WEATHER FORECASTING APPLICATION

A PROJECT REPORT

Submitted by

Pranav Chaurasiya(21BCS11795)

Dharmendra Yadav(21BCS11791)

In the partial fulfilment for the award of the degree of

BACHELOR OF ENGINNERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

February 2023

BONAFIDE CERTIFICATE

Certified that this project report "**WEATHER FORECASTING APPLICATION**" is the bonafide work of "Dharmendra Yadav and Pranav Chaurasiya" who carried out the project work under my/our supervision.

SIGNATURE

Dr. Ajay Kumar Singh

HEAD OF THE DEPARTMENT

(Computer Science & Engineering)

SIGNATURE

Er. Amandeep Kaur

SUPERVISOR

(Computer Science & Engineering)

Submitted for the project viva-voice examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

The success and final outcome of our project work required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project work. All that we have done is only due to such supervision and encouragement and we wish to convey our profound thanks to each of them for their exemplary contribution towards the completion of this project.

We also wish to express our gratitude to our project In-Charge **Er. AMANDEEP KAUR** and all the members of our institute, not only for giving us the chance to collaborate with them on this project, but also for their encouragement and support all along the way. The project was incredibly fruitful and enriching, both technically and in terms of the development of transferable skills.

Finally, we acknowledge the people who mean a lot to us, our parents, for their inspiration, unconditional love, support, and faith for carrying out this work to the finishing line. We want to give special thanks to all our friends who went through hard times together, cheered us on, helped us a lot, and celebrated eachaccomplishment.

DECLARATION

We, student of **Bachelor of Engineering** in **Computer Science & Engineering, 4th Semester**, **session: Feb-June, Chandigarh University,** hereby declare that the work presented in this Project Report entitled **"WEATHER FORECASTING APPLICATION"** is the outcome of our own work, is bonafide and correct to the best of my knowledge and this work has been carried out taking care of Engineering Ethics. The work presented does not infringe any patented work and has not been submitted to any other university or anywhere else for the award of any degree or any professional diploma.

Student details and Signature

Dharmendra Yadav (21BCS11791)

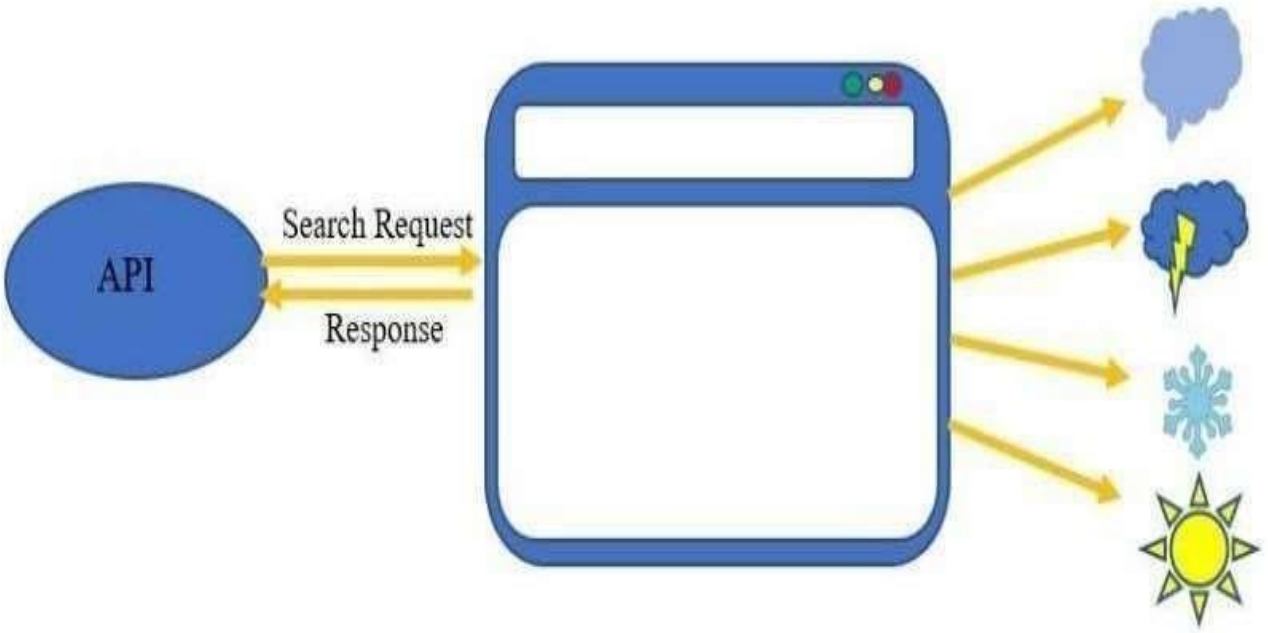
Pranav Chaurasiya (21BCS11795)

