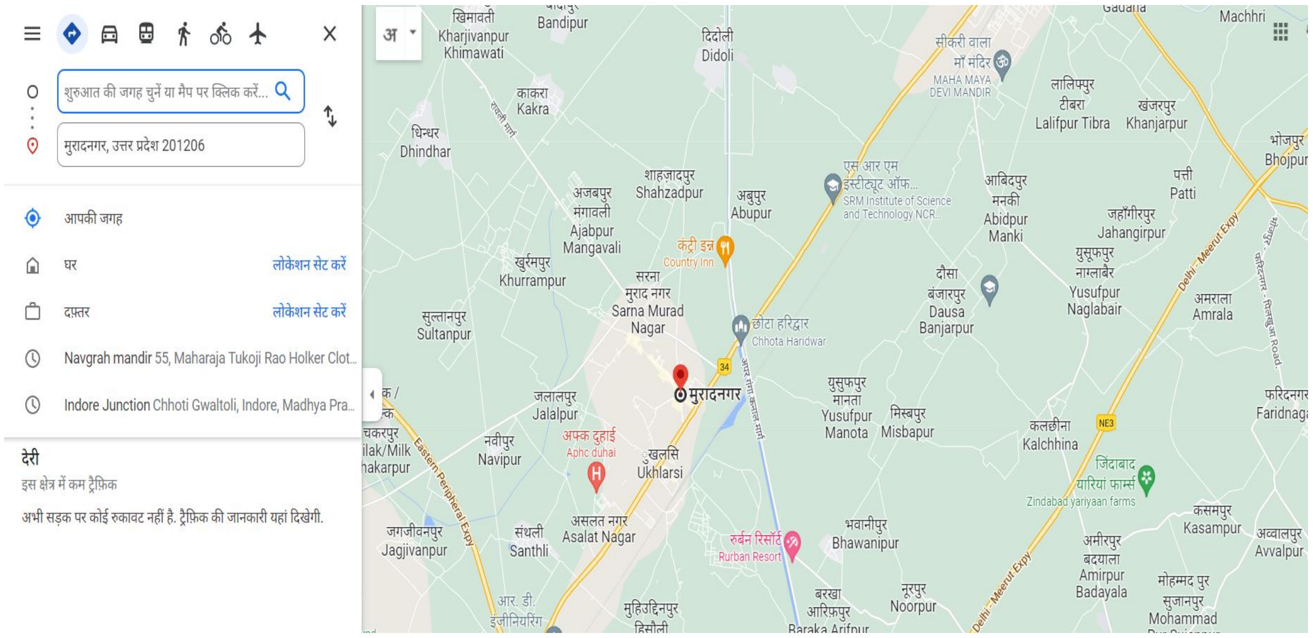
Felt Temp.	44.35° C
Humidity	24%
Wind	9.25 Km/h
Visibility	3.00 Km
Max Temp.	42.20° C
Min Temp.	42.20° C



31/05/2022 Tuesday	01/06/2022 Wednesday	02/06/2022 Thursday	03/06/2022 Friday	04/06/2022 Saturday	05/06/2022 Sunday	06/06/2022 Monday	07/06/2022 Tuesday
43° C	44° C	45° C	46° C	45° C	43° C	43° C	44° C
37° C	38° C	39° C	37° C	36° C	36° C	36° C	38° C
Clear	Clear	Clear	Clear	Clouds	Clouds	Clouds	Clouds



Data sync successfully



शुरुआत की जगह चुनें या मैप पर क्लिक करें...

मुरादनगर, उत्तर प्रदेश 201206

आपकी जगह

घर लोकेशन सेट करें

दफ़्तर लोकेशन सेट करें

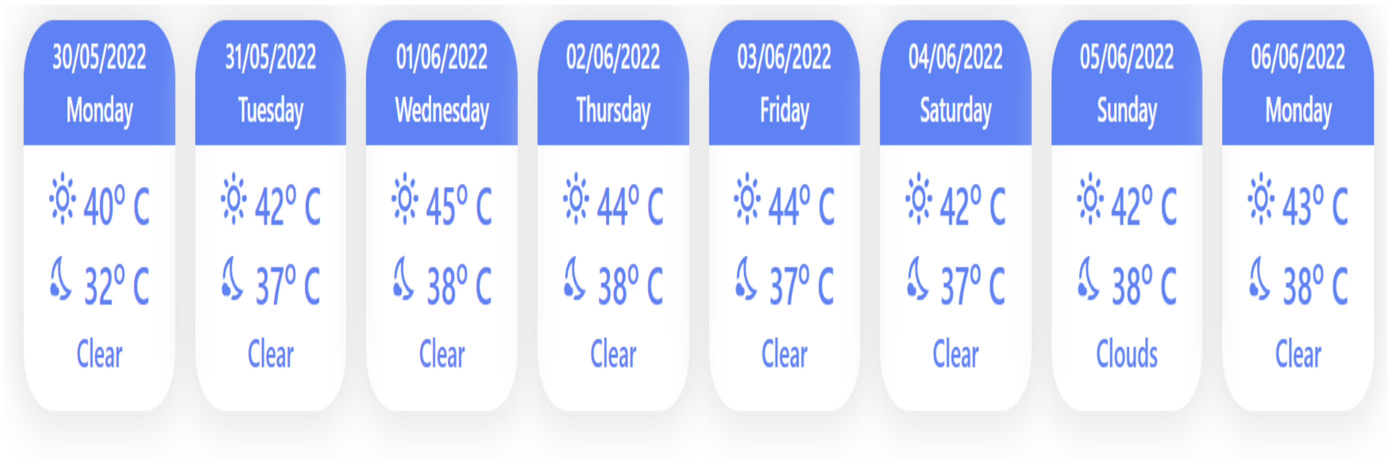
Navgrah mandir 55, Maharaja Tukoji Rao Holker Clot...

Indore Junction Chhoti Gwaltoli, Indore, Madhya Pra...

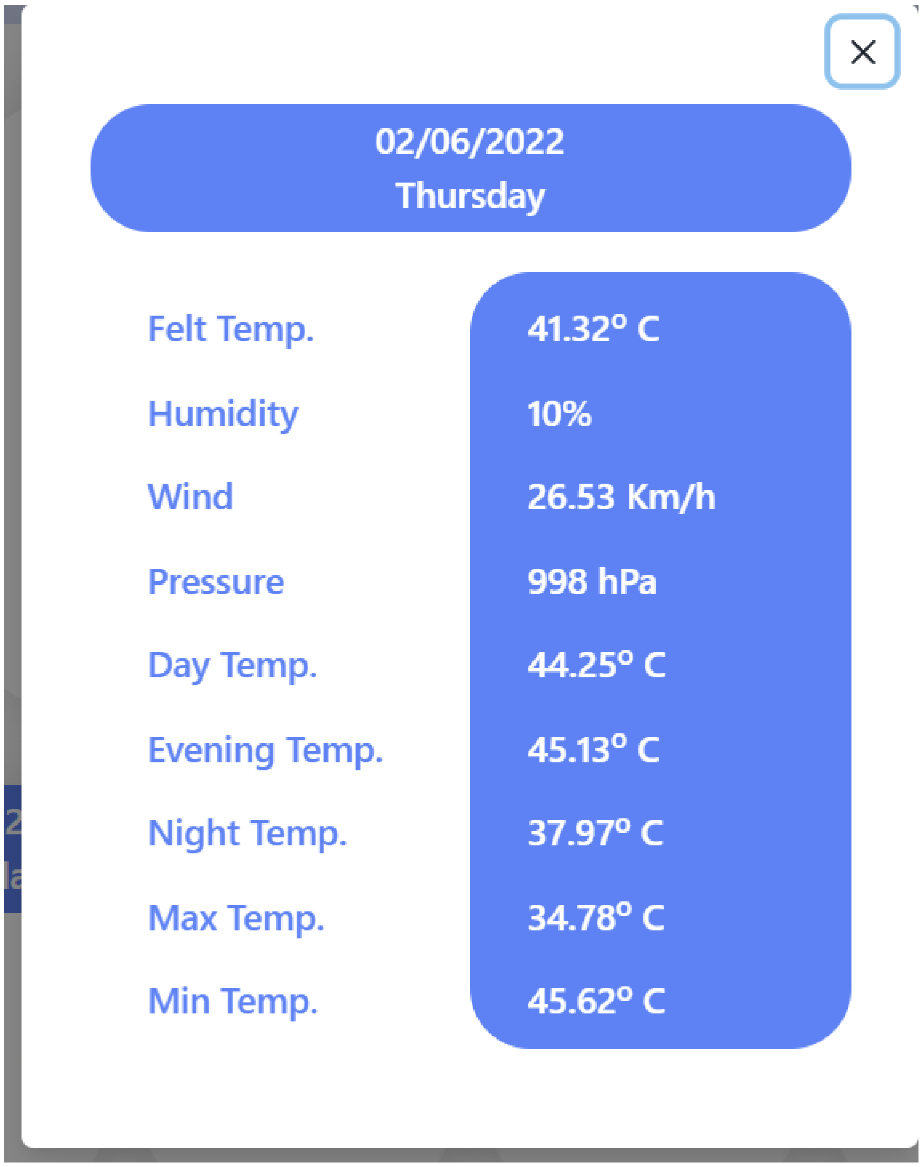
देरी

इस क्षेत्र में कम ट्रैफ़िक

अभी सड़क पर कोई रुकावट नहीं है. ट्रैफ़िक की जानकारी यहां दिखेगी.



30/05/2022 Monday	31/05/2022 Tuesday	01/06/2022 Wednesday	02/06/2022 Thursday	03/06/2022 Friday	04/06/2022 Saturday	05/06/2022 Sunday	06/06/2022 Monday
40⁰ C	42⁰ C	45⁰ C	44⁰ C	44⁰ C	42⁰ C	42⁰ C	43⁰ C
32⁰ C	37⁰ C	38⁰ C	38⁰ C	37⁰ C	37⁰ C	38⁰ C	38⁰ C
Clear	Clear	Clear	Clear	Clear	Clear	Clouds	Clear



02/06/2022
Thursday

Felt Temp.	41.32⁰ C
Humidity	10%
Wind	26.53 Km/h
Pressure	998 hPa
Day Temp.	44.25⁰ C
Evening Temp.	45.13⁰ C
Night Temp.	37.97⁰ C
Max Temp.	34.78⁰ C
Min Temp.	45.62⁰ C

CHAPTER 5

MODULE WISE CODE

5.1 Main.jsx

```
import { ChakraProvider } from '@chakra-ui/react'
import axios from 'axios'
import React from 'react'
import ReactDOM from 'react-dom'
import { Provider as ReduxProvider } from 'react-redux'
import App from './App'
import './index.css'
import { store } from './redux/store'

axios.defaults.baseURL = "https://api.openweathermap.org/data/2.5";

ReactDOM.render(
  <React.StrictMode>
    <ChakraProvider>
      <ReduxProvider store={store}>
        <App />
      </ReduxProvider>
    </ChakraProvider>
  </React.StrictMode>,
  document.getElementById('root')
)
```

5.2 App.jsx

```
import './App.css'

import { Deatils } from './components/Details';

import { Navbar } from './components/Navbar';



function App(

 return (

        <div>

            <Navbar />

            <Deatils />

        </div>

    )

}
```

```
export default App;
```

5.3 Details .jsx

```
import { Box, Flex, Grid, Heading, Icon, Text, useToast } from "@chakra-ui/react";
import { useEffect, useState } from "react";
import { shallowEqual, useDispatch, useSelector } from "react-redux";
import { celsius } from "../helpers/extraFunctions";
import { getItem } from "../helpers/sessionStorage";
import { getWeatherByLocation, syncData } from "../redux/actions";
import { Error } from "./Error";
import { Loading } from "./Loading";
import { Map } from "./Map";
import { FaSyncAlt } from "react-icons/fa";
import { Newbox, NewText } from "./SmallComponents";
import { Forcast } from "./Forcast";


export const Deatils = () => {

  const { isLoading, weatherData: data, forcastData, isError } = useSelector((state) => state,
shallowEqual);
  const [isRotate, setIsRotate] = useState(false);
  const dispatch = useDispatch();
  const toast = useToast();

  useEffect(() => {
    let weather = getItem("weather");
    !weather && dispatch(getWeatherByLocation(toast));
  }, []);

  const handleSyncData = () => {
    setIsRotate(true);
    dispatch(syncData(data.name, toast))
  }

  return isLoading ? (
    <Loading />
  ) : isError ? (
    <Error />
    ) : (
      <>
        <Box maxW={'1400px'} m={'20px auto 5px'} p={'20px'} minH={'550px'}>
          <Grid gridTemplateColumns={['100%', 'repeat(2, 1fr)', 'repeat(2, 1fr)', '30% 27.5%
38%']} gap={'30px'}>
            <Newbox>
              <Box color={'#5e82f4'} p={'20px'} textAlign={'center'}>
                <Flex justify={'end'}>
                  <Icon
                    onClick={handleSyncData}
                    onAnimationEnd={() => { setIsRotate(false) }}
                    className={isRotate ? "iconRotate" : undefined}
                    cursor={'pointer'} w={'23px'} h={'23px'} as={FaSyncAlt}
                  />
                </Flex>
                <Heading>{data.name}</Heading>
                <Heading fontSize={['100px', '120px', '120px', '100px',
'120px']}>{Math.round(data.main.temp - 273)}<sup>o</sup>C</Heading>
                <Heading>{data.weather[0].main}</Heading>
              </Box>
            </Newbox>

            <Newbox>
              <Grid templateColumns={'50% 50%'} h={'100%'} p={'8px'}>
                <Box py={'10px'} pl={'15%'}>
                  {['Felt Temp.', 'Humidity', 'Wind', 'Visibility', 'Max Temp.', 'Min
Temp.'].map((e, i) => (
                    <Text key={i} color={'#5e82f4'} fontWeight={500} mt={'15px'}
fontSize={'18px'} >{e}</Text>
                  ))}
                </Box>
```

```
                    <Box borderRadius={'30px'} bg={'#5e82f4'} py={'10px'} pl={'15%'}>
                        <NewText>{celsius(data.main.feels_like)}<sup>o</sup> C</NewText>
                        <NewText>{data.main.humidity}%</NewText>
                        <NewText>{(data.wind.speed * 3.6).toFixed(2)} Km/h</NewText>
                        <NewText>{(data.visibility * 0.001).toFixed(2)} Km</NewText>
                        <NewText>{celsius(data.main.temp_max)}<sup>o</sup> C</NewText>
                        <NewText>{celsius(data.main.temp_min)}<sup>o</sup> C</NewText>
                    </Box>
                </Grid>
            </Newbox>

            <Newbox>
                <Map city={data.name} />
            </Newbox>
        </Grid>

        <Grid mt={'40px'} templateColumns={['repeat(2, 1fr)', 'repeat(3, 1fr)', 'repeat(4, 1fr)',
'repeat(5, 1fr)', 'repeat(8, 1fr)']} gap={'20px'}>
            {forcastData.map((e, i) => <Forcast key={i} data={e} />)}
        </Grid>
    </Box >
  </>
 );
};
```

5.4 Error.jsx

```
import { Container, Image } from "@chakra-ui/react";

export const Error = () => {

    return (
        <Container mt={['200px', '100px']} p={'100px'}>
            <Image src='/images/Error.gif' />
        </Container>
    );
};
```

5.5 Forcast.jsx

```
import { Box, Icon, Text } from "@chakra-ui/react";
import { ForcastBox } from "./SmallComponents";
import { ImSun } from "react-icons/im";
import { MdOutlineNightsStay } from "react-icons/md";
import { dateFormat } from "../helpers/extraFunctions";
import { setItem } from "../helpers/sessionStorage";
import { ForcastModal } from "./ForcastModal";

export const Forcast = ({ data }) => {

    const { date, day } = dateFormat(data.dt);

    return (
        <Box>
            <ForcastBox >
            <Box p={'5px'} bg={'#5e82f4'}>
              <Text fontWeight={500} color={'white'} fontSize={'18px'}>{date}</Text>
              <Text fontWeight={500} color={'white'} fontSize={'18px'}>{day}</Text>
            </Box>

            <ForcastModal data={data} />

          </ForcastBox>
        </Box>
    );
};
```