TABLE OF CONTENTS

ABSTRACT

The weather forecasting web application project aims to provide accurate and up-to- date weather information based on the user's latitude and longitude. The application willutilize APIs from reliable weather data sources to collect real-time weather data and display it in a user-friendly format. Users will be able to enter their location by inputting their latitude and longitude coordinates, and the application will display the current weather conditions, as well as a five-day forecast. The user interface will be designed to be intuitive and easy to use, with simple navigation and clear visualizations.The project will also incorporate features such as notifications for severe weather alerts and the ability to save favorite locations for quick access. The application will be accessible from any web browser, making it easily accessible to users on any device. Overall, this project aims to provide users with an efficient and reliable tool for accessing weather information in real-time, ensuring they are prepared for any weather conditions.

GRAPHICAL ABSTRACT



CHAPTER 1

INTRODUCTION

1.5 Identification of the Client needs

As a weather forecasting web app, there are several Client needs that you should consider. Here are a few key considerations:

Accurate and up-to-date weather information: One of the primary needs of your clients
is accurate and reliable weather information. Make sure that your app provides the latest weather forecasts and updates in real-time.

User-friendly interface: Your app should have a user-friendly interface that is easy to navigate and understand. Provide clear and concise information, as well as visual aids like maps and graphs to help users understand the weather patterns.

Personalization options: Allow users to personalize their experience by letting them choose the locations they want to track, set up alerts, and receive notifications for severe weather conditions.

Historical data: Some users may want to access historical weather data to make comparisons or plan future activities. Consider providing this data, along with forecasts.

1.6 Identification of Problem

Identifying a problem in Weather Forecasting Application can be a complex task that requires careful analysis and investigation. However, some common issues that may arise in crowd funding platform include:

Accuracy: The primary goal of a weather forecasting app is to provide accurate weather predictions to its users. The identification of the problem in this area may include challenges in accurately predicting weather conditions, lack of reliable data sources, or other factors that may impact the app's ability to provide accurate information.

User Experience: A weather forecasting app must be user-friendly, visually appealing, and provide a good user experience. The identification of the problem in this area may include issues with the app's interface, design, or features that negatively impact the user experience.

Technical Challenges: Developing a weather forecasting app may require integrating various technologies, including APIs, data storage systems, and advanced algorithms. The identification of the problem in this area may include technical difficulties, integration issues, or other technical challenges that may arise during the development process.

Data Sources: Weather forecasting apps rely on various data sources, including satellite imagery, radar data, and other weather-related data. The identification of the problem in this area may include issues with data quality, availability, or access to data sources.

Scalability: As the app gains popularity and users, it must be able to handle the increased load and traffic. The identification of the problem in this area may include scalability issues, server overload, or other performance-related issues.

Competition: Weather forecasting apps are a popular category in app stores, and there may be several competitors in the market. The identification of the problem in this area may include challenges in differentiating the app from its competitors or providing unique features

1.7 Identification of Tasks

Identifying a problem in Weather Forecasting Application can be a complex task that requires careful analysis and investigation. However, some common issues that may arise in crowd funding platform include:

Research: Conducting research on weather forecasting technologies, data sources, and user needs to inform the development of the app.