5.6 ForcastModal.jsx

```
import { Box, Grid, Icon, Modal, ModalBody, ModalCloseButton, ModalContent, ModalFooter,
ModalHeader, ModalOverlay, Text, useDisclosure } from "@chakra-ui/react";
import { dateFormat } from "../helpers/extraFunctions";
import { NewText } from "./SmallComponents";
import { ImSun } from "react-icons/im";
import { MdOutlineNightsStay } from "react-icons/md";


export const ForcastModal = ({ data }) => {

    const { date, day } = dateFormat(data.dt);
    const { isOpen, onOpen, onClose } = useDisclosure();

    return (
        <>
        <Box onClick={onOpen} cursor={'pointer'} mt={'10px'}>
          <Text color={'#5e82f4'} fontWeight={500} fontSize={'27px'}>
            <Icon as={ImSun} /> {Math.round(data.temp.day)}<sup>o</sup> C
          </Text>
          <Text color={'#5e82f4'} fontWeight={500} fontSize={'27px'}>
            <Icon as={MdOutlineNightsStay} /> {Math.round(data.temp.night)}<sup>o</sup>
          </Text>
          <Text color={'#5e82f4'} fontWeight={500} fontSize={'20px'}>
            {data.weather[0].main}
          </Text>
        </Box>

        <Modal isOpen={isOpen} onClose={onClose} >
          <ModalOverlay />
          <ModalContent>
            <ModalHeader></ModalHeader>
            <ModalCloseButton />

            <ModalBody>
              <Box p={'10px'}>
                <Box p={'5px'} bg={'#5e82f4'} textAlign={'center'} borderRadius={'30px'}
mb={'20px'} >
                  <Text fontWeight={500} color={'white'} fontSize={'18px'}>{date}</Text>
                  <Text fontWeight={500} color={'white'} fontSize={'18px'}>{day}</Text>
                </Box>

                <Grid templateColumns={'50% 50%'} >
                  <Box pb={'10px'} pl={'15%'}>
                    {['Felt Temp.', 'Humidity', 'Wind', 'Pressure', 'Day Temp.', 'Evening
Temp.', 'Night Temp.', 'Max Temp.', 'Min Temp.'].map((e, i) => (
                      <Text key={i} color={'#5e82f4'} fontWeight={500} mt={'15px'}
fontSize={'18px'} >{e}</Text>
                              ))}
                          </Box>
                          <Box borderRadius={'30px'} bg={'#5e82f4'} pb={'10px'} pl={'15%'}>
                              <NewText>{data.feels_like.day}<sup>o</sup> C</NewText>
                              <NewText>{data.humidity}%</NewText>
                              <NewText>{(data.wind_speed * 3.6).toFixed(2)} Km/h</NewText>
                              <NewText>{data.pressure} hPa</NewText>
                              <NewText>{data.temp.day}<sup>o</sup> C</NewText>
                              <NewText>{data.temp.eve}<sup>o</sup> C</NewText>
                              <NewText>{data.temp.night}<sup>o</sup> C</NewText>
                              <NewText>{data.temp.min}<sup>o</sup> C</NewText>
                              <NewText>{data.temp.max}<sup>o</sup> C</NewText>
                          </Box>
                      </Grid>
                  </Box>
              </ModalBody>

              <ModalFooter></ModalFooter>
          </ModalContent>
      </Modal>
    </>
  );
};
```

5.7 Loading.jsx

```
import { Container, Image } from "@chakra-ui/react";

export const Loading = () => {

  return (
        <Container mt={['200px', '100px']} >
            <Image src='/images/loading.gif' />
        </Container>
    );
};
```

5.8 Map.jsx

```
import { googleMapAPI } from "../helpers/API";

export const Map = ({ city }) => {

    return (
        <div>
            <iframe
                width="100%"
                height="300"
                loading="lazy"
                allowFullScreen
                referrerPolicy="no-referrer-when-downgrade"

src={`https://www.google.com/maps/embed/v1/place?key=${googleMapAPI}&q=${city}`}>
            </iframe>
        </div>
    );
};
```

5.9 Navbar.jsx

```
import { Button, Center, Flex, Icon, Input, useToast } from "@chakra-ui/react";
import { useState } from "react";
import { useDispatch } from "react-redux";
import { getWeatherByCity, getWeatherByLocation } from "../redux/actions";
import { HiLocationMarker } from "react-icons/hi";

export const Navbar = () => {

    const [city, setCity] = useState("");
    const dispatch = useDispatch();
    const toast = useToast();

    const handleChnage = () => {
        dispatch(getWeatherByCity(city, toast));
    }

    const handleLocationData = () => {
        dispatch(getWeatherByLocation(toast));
    }

    return (
        <Flex p={'10px'} minH={'70px'} bg={'#d7defa'} justifyContent={'center'}
flexDirection={['column', 'row']} gap={['10px', '0px']}>
            <Center px={'10px'}>
                <Input
                    onKeyPress={({ key }) => { key === "Enter" ? handleChnage() : undefined }}
                    onInput={(e) => { setCity(e.target.value) }}
                    value={city}
                    borderRadius={'15px 0px 0px 15px'}
                    bg={'white'}
                    _focus={{ 'border': 'none' }}
                    placeholder="City"
                />
                <Button
                    onClick={handleChnage}
                    borderRadius={'0px 15px 15px 0px'}
                    color={'white'}
                    bg={'#5e82f4'}
                    _hover={{ 'bg': '5e82f4' }}
                >
                    Search
                </Button>
            </Center>
            <Center px={'10px'}>
                <Button
                    bg={'#5e82f4'}
                    _hover={{ 'bg': '5e82f4' }}
                    color={'white'}
                    w={'100%'}
                    borderRadius={'15px'}
                    leftIcon={<Icon w={'30px'} h={'30px'} as={HiLocationMarker} />}
                    onClick={handleLocationData}
                >
                    Your Location Weather
                </Button>
            </Center>
        </Flex >
    );
};
```

5.10 SmallComponents.jsx

```
import { Box, Text } from "@chakra-ui/react";

export const ForcastBox = ({ children }) => {
   return (
      <Box className="zoom" textAlign={'center'} overflow={'hidden'} borderRadius={'30px'}
shadow={'0px 0px 30px 6px #E2E2E2'} h={'200px'}>
         {children}
      </Box>
   );
};

export const Newbox = ({ children }) => {
   return (
      <Box className="zoom" overflow={'hidden'} shadow={'0px 0px 30px 6px #E2E2E2'}
borderRadius={'30px'} h={'300px'}>
         {children}
      </Box>
   );
};

export const NewText = ({ children }) => {
   return (
      <Text color={'white'} fontWeight={500} mt={'15px'} fontSize={'18px'}>
         {children}
      </Text>
   );
};
```

5.11 App.css

```
.iconRotate {
   animation: iconRotator 1s linear 0s 1;
}

@keyframes iconRotator {
   0% {
      transform: rotate(0deg);
   }

   100% {
      transform: rotate(360deg);
   }
}

.zoom {
   transition: all 0.5s linear;
}

.zoom:hover {
   transform: scale(1.05);
}


.tagLine {
   display: flex;
   justify-content: center;
   margin-top: 20px;
}

#madeByMohit {
   color: #5e82f4;
   font-weight: 500;
   width: 0ch;
   overflow: hidden;
   white-space: nowrap;
   animation: tagline 10s  infinite alternate;
   border-right: 3px solid #5e82f4;
}

@keyframes tagline {
     0% {
        width: 0ch;
```

```
    }
    50% {
      width: 28ch;
    }
}
```

5.12 Favicon.svg

```
<svg width="410" height="404" viewBox="0 0 410 404" fill="none"
xmlns="http://www.w3.org/2000/svg">
<path d="M399.641 59.5246L215.643 388.545C211.844 395.338 202.084 395.378 198.228
388.618L10.5817 59.5563C6.38087 52.1896 12.6802 43.2665 21.0281 44.7586L205.223
77.6824C206.398 77.8924 207.601 77.8904 208.776 77.6763L389.119 44.8058C397.439
43.2894 403.768 52.1434 399.641 59.5246Z" fill="url(#paint0_linear)"/>
<path d="M292.965 1.5744L156.801 28.2552C154.563 28.6937 152.906 30.5903 152.771
32.8664L144.395 174.33C144.198 177.662 147.258 180.248 150.51 179.498L188.42
170.749C191.967 169.931 195.172 173.055 194.443 176.622L183.18 231.775C182.422 235.487
185.907 238.661 189.532 237.56L212.947 230.446C216.577 229.344 220.065 232.527 219.297
236.242L201.398 322.875C200.278 328.294 207.486 331.249 210.492 326.603L212.5
323.5L323.454 102.072C325.312 98.3645 322.108 94.137 318.036 94.9228L279.014
102.454C275.347 103.161 272.227 99.746 273.262 96.1583L298.731 7.86689C299.767 4.27314
296.636 0.855181 292.965 1.5744Z" fill="url(#paint1_linear)"/>
<defs>
<linearGradient id="paint0_linear" x1="6.00017" y1="32.9999" x2="235" y2="344"
gradientUnits="userSpaceOnUse">
<stop stop-color="#41D1FF"/>
<stop offset="1" stop-color="#BD34FE"/>
</linearGradient>
<linearGradient id="paint1_linear" x1="194.651" y1="8.81818" x2="236.076" y2="292.989"
gradientUnits="userSpaceOnUse">
<stop stop-color="#FFEA83"/>
<stop offset="0.0833333" stop-color="#FFDD35"/>
<stop offset="1" stop-color="#FFA800"/>
```

```
</linearGradient>
</defs>
</svg>
```

5.13 Index.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

5.14 Action.js

```
import axios from "axios";
import { weatherAppAPI } from "../helpers/API";
import { myToast } from "../helpers/extraFunctions";
import { setItem } from "../helpers/sessionStorage";
import { GET_DATA_ERROR, GET_DATA_LOADING, GET_DATA_SUCCESS } from
"./actionTypes";

export const getDataLoading = () => {
   return { type: GET_DATA_LOADING };
}

export const getDataSuccess = (payload) => {
   return { type: GET_DATA_SUCCESS, payload };
}

export const getDataError = () => {
   return { type: GET_DATA_ERROR };
}

export const getWeatherByLocation = (toast) => (dispatch) => {

   const success = async (position) => {
       try {
           let { latitude, longitude } = position.coords;
           dispatch(getDataLoading());
           let weatherData = await
axios.get(`/weather?lat=${latitude}&lon=${longitude}&appid=${weatherAppAPI}`);
           weatherData = weatherData.data;
           let forcastData = await
axios.get(`/onecall?lat=${latitude}&lon=${longitude}&exclude=hourly,minutely&units=metric
&appid=${weatherAppAPI}`);
           forcastData = forcastData.data.daily;
           let payload = { weatherData, forcastData }
           dispatch(getDataSuccess(payload));
```

```
            setItem("weather", payload);
            myToast(toast, "Your location weather updated", "success")
        } catch (err) {
            console.log(err);
            dispatch(getDataError());
        }
    }

    const error = (err) => {
        console.warn(`ERROR(${err.code}): ${err.message}`);
        myToast(toast, "Please turn on your location", "error")
    }

    navigator.geolocation.getCurrentPosition(success, error);
}

export const getWeatherByCity = (city, toast) => async (dispatch) => {
    try {
        dispatch(getDataLoading());
        let weatherData = await axios.get(`/weather?q=${city}&appid=${weatherAppAPI}`);
        weatherData = weatherData.data;
        let { lon, lat } = weatherData.coord;
        let forcastData = await
axios.get(`/onecall?lat=${lat}&lon=${lon}&exclude=hourly,minutely&units=metric&appid=${
weatherAppAPI}`);
        forcastData = forcastData.data.daily;
        let payload = { weatherData, forcastData };
        dispatch(getDataSuccess(payload));
        setItem("weather", payload);
        myToast(toast, "City weather data updated", "success");
    } catch (err) {
        console.log(err);
        dispatch(getDataError());
        myToast(toast, "City weather data doesn't exist", "error");
    }
}

export const syncData = (city, toast) => async (dispatch) => {
    try {
        let weatherData = await axios.get(`/weather?q=${city}&appid=${weatherAppAPI}`);
        weatherData = weatherData.data;
        let { lon, lat } = weatherData.coord;
        let forcastData = await
axios.get(`/onecall?lat=${lat}&lon=${lon}&exclude=hourly,minutely&units=metric&appid=${
weatherAppAPI}`);
        forcastData = forcastData.data.daily;
        let payload = { weatherData, forcastData };
        dispatch(getDataSuccess(payload));
        setItem("weather", payload);
        myToast(toast, "Data sync successfully", "success");
    } catch (err) {
        console.log(err);
        dispatch(getDataError());
        myToast(toast, "City weather data doesn't exist", "error");
    }
}
```

5.15 ActionTypes.js

```
export const GET_DATA_LOADING = "GET_DATA_LOADING";
export const GET_DATA_SUCCESS = "GET_DATA_SUCCESS";
export const GET_DATA_ERROR = "GET_DATA_ERROR";
```

5.16 Reducer.js

```
import { getItem } from "../helpers/sessionStorage";
import { GET_DATA_ERROR, GET_DATA_LOADING, GET_DATA_SUCCESS } from
"./actionTypes";


const initState = {
  isLoading: getItem("weather") ? false : true,
  weatherData: getItem("weather") ? getItem("weather").weatherData : {},
  forcastData: getItem("weather") ? getItem("weather").forcastData : [],
  isError: false
}


export const reducer = (state = initState, { type, payload }) => {
  switch (type) {
    case GET_DATA_LOADING:
      return {
        ...state,
        isLoading: true,
        isError: false
      };
    case GET_DATA_SUCCESS:
      return {
        ...state,
        isLoading: false,
        isError: false,
        weatherData: payload.weatherData,
                forcastData: payload.forcastData
            };
        case GET_DATA_ERROR:
            return {
                ...state,
                isLoading: false,
                isError: true
            };
        default:
            return state;
    }
}
```

5.17 Store.js

```
import { applyMiddleware, createStore } from "redux";
import thunk from "redux-thunk";
import { reducer } from "./reducer";


export const store = createStore(reducer, applyMiddleware(thunk));
```

5.18 Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="apple-touch-icon" sizes="180x180" href="/apple-touch-icon.png">
  <link rel="icon" type="image/png" sizes="32x32" href="/favicon-32x32.png">
  <link rel="icon" type="image/png" sizes="16x16" href="/favicon-16x16.png">
  <link rel="manifest" href="/site.webmanifest">
  <link rel="mask-icon" href="/safari-pinned-tab.svg" color="#5bbad5">
  <meta name="msapplication-TileColor" content="#da532c">
  <meta name="theme-color" content="#ffffff">
  <title>Weather App</title>
</head>

<body>
  <div id="root"></div>
  <script type="module" src="/src/main.jsx"></script>
</body>

</html>
```

5.19 Spinner

```
import React from 'react';
import { StyledSpinner } from '../../styles';
const Spinner = () => (
<StyledSpinner>
<svg width="200" height="150" viewBox="0 0 800 600" fill="none" xmlns="http://www.w3.org/2000/svg" data-reactroot="">
<path d="M667.214 299.996C667.214 346.785 655.45 390.834 634.705 429.331C634.63 429.451 634.57 429.586 634.495 429.706C588.335 515.095 497.991 573.11 394.101 573.11C243.273 573.11 120.988 450.84 120.988 299.996C120.988 284.685 122.246 269.673 124.67 255.034C124.76 254.45 124.865 253.851 124.97 253.268C146.029 131.133 248.213 36.7618 374.045 27.6322C380.66 27.1382 387.351 26.8984 394.101 26.8984C476.528 26.8984 550.423 63.4045 600.489 121.134C642.069 169.06 667.214 231.594 667.214 299.996Z" fill="none"
```