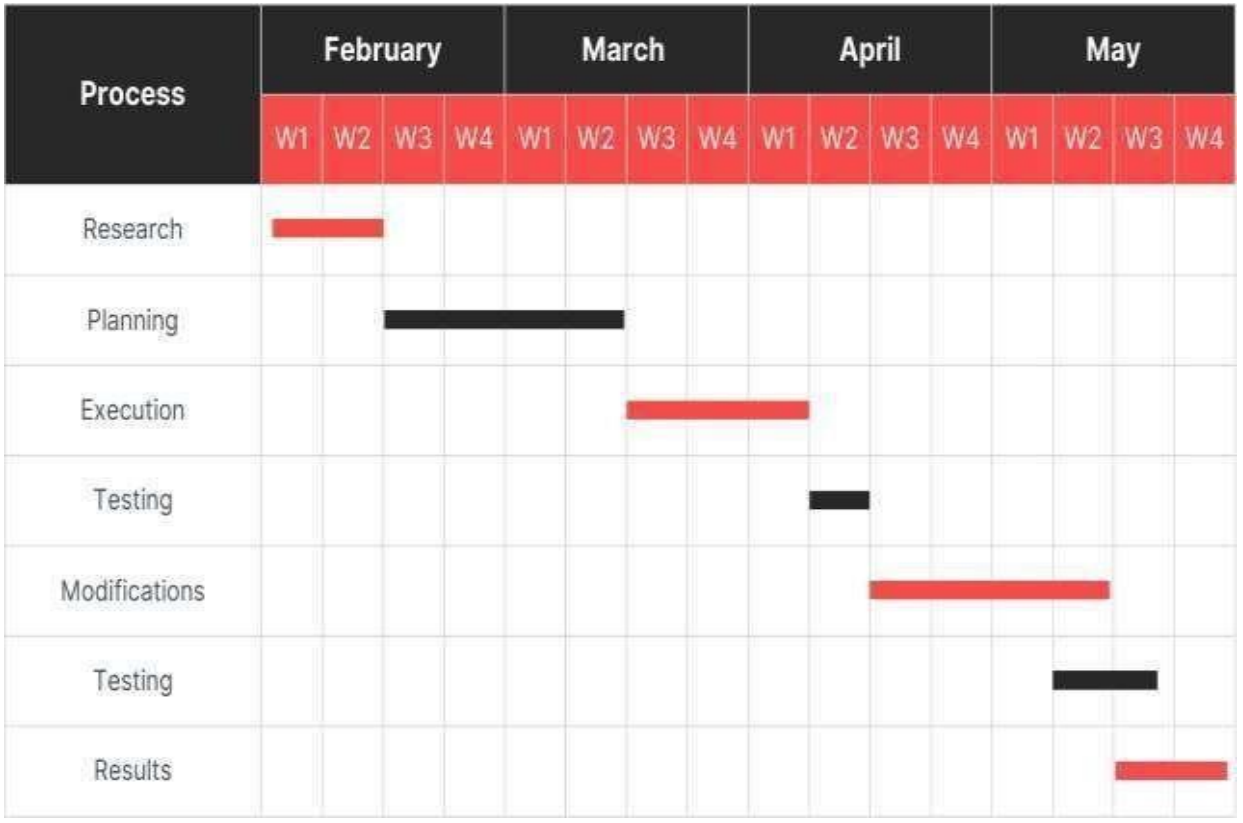
Design: Designing the user interface and user experience of the app, including wireframing and prototyping**.**

Development: Developing the app using programming languages, frameworks, and libraries that are appropriate for the project

Integration: Integrating various technologies, such as APIs, data storage systems, and weather forecasting algorithms, to ensure the app's accuracy and reliability.

Testing: Conducting various tests, such as unit testing, integration testing, and user testing, to ensure that the app functions correctly and provides accurate weather predictions.

1.8 Timeline


Process
February
March
April
May
W1 W2 W3 W4 W1 W2 W3 W4 W1 W2 W3 W4 W1 W2 W3 W4
Research
Planning
Execution
Testing
Modifications
Testing
Results

1.9 Organization of the Report

Chapter 1: Introduction

This chapter introduces the report, including background information, objectives, and scope of the study. It also outlines the methodology used to gather data and the structure of the report.

Chapter 2: Literature Review

This chapter presents a comprehensive review of the literature related to crowdfunding platform. It covers the benefits and challenges of these systems, as wellas the various models and technologies that have been used to implement them.

Chapter 3: Methodology

This chapter describes the research methodology used to gather data for the study. It also outlines the data collection methods, data analysis techniques, and the limitations of the study.

Chapter 4: Findings

This chapter presents the findings of the study, including the current state of crowdfunding platform among its users and the potential barriers to implementation.

Chapter 5: Discussion

This chapter provides a critical analysis of the findings and discusses their implications for the customers. It also provides recommendations for implementing an unimplemented feature and discusses the potential benefits and challenges of such a system.

Chapter 6: Conclusion

This chapter summarizes the key findings of the study and provides a conclusion based on these findings. It also highlights the contributions of the study and outlinesthe directions for future research in this area.

Chapter 7: References

This chapter provides a list of the references used in the report.

CHAPTER 2

LITERATURE REVIEW/BACKGROUND STUDY

On a worldwide scale, large numbers of attempts have been made by different researchers to forecast Weather accurately using various techniques. But due to the nonlinear nature of Weather, prediction accuracy obtained by these techniques is still below the satisfactory level. Ali, Lin etc. [3] developed an ANN technique to estimate tropical cyclone heat potential (TCHP) for estimating the Cyclone and Intensity prediction.

Twenty years of daily precipitation data were used to train the networks while ten years of daily precipitation data were used to test the effectiveness of the models. They found that the models were able to predict the occurrence of daily precipitation with an accuracy of 79±3% and Fuzzy classification produced a higher accuracy in predicting 'trace' precipitation than other categories [5].

In the case study of The chaotic time series of Indian monsoon rainfall, Basu and Andharia,1992,have found that the resulting forecast formula uses only the rainfall of past seven years as predictors, making a forecast eight months in advance[9].

After comparative study of short term rainfall prediction models for real time flood forecasting, E. Toth et al., have found that the time series analysis technique based on ANN provides significant improvement in the flood forecasting accuracy in comparison to the use of simple rainfall prediction approaches[8].

Karmakar et.al., 2008, et al., 2008, have developed the ANN models for Long-Range Meteorological Parameters Pattern Recognition over the Smaller Scale Geographical Region and the performances of these models in pattern recognition and prediction have been found to be extremely good [6].

Literature research to classify our research with existing knowledge

- Timeline of reported issues
- Recommended solution
- Literature analysis
- Summary of the review

2.1 Timeline of the reported problem:

The problem of weather forecasting has been recognized for centuries, with early attempts to predict the weather dating back to ancient civilizations such as the Greeks and Chinese. However, modern weather forecasting as we know it today has only been developed in the last few centuries.

Rainfall prediction is one of the most important and challenging Task in the modern world. ANN has been successfully used by most of the researchers in this field for the lasttwenty-five years. [3] Provides a survey of available literature of some methodologies employed by different researchers to utilize ANN for rainfall prediction. From the survey it has been found that most of the researchers used back propagation network for rainfall prediction and got significant results. The survey reports that rainfall prediction using ANN technique is more suitable and also gives a conclusion that the forecasting techniques that use MLP, BPN, RBFN, SOM and SVM are suitable to predict rainfall than other forecasting techniques such as statistical and numerical methods.

One of the most significant milestones in the history of weather forecasting was the establishment of the first national meteorological service by the United Kingdom in 1854. This service was created to provide information about weather conditions to the maritime industry, which was vital for navigation and safety at sea. Other countries followed suit in establishing their own national meteorological services, and the science of weather forecasting began to advance rapidly.

In the early 20th century, the development of new technologies such as radiosondes, weather satellites, and computer models greatly improved the accuracy of weather forecasts. In the 1950s and 1960s, the United States and other countries established the World Meteorological Organization (WMO) to coordinate international efforts in weather forecasting and monitoring.

In recent years, climate change has become an increasingly urgent concern, and weather forecasting has played a crucial role in monitoring and predicting the effects of climate change. Documentary evidence of incidents related to weather forecasting can be found in various sources, including historical records, scientific publications, and news reports. Some notable events in the history of weather forecasting include the development of the first numerical weather prediction models in the 1950s, the launch of the first weather satellite in 1960, and the advent of Doppler radar in the 1980s.

2.2 Existing Solutions:

Weather forecasting is a complex process that involves analysing data from varioussources such as weather satellites, radar, and ground-based weather stations. The goal isto provide accurate predictions of future weather conditions. There are several existing solutions for weather forecasting:

1. National Weather Service (NWS): This is a government agency responsible for providing weather forecasts and warnings to the public. The NWS uses a combination of data from weather satellites, radar, and ground-based weather stations to create forecasts.

2. AccuWeather: This is a private weather forecasting company that uses advanced technology to provide weather forecasts. AccuWeather uses a proprietary forecasting system called "AccuWeather Real Feel® Temperature" which takes into account various factors such as humidity, wind, and sunshine to create a more accurate forecast.

3. The Weather Channel: This is a popular weather forecasting company that provides weather forecasts and warnings to the public. The Weather Channel uses a combination of data from weather satellites, radar, and ground-based weather stations to create forecasts.

4. Weather Underground: This is a popular weather forecasting website that provides weather forecasts and warnings to the public. Weather Underground uses a combination of data from weather satellites, radar, and ground-based weather stations to create forecasts. They also have a network of personal weather stations that contribute to their forecasts.

5. Dark Sky: This is a weather forecasting app that uses machine learning algorithms to provide hyperlocal weather forecasts. Dark Sky uses data from weather satellites, radar, and ground-based weather stations to create forecasts. They also have a network of personal weather stations that contribute to their forecasts.

These are just a few examples of the many existing solutions for weather forecasting. Each solution uses a combination of data sources and algorithms to create forecasts, and their accuracy can vary depending on various factors such
as the location and the time of year.