```
stroke="#221b38" strokeWidth="1"></path>
<path fillRule="evenodd" clipRule="evenodd" d="M394.105 11C386.961 11 379.882 11.2537
372.883 11.7764L372.881 11.7765C239.727 21.4373 131.596 121.301 109.311
250.543C106.539 266.619 105.098 283.139 105.098 299.992C105.098 459.614 234.499 589
394.105 589C504.199 589 599.928 527.437 648.712 436.853L649.593 437.327L648.712
436.853C670.664 396.117 683.113 349.506 683.113 299.992C683.113 227.609 656.505 161.436
612.505 110.72L613.221 110.099L612.505 110.72C559.523 49.6293 481.329 11 394.105
11ZM372.735 9.78185C379.785 9.25536 386.914 9 394.105 9C481.933 9 560.67 47.8994
614.016 109.41L613.26 110.065L614.016 109.41C658.319 160.476 685.113 227.11 685.113
299.992C685.113 349.845 672.578 396.781 650.473 437.802C601.354 529.007 504.963 591
394.105 591C233.395 591 103.098 460.719 103.098 299.992C103.098 283.025 104.549 266.391
107.34 250.203L108.324 250.373L107.34 250.203C129.779 120.064 238.656 19.5103 372.735
9.78185Z" fill="#B5CDFB"></path>
<path className="fill" d="M634.495 429.706C588.335 515.096 497.991 573.111 394.101
573.111C243.273 573.111 120.988 450.841 120.988 299.997C120.988 284.685 122.246 269.673
124.67 255.035C125.434 255.184 126.167 255.394 126.93 255.633C126.93 255.633 188.627
215.565 255.906 274.388C323.185 333.195 308.816 497.569 394.715 468.637C480.614 439.69
496.839 424.647 530.411 426.952C563.968 429.272 594.442 436.217 600.04 417.687C604.8
401.971 627.805 422.746 634.495 429.706Z" fill="#000000"></path>
<path className="fill" d="M581.966 299.891C602.187 320.651 594.644 355.286 567.627
365.778C554.156 371.002 541.045 375.283 531.93 376.42C504.419 379.863 476.894 390.191
452.826 335.723C428.743 281.272 395.5 324.842 367.99 298.5C340.465 272.171 417.278
219.366 427.77 117.332C438.263 15.2986 394.348 96.692 377.853 94.4015C367.197 92.9201
370.31 54.603 373.857 27.6614C373.962 27.6468 374.052 27.6315 374.157 27.6315C380.637
27.1374 387.358 26.8984 394.108 26.8984C472.763 26.8984 547.451 60.9343 599.089
120.31C581.322 128.767 563.257 134.784 552.57 138.885C526.197 148.989 526.197 207.901
522.919 225.099C521.063 234.828 553.483 270.645 581.966 299.891Z" fill="#000000"></path>
<path d="M200.037 92.899C200.281 91.2473 200.448 89.5713 200.448 87.8528C200.448
68.8544 185.047 53.4531 166.049 53.4531H153.278C134.28 53.4531 118.879 68.8544 118.879
87.8528H98.3994C79.4012 87.8528 64 103.254 64 122.252C64 141.251 79.4012 156.652
98.3994 156.652H182.169C201.168 156.652 216.569 141.251 216.569 122.252C216.569
```

109.807 209.937 98.9383 200.037 92.899Z" fill="#B5CDFB"></path>
<path d="M171.135 69.3307C171.821 69.3307 172.377 68.7746 172.377 68.0887C172.377 67.4027 171.821 66.8467 171.135 66.8467C170.449 66.8467 169.893 67.4027 169.893 68.0887C169.893 68.7746 170.449 69.3307 171.135 69.3307Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M181.406 76.4518C182.092 76.4518 182.648 75.8957 182.648 75.2098C182.648 74.5238 182.092 73.9678 181.406 73.9678C180.72 73.9678 180.164 74.5238 180.164 75.2098C180.164 75.8957 180.72 76.4518 181.406 76.4518Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M171.754 80.4644C172.44 80.4644 172.996 79.9084 172.996 79.2225C172.996 78.5365 172.44 77.9805 171.754 77.9805C171.068 77.9805 170.512 78.5365 170.512 79.2225C170.512 79.9084 171.068 80.4644 171.754 80.4644Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M203.762 136.591C204.447 136.591 205.004 136.035 205.004 135.349C205.004 134.663 204.447 134.107 203.762 134.107C203.076 134.107 202.52 134.663 202.52 135.349C202.52 136.035 203.076 136.591 203.762 136.591Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M190.406 114.713C191.092 114.713 191.648 114.157 191.648 113.471C191.648 112.786 191.092 112.229 190.406 112.229C189.72 112.229 189.164 112.786 189.164 113.471C189.164 114.157 189.72 114.713 190.406 114.713Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M183.4 99.9029C184.086 99.9029 184.642 99.3469 184.642 98.6609C184.642 97.975 184.086 97.4189 183.4 97.4189C182.714 97.4189 182.158 97.975 182.158 98.6609C182.158 99.3469 182.714 99.9029 183.4 99.9029Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M190.406 130.329C191.092 130.329 191.648 129.773 191.648 129.087C191.648 128.401 191.092 127.845 190.406 127.845C189.72 127.845 189.164 128.401 189.164 129.087C189.164 129.773 189.72 130.329 190.406 130.329Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M197.098 106.978C198.14 106.978 198.985 106.133 198.985 105.091C198.985 104.049 198.14 103.204 197.098 103.204C196.056 103.204 195.211 104.049 195.211

105.091C195.211 106.133 196.056 106.978 197.098 106.978Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M203.76 121.53C204.802 121.53 205.647 120.685 205.647 119.643C205.647 118.601 204.802 117.756 203.76 117.756C202.718 117.756 201.873 118.601 201.873 119.643C201.873 120.685 202.718 121.53 203.76 121.53Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M177.68 147.939C178.722 147.939 179.567 147.094 179.567 146.052C179.567 145.01 178.722 144.165 177.68 144.165C176.638 144.165 175.793 145.01 175.793 146.052C175.793 147.094 176.638 147.939 177.68 147.939Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M187.276 86.723C188.318 86.723 189.162 85.8782 189.162 84.8361C189.162 83.794 188.318 82.9492 187.276 82.9492C186.233 82.9492 185.389 83.794 185.389 84.8361C185.389 85.8782 186.233 86.723 187.276 86.723Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M194.326 146.056C195.368 146.056 196.213 145.211 196.213 144.169C196.213 143.127 195.368 142.282 194.326 142.282C193.284 142.282 192.439 143.127 192.439 144.169C192.439 145.211 193.284 146.056 194.326 146.056Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M703.625 376.271H626.957C609.57 376.271 595.475 362.176 595.475 344.788V344.788C595.475 327.401 609.57 313.306 626.957 313.306H703.625C721.012 313.306 735.107 327.401 735.107 344.788V344.788C735.107 362.176 721.012 376.271 703.625 376.271Z" fill="#B5CDFB"></path>
<path d="M677.02 344.788H665.332C647.945 344.788 633.85 330.692 633.85 313.305V313.305C633.85 295.918 647.945 281.822 665.332 281.822H677.02C694.408 281.822 708.503 295.918 708.503 313.305V313.305C708.503 330.692 694.408 344.788 677.02 344.788Z" fill="#B5CDFB"></path>
<path d="M682.765 292.521C683.451 292.521 684.007 291.965 684.007 291.279C684.007 290.593 683.451 290.037 682.765 290.037C682.079 290.037 681.523 290.593 681.523 291.279C681.523 291.965 682.079 292.521 682.765 292.521Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M693.037 299.632C693.723 299.632 694.279 299.076 694.279 298.39C694.279

297.704 693.723 297.148 693.037 297.148C692.351 297.148 691.795 297.704 691.795 298.39C691.795 299.076 692.351 299.632 693.037 299.632Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M683.385 303.645C684.07 303.645 684.627 303.089 684.627 302.403C684.627 301.717 684.07 301.161 683.385 301.161C682.699 301.161 682.143 301.717 682.143 302.403C682.143 303.089 682.699 303.645 683.385 303.645Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M724.99 359.778C725.676 359.778 726.232 359.222 726.232 358.536C726.232 357.85 725.676 357.294 724.99 357.294C724.304 357.294 723.748 357.85 723.748 358.536C723.748 359.222 724.304 359.778 724.99 359.778Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M711.631 337.899C712.317 337.899 712.873 337.343 712.873 336.657C712.873 335.971 712.317 335.415 711.631 335.415C710.945 335.415 710.389 335.971 710.389 336.657C710.389 337.343 710.945 337.899 711.631 337.899Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M704.625 323.095C705.311 323.095 705.867 322.539 705.867 321.853C705.867 321.167 705.311 320.611 704.625 320.611C703.939 320.611 703.383 321.167 703.383 321.853C703.383 322.539 703.939 323.095 704.625 323.095Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M711.631 353.523C712.317 353.523 712.873 352.967 712.873 352.281C712.873 351.595 712.317 351.039 711.631 351.039C710.945 351.039 710.389 351.595 710.389 352.281C710.389 352.967 710.945 353.523 711.631 353.523Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M718.324 330.158C719.366 330.158 720.211 329.314 720.211 328.272C720.211 327.23 719.366 326.385 718.324 326.385C717.282 326.385 716.438 327.23 716.438 328.272C716.438 329.314 717.282 330.158 718.324 330.158Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M724.988 344.71C726.031 344.71 726.875 343.865 726.875 342.823C726.875 341.781 726.031 340.937 724.988 340.937C723.946 340.937 723.102 341.781 723.102 342.823C723.102 343.865 723.946 344.71 724.988 344.71Z" fill="none" stroke="#221b38" strokeWidth="1"></path>

```
<path d="M698.906 371.118C699.949 371.118 700.793 370.274 700.793 369.232C700.793 368.19 699.949 367.345 698.906 367.345C697.864 367.345 697.02 368.19 697.02 369.232C697.02 370.274 697.864 371.118 698.906 371.118Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M698.906 309.91C699.949 309.91 700.793 309.066 700.793 308.024C700.793 306.981 699.949 306.137 698.906 306.137C697.864 306.137 697.02 306.981 697.02 308.024C697.02 309.066 697.864 309.91 698.906 309.91Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M715.553 369.237C716.595 369.237 717.44 368.392 717.44 367.35C717.44 366.308 716.595 365.463 715.553 365.463C714.511 365.463 713.666 366.308 713.666 367.35C713.666 368.392 714.511 369.237 715.553 369.237Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
</svg>
<span>Loading...</span>
</StyledSpinner>
)
export default Spinner;
```

Spinner Container

```
import React from 'react';
import { StyledSpinnerContainer } from '../../styles';

const SpinnerContainer = () => (
<StyledSpinnerContainer>
<div className="loader"></div>
<span>Loading...</span>
</StyledSpinnerContainer>
)
export default SpinnerContainer;
```

5.20. Styled Spinner Container

```
import styled from 'styled-components';

const StyledSpinnerContainer = styled.div`
text-align:center;
display:flex;
justify-content:center;
align-items:center;
flex-direction: column;
height: 80vh;
width:100%;
span{
font-size:15px;
padding-top:15px;
color:${props => props.theme.textColor};
font-family: "Averta-Regular";
font-style:italic;
}
.loader{
border: 5px solid #f3f3f3; /* Light grey */
border-top: 5px solid ${props => props.theme.textColor};
border-radius: 50%;
width: 20px;
height: 20px;
animation: spin 0.8s linear infinite;
margin: 20px auto;
@keyframes spin {
0% {
transform: rotate(0deg);
}
100% {
transform: rotate(360deg);
}
}
}
`;

export default StyledSpinnerContainer;
```

5.21 Package.json

```
{
  "name": "weather-app",
  "private": true,
  "version": "0.0.0",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "@chakra-ui/react": "^1.8.8",
    "@emotion/react": "^11.9.0",
    "@emotion/styled": "^11.8.1",
    "axios": "^0.27.2",
    "framer-motion": "^6.3.3",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-icons": "^4.3.1",
    "react-redux": "^8.0.1",
    "redux": "^4.1.2",
    "redux-thunk": "^2.4.1"
  },
  "devDependencies": {
    "@types/react": "^17.0.2",
    "@types/react-dom": "^17.0.2",
    "@vitejs/plugin-react": "^1.3.0",
    "vite": "^2.9.5"
  }
}
```

5.22 API.js

```
export const weatherAppAPI = "c0d290eeee9dd399b017a6d2ba64be7e";

export const googleMapAPI = "AIzaSyAq15HbfCRMW7RqNb5LUNyOLyfzpYI0wl4";
```

5.23 extraFunction.js

```
export const celsius = (x) => (x - 273).toFixed(2);

export const myToast = (toast, title, status, description) => toast({
    title,
    description,
    status,
    duration: 5000,
    isClosable: true,
});


export const dateFormat = (dt) => {

    const milliseconds = dt * 1000;

    let myDate = new Date(milliseconds);

    let date = myDate.toLocaleString('en-GB').split(",")[0];

    let day = myDate.toLocaleString("en-US", { weekday: "long" });
    return { date, day };
}
```

5.24 sessionStorage.js

```
export const getItem = (key) => {

    if (sessionStorage.getItem(key)) {

        return JSON.parse(sessionStorage.getItem(key));

    } else {
        return undefined;
    }
};

export const setItem = (key, data) => {

    return sessionStorage.setItem(key, JSON.stringify(data));
};
```

5.25 vite.config.js

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'


// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()]
})
```

5.26 package-lock.json

```
{
  "name": "Weather-App-master",
  "lockfileVersion": 2,
  "requires": true,
  "packages": {}}
```

CHAPTER 6

TESTING

1.Testing

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification.

- **Unit Testing**

  While coding, the programmer performs some tests on that unit of program to know if it is error free. Testing is performed under white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

- **Integration Testing**

  Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passes and data updating etc.

- **System Testing**

  The software is compiled as product and then it is tested. This can be accomplished using one or more of the following tests:

- **Functionality testing**

  Tests all functionalities of the software against the requirement.

- **Performance testing**

  This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.

- **Acceptance Testing**

  When the software is ready to hand over to the customer it must go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if user does not like the way it appears or works, it may be rejected.

- **Alpha testing**

  The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find out how user would react to some action in software and how the system should respond to inputs.

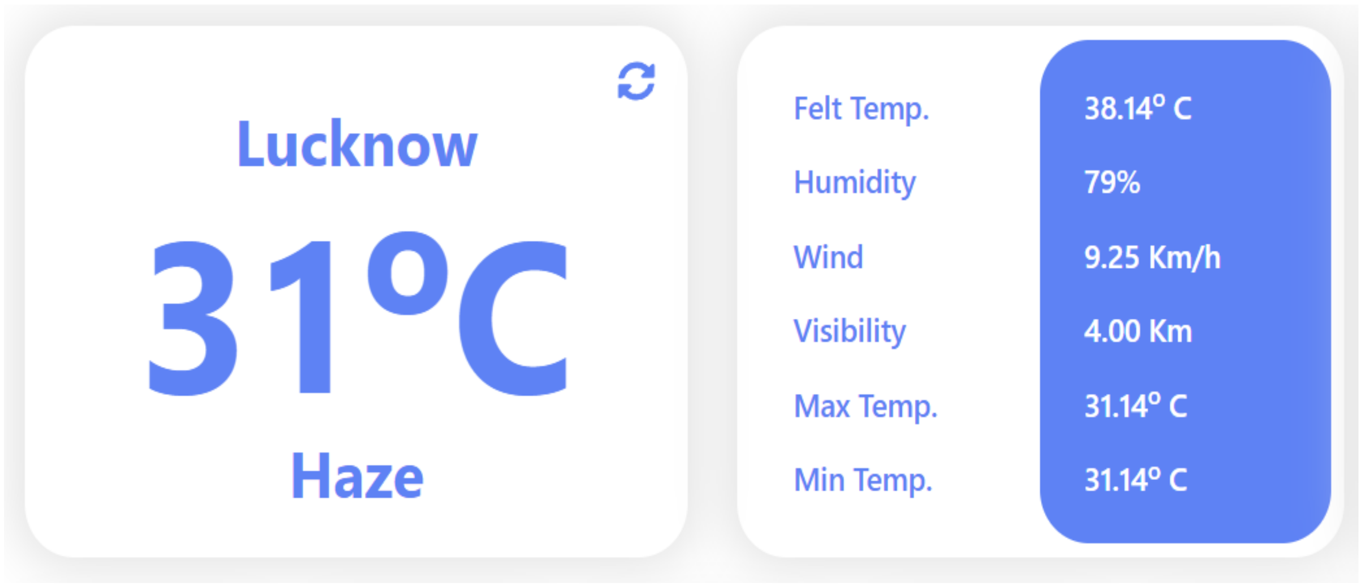
- **Beta testing**

  After the software is tested internally, it is handed over to the users to use it under their production environment only for testing purpose. This is not yet the delivered product. Developers expect that users at this stage will bring minute problems, which were skipped to attend.

2. Test cases-1

Functional Test Cases

- It should also shows the Precipitation, humidity and wind percentage apart from the temperature.