2.3 Bibliometric Analysis:

Bibliometric analysis is a quantitative method used to evaluate and measure the research output and impact of scientific publications. In the field of weather forecasting, bibliometric analysis can provide insights into the research trends, influential researchers, and major research topics.

One study that conducted a bibliometric analysis of weather forecasting was published in the journal Weather in 2016. The study analyzed publications on weather forecasting from the Web of Science database from 1900 to 2014.

The study found that the number of publications on weather forecasting has increased significantly over time, with a major increase in the 1960s and 1970s. The study also found that the United States and the United Kingdom were the leading countries in weather forecasting research, followed by Germany and France.

In terms of research topics, the study found that numerical weather prediction, atmospheric modeling, and climate change were the most popular topics. The study also identified influential researchers in the field, including Edward Lorenz, who developed the concept of the "butterfly effect" in weather forecasting.

Another bibliometric analysis of weather forecasting was published in the Journal of Meteorological Research in 2020. This study analyzed publications on weather forecasting from the Science Citation Index Expanded database from 1999 to 2018.

The study found that the number of publications on weather forecasting increased significantly during the study period, with a major increase in the number of publications on climate change and extreme weather events. The study also identified China as an emerging leader in weather forecasting research, with a significant increase in the number of publications from Chinese researchers.

Overall, bibliometric analysis provides valuable insights into the research output and impact of weather forecasting publications, including the identification of influentialresearchers, emerging research trends, and the geographic distribution of research activity.

2.4 Review Summary:

Weather forecasting applications have become an essential part of our daily lives, providing convenient access to weather information on our smartphones, tablets, and other devices. These applications offer a wide range of features, including current weatherconditions, hourly and daily forecasts, radar and satellite imagery, severe weather alerts, and more. They allow users to plan their activities, make informed travel decisions, and take appropriate precautions during extreme weather events.

Weather forecasting applications rely on a combination of meteorological data, computer models, and real-time observations to generate forecasts. They use sophisticated algorithms to analyze and interpret data from weather stations, satellites, and other sources to produce accurate and up-to-date weather predictions. These applications often incorporate user-friendly interfaces, providing intuitive and visually appealing displays of weather information.

One of the strengths of weather forecasting applications is their ability to provide hyper-local forecasts, allowing users to obtain weather information for their specific location. This can be particularly useful for planning outdoor activities, such as hiking, sports, or events, as it provides detailed weather insights for a particular area.

Weather forecasting applications also play a vital role in enhancing public safety by providing timely severe weather alerts, such as thunderstorm warnings, tornado warnings, and hurricane alerts. These notifications can help users take appropriate precautions and stay safe during hazardous weather conditions.

However, it's important to note that weather forecasting applications are not infallible and have limitations. Weather can be unpredictable, and forecast accuracy can vary depending on factors such as location, time frame, and the availability of data. Users should always exercise caution and verify weather information from multiple sources, especially during severe weather events.

In summary, weather forecasting applications have become indispensable tools for staying informed about weather conditions and planning our daily activities. They offer a wide range of features and provide convenient access to accurate weather information, helping users make informed decisions and stay safe in changing weather conditions.

2.5 Problem Definition:

The problem at hand is weather forecasting, which involves predicting the future state of the atmosphere by analyzing current and historical weather data.

The goal of weather forecasting is to provide accurate and timely information about upcoming weather conditions to help people make informed decisions about their daily activities, such as whether to bring an umbrella or cancel an outdoor event.

1. What is to be done?

Weather forecasting involves analyzing data, running simulations, and making predictions based on a wide range of meteorological factors such as temperature, humidity, airpressure, wind speed, and precipitation.

2. How it is to be done?

To do this effectively, weather forecasters need to have a deep understanding of atmospheric physics and be able to interpret complex data sets. They must also have access to sophisticated computer models and algorithms that can process vast amounts of data in real-time.

3. What not to be done?

However, there are also some things that should not be done in weather forecasting. For example, forecasters should avoid making predictions that are overly specific or that go beyond the limits of the available data. They should also be cautious about making predictions that could cause unnecessary alarm or panic, such as predicting severe weather events that are unlikely to occur. Additionally, forecasters should be transparent about the uncertainty inherent in weather forecasting and be clear about the limitations of their predictions.

2.6 Goals/Objectives:

The goals and objectives of weather applications are:

1. **Providing accurate and timely weather information:** The primary goal of weather applications is to provide users with accurate and up-to-date weather information. This includes information about current weather conditions, as well as forecasts for upcoming weather events.

2. **Enhancing safety and preparedness**: Weather applications can help users prepare for severe weather events by providing alerts and warnings for dangerous conditions such as hurricanes, tornadoes, and thunderstorms. This can help users stay safe and take appropriate action to protect themselves and their property.

3. **Supporting outdoor activities:** Weather applications can provide information about weather conditions that can impact outdoor activities such as hiking, camping, and sports. This can help users plan their activities and make informed decisions about when to participate in outdoor activities.

4. **Supporting travel:** Weather applications can provide information about weather conditions in different locations, which can be useful for travellers who need to plan their trips and pack accordingly.

5. **Providing educational resources:** Weather applications can provide educational resources such as explanations of weather phenomena, graphics and animations that illustrate weather patterns, and news and feature articles about weather-relatedtopics.

Overall, the goals and objectives of weather applications are to provide accurate and timely weather information, enhance safety and preparedness, support outdoor activities and travel, and provide educational resources for users.

CHAPTER 3

DESIGN FLOWS

3.1. Evaluation & Selection of Specification

Hardware used:-

Personal Computer/ laptop

Software and Technology/languages used:-

- Visual studio code
- HTML
- CSS
- JavaScript

Operating System

- Windows
- Mac

1. **Weather based on current location:** It refers to the current state of the weather of your current location, usually described in terms of various meteorological parameters (longitude & latitude). These parameters may include temperature, humidity, wind speed and direction, atmospheric pressure, cloud cover, and precipitation.

    - **Temperature:** This refers to the degree of hotness or coldness of the air, usually measured in Celsius or Fahrenheit. Temperature is a fundamental parameter for understanding current weather conditions and for predicting future weather.

    - **Humidity:** This refers to the amount of water vapor present in the air. It is usually expressed as a percentage of the maximum amount of water vapor that the air can hold at a given temperature.

    - **Wind speed and direction:** Wind speed is the speed at which the air is moving, usually measured in miles per hour (mph) or kilometers per hour (km/h). Wind

direction refers to the direction from which the wind is blowing, usually measured in degrees.

- **Atmospheric pressure:** This refers to the pressure exerted by the atmosphere on the earth's surface. It is usually measured in millibars (mb) or inches of mercury (inHg).

- **Precipitation**: This refers to any form of moisture that falls from the sky, including rain, snow, sleet, or hail. Precipitation is usually measured in millimeters or inches.

2. **Location-based Weather Forecast:** It refers to the current state of the weather at a specific location, usually described in terms of various meteorological parameters. These parameters may include temperature, humidity, wind speed and direction, atmospheric pressure, cloud cover, and precipitation.

3. **User Interface:** The user interface allows users to enter the name of a location or select a location from a list. The interface would also display weather information for the selected location, including current conditions, forecasts, and other relevant weather data.

4. **Data Accuracy and Availability:** Weather forecasting apps depend on accurate and reliable weather data to provide users with reliable forecasts. Therefore, the web app should be designed to ensure that it can access and process up-to-date weather data from reliable sources.

5. **Web Friendly** :It doesn't matter how informative, beautiful and user friendly your website design is if it isn't web friendly. It is important that the web designer understands the keys to making her website work in all major browsers, uses meta tags, alt tags and has knowledge about SEO (search engine optimization). Many factors affect search engine rankings and the appearance of a website, so web designers should know their stuff.

3.2 Design Constraints

Data Accuracy: The app should be designed to provide accurate and reliable weather data to users. This requires using reliable data sources and ensuring that the data is updated regularly.

User Experience: The app should be designed to provide a positive user experience, with an intuitive interface that is easy to use and understand.

Performance: The app should be designed to perform well, with fast load times and minimal lag when retrieving and displaying weather data.

Scalability: The app should be designed to scale as the user base grows, with the ability to handle large volumes of traffic and weather data.

Security: The app should be designed to be secure, with measures in place to protect user data and prevent unauthorized access.

Accessibility: The app should be designed to be accessible to users with disabilities, ensuring that all users can access and use the app.

Compatibility: The app should be designed to be compatible with a wide range of web browsers and devices, ensuring that users can access the app regardless of their platform.

Data Visualization: The app should be designed to present weather data in an easy-to-understand and visually appealing format, with charts, graphs, and other visualizations that help users make sense of the data.

3.3 Analysis and Features Finalization subject to Constraints:

HTML

HTML (Hypertext Markup Language) is the standard markup language used to create web pages and other web-based documents. Here are some advantages of using HTML:

Platform independent: HTML can be used on any platform such as Windows, Mac, Linux, etc. This means that web pages created using HTML can be accessed by anyone using any device or operating system.

Easy to learn and use: HTML is a simple and easy-to-learn language, making it accessible to beginners. It uses a straightforward syntax and is supported by numerous resources, making it an excellent language for web development.

Compatibility: HTML is compatible with all major web browsers. This means that web pages created using HTML can be viewed in any browser, without the need for additional plugins or software.

Accessibility: HTML provides a wide range of accessibility features that allow web pages to be accessed by people with disabilities, including the use of alt text for images, semantic markup for content, and ARIA (Accessible Rich Internet Applications) attributes.

SEO friendly: HTML is optimized for search engines, making it easier for search engines to index and rank web pages. This can help improve the visibility and accessibility of a website.

CSS

Since the UI is main part of the website as it make a website make look good. The language used to do so is CSS.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, etc.

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

The following are the advantages of CSS −

- CSS saves time − you can write CSS once and then reuse the same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- Easy maintenance − To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- Global web standards − Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's a good idea to start using CSS in all the HTML pages to make them compatible with future browsers.
- Platform Independence − The Script offer consistent platform independence and can support latest browsers as well.

JavaScript:-

JavaScript, often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries

Following are the advantages of JavaScript −

- Simple − JavaScript is simple to comprehend and pick up. Both users and developers will find the structure to be straightforward. Additionally, it is very doable to implement, saving web developers a ton of money when creating dynamic content.

- Speed − JavaScript is a "interpreted" language, it cuts down on the time needed for compilation in other programming languages like Java. Another client-side script is JavaScript, which accelerates programme execution by eliminating the wait time for server connections.
- No matter where JavaScript is hosted, it is always run in a client environment to reduce bandwidth usage and speed up execution.
- Interoperability − because JavaScript seamlessly integrates with other programming languages, many developers favor using it to create a variety of applications. Any webpage or the script of another programming language can contain it.
- Server Load − Data validation can be done within the browser itself rather than being forwarded to the server because JavaScript is client-side. The entire website does not need to be reloaded in the event of any discrepancy.

  Only the chosen area of the page is updated by the browser.

3.4 Design Flow

Weather forecasting works using an API

Here is an overview of how an API works in a weather forecasting web app:

Request: The weather forecasting web app sends a request to the weather data provider's API, specifying the location and time range for which weather data is required.

Processing: The weather data provider's API processes the request, retrieves the relevant weather data from its database, and formats the data into a standardized format, such as JSON (JavaScript Object Notation).

Response: The weather data provider's API sends the formatted weather data back to the weather forecasting web app in the requested format.

Integration: The weather forecasting web app integrates the weather data into its own data model, using JavaScript and presents the weather data to the user in a visually appealing and easily understandable way.

3.5 Design Selection

1. Simple and clean design: The design of the web app should be simple, clean, and easy to navigate. Users should be able to quickly find the information they are looking for without being overwhelmed by too much information or a cluttered interface.

2. Use of visual elements: Weather data can be complex, so the use of visual elements like icons, graphs, and charts can help users understand the information more easily. Use of color can also help convey information about the weather at a glance.

3. Responsive design: The web app should be designed to be responsive, meaning that it should look and function well on a variety of devices and screen sizes, from desktop computers to mobile phones.

4. Accurate and reliable data: The weather data presented in the web app is accurate and up-to-date, sourced from reliable weather data providers.