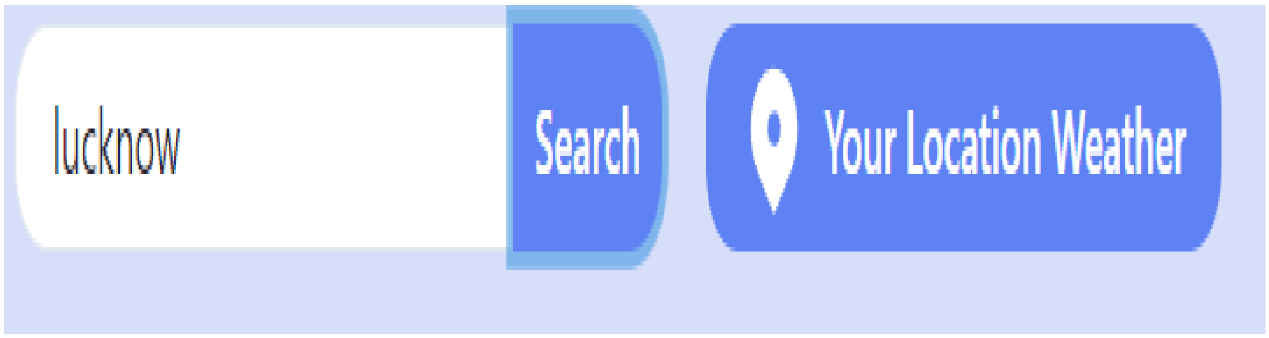
Lucknow

31°C

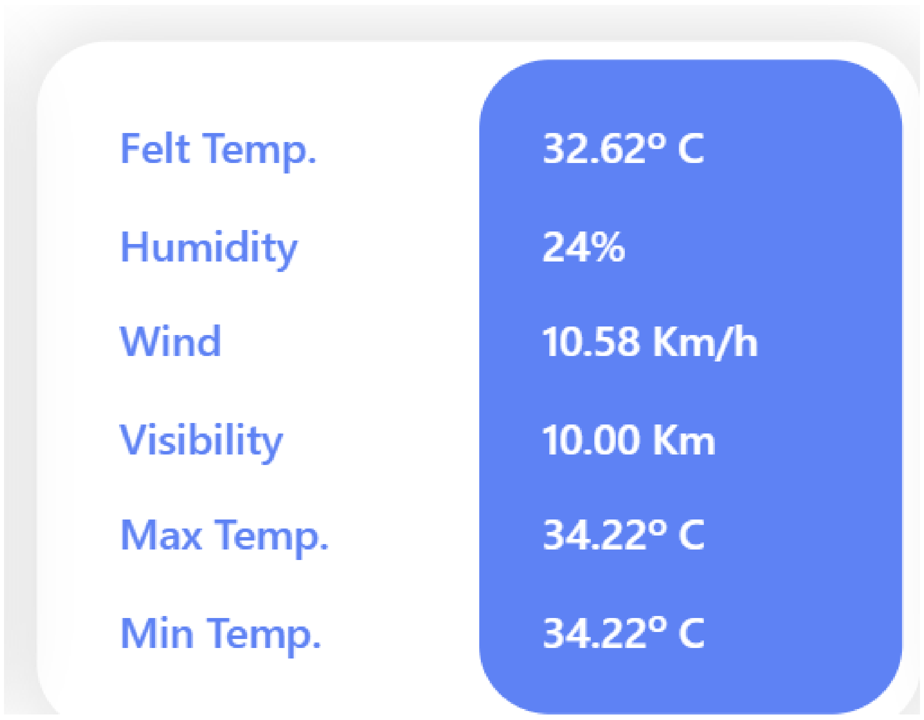
Haze

Felt Temp.	38.14° C
Humidity	79%
Wind	9.25 Km/h
Visibility	4.00 Km
Max Temp.	31.14° C
Min Temp.	31.14° C

- There should be search fields to find weather for other places.



- Weather forecast should also display the minimum and maximum temperature of the day.

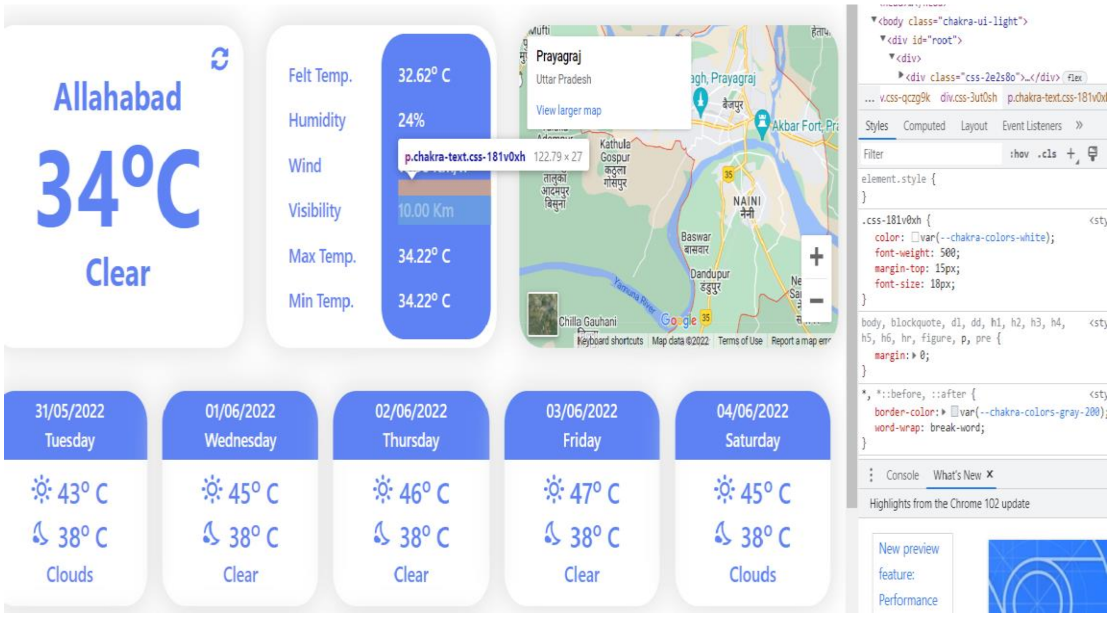


Felt Temp.	32.62° C
Humidity	24%
Wind	10.58 Km/h
Visibility	10.00 Km
Max Temp.	34.22° C
Min Temp.	34.22° C

- It will show the map also for any city you search in box.

Prayagraj
Uttar Pradesh
View larger map
Directions
Akbar Fort, Prayagraj
NAINI
नैनी
Baswar
Dandupur
Chilla Gauhani
Newada Samogar
Kathula Gospur
Adampur Bisuna
Yamuna River
Keyboard shortcuts
Map data ©2022
Terms of Use
Report a map error

For Tablet



Allahabad

34°C

Clear

Felt Temp.	32.62° C
Humidity	24%
Wind	
Visibility	10.00 Km
Max Temp.	34.22° C
Min Temp.	34.22° C