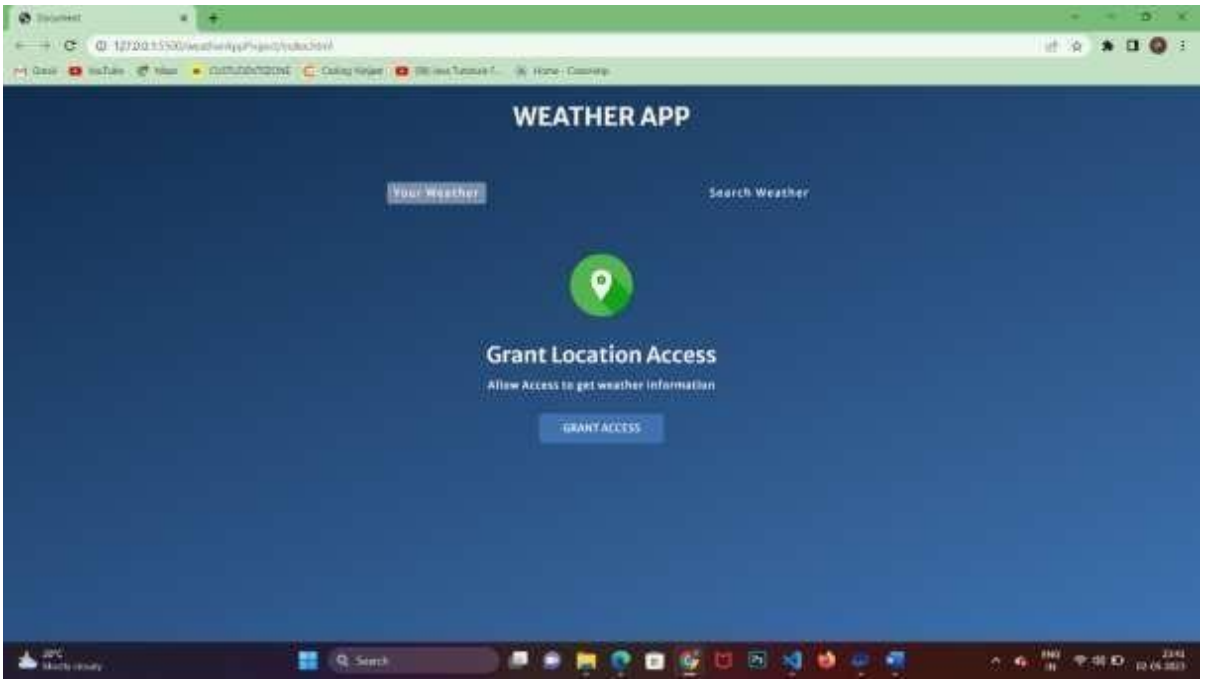
5. Interactive features: Interactive features like maps, animations, and radar provides users with a more engaging and immersive experience, helping them to better understand weather patterns and forecasts.

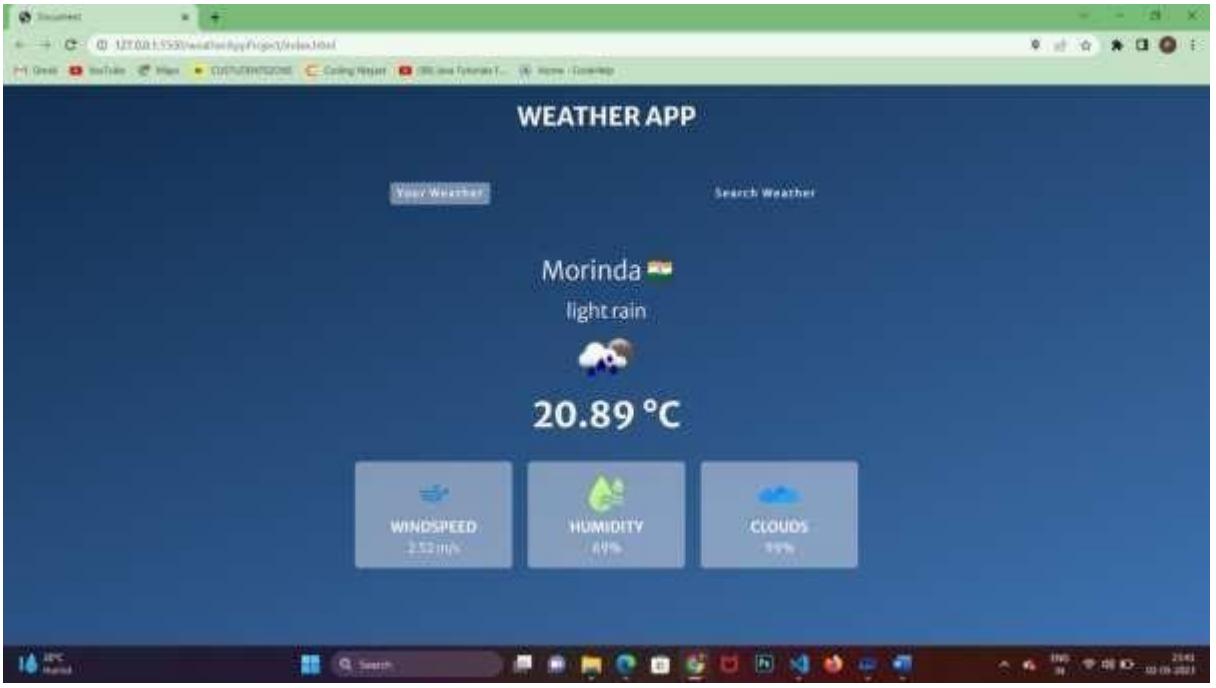
6. User-friendly interface: The interface of the web app is user-friendly, with clear labels, intuitive navigation, and helpful tips or guidance