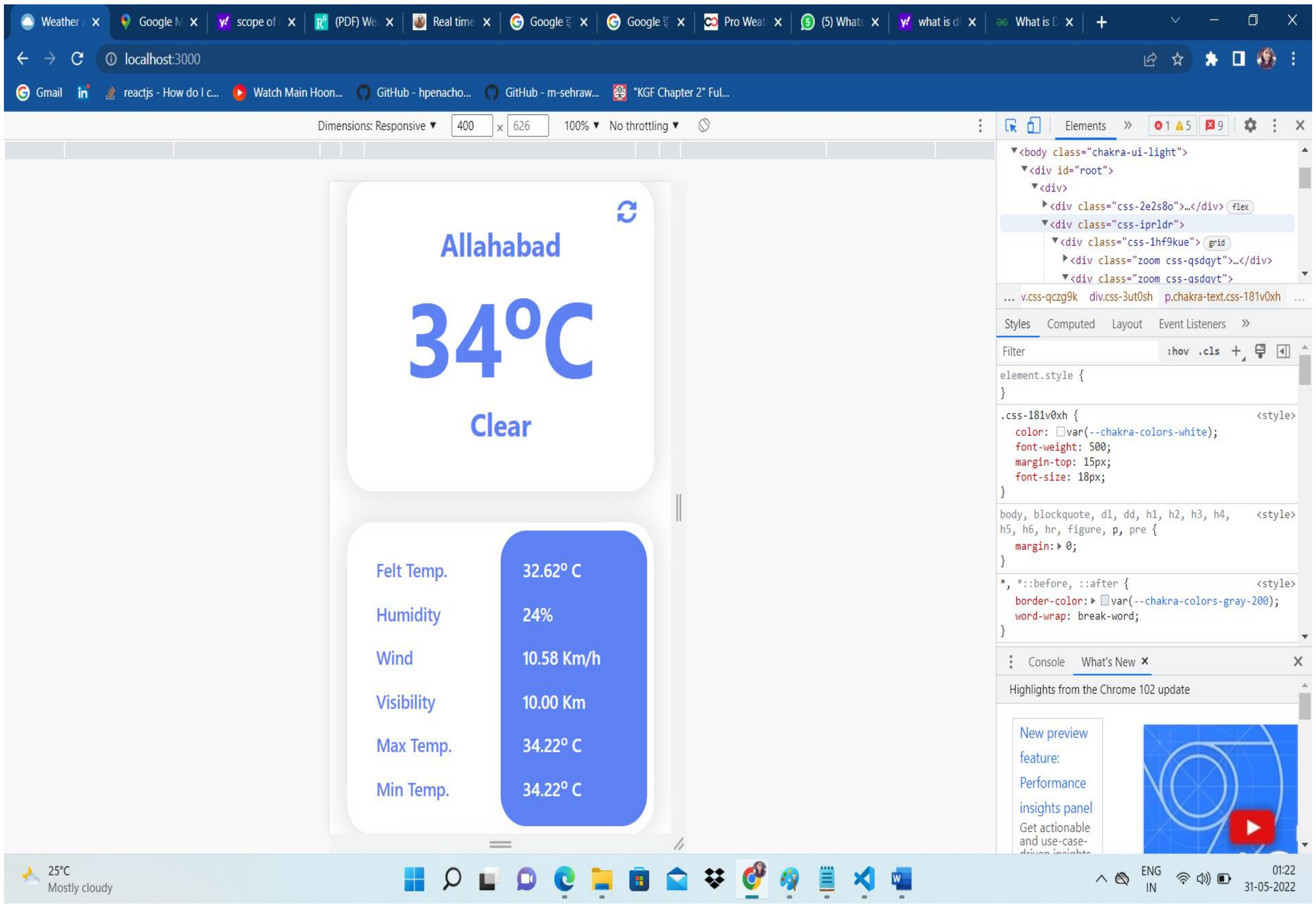
p.chakra-text.css-181v0xh 122.79 × 27

31/05/2022 Tuesday	01/06/2022 Wednesday	02/06/2022 Thursday	03/06/2022 Friday	04/06/2022 Saturday
43° C	45° C	46° C	47° C	45° C
38° C	38° C	38° C	38° C	38° C
Clouds	Clear	Clear	Clear	Clouds

```
▼<body class="chakra-ui-light">
  ▼<div id="root">
    ▼<div>
      ▶<div class="css-2e2s8o">…</div> flex
... v.css-qczg9k  div.css-3ut0sh  p.chakra-text.css-181v0x

Styles  Computed  Layout  Event Listeners  »
Filter                          :hov .cls +
element.style {
}
.css-181v0xh {                                  <sty
  color: var(--chakra-colors-white);
  font-weight: 500;
  margin-top: 15px;
  font-size: 18px;
}
body, blockquote, dl, dd, h1, h2, h3, h4,      <sty
h5, h6, hr, figure, p, pre {
  margin: ▶ 0;
}
*, *::before, ::after {                         <sty
  border-color: ▶ var(--chakra-colors-gray-200);
  word-wrap: break-word;
}
⋮  Console  What's New ×
Highlights from the Chrome 102 update
New preview feature: Performance
```

For Mobiles



Allahabad

34°C

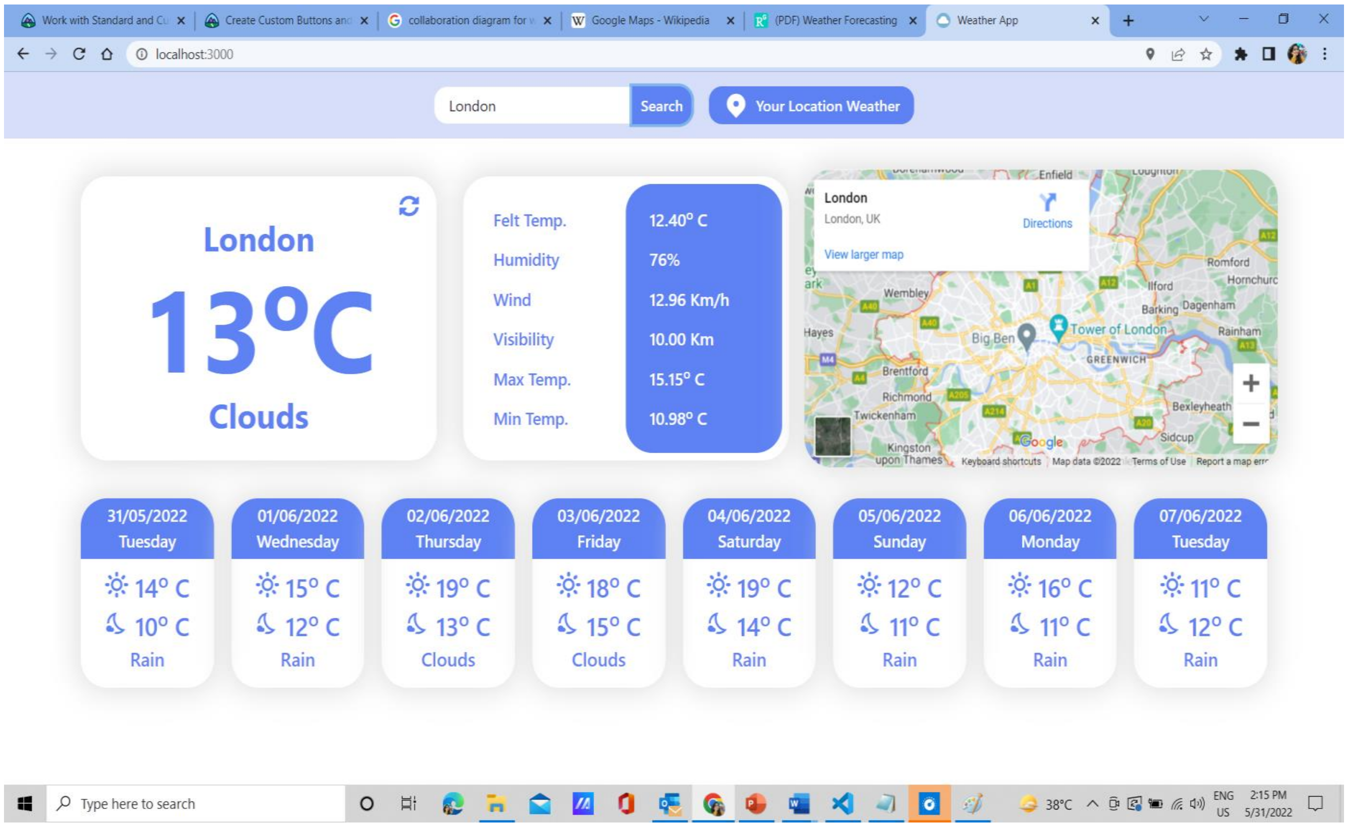
Clear

Felt Temp.	32.62° C
Humidity	24%
Wind	10.58 Km/h
Visibility	10.00 Km
Max Temp.	34.22° C
Min Temp.	34.22° C

```
▼<body class="chakra-ui-light">
  ▼<div id="root">
    ▼<div>
      ▶<div class="css-2e2s8o">…</div> flex
      ▼<div class="css-iprldr">
        ▼<div class="css-1hf9kue"> grid
          ▶<div class="zoom css-qsdqyt">…</div>
          ▼<div class="zoom css-qsdqyt">
```

```
element.style {
}
.css-181v0xh {                                        <style>
  color: var(--chakra-colors-white);
  font-weight: 500;
  margin-top: 15px;
  font-size: 18px;
}
body, blockquote, dl, dd, h1, h2, h3, h4,             <style>
h5, h6, hr, figure, p, pre {
  margin: ▶ 0;
}
*, *::before, ::after {                               <style>
  border-color: ▶ var(--chakra-colors-gray-200);
  word-wrap: break-word;
}
```

Highlights from the Chrome 102 update



London

13°C

Clouds

Felt Temp.	12.40° C
Humidity	76%
Wind	12.96 Km/h
Visibility	10.00 Km
Max Temp.	15.15° C
Min Temp.	10.98° C

31/05/2022 Tuesday	01/06/2022 Wednesday	02/06/2022 Thursday	03/06/2022 Friday	04/06/2022 Saturday	05/06/2022 Sunday	06/06/2022 Monday	07/06/2022 Tuesday
14° C	15° C	19° C	18° C	19° C	12° C	16° C	11° C
10° C	12° C	13° C	15° C	14° C	11° C	11° C	12° C
Rain	Rain	Clouds	Clouds	Rain	Rain	Rain	Rain

CHAPTER 7

BIBLIOGRAPHY

1. **Weather API:-** **https://api.openweathermap.org/data/2.5/onecall?exclude=minutely&appid=${API_APPID}&lat={lat}&lon={lon}`**

2. **- https://www.weather.gov/enterprise/fi-app-2d**https://www.simplifiedcoding.net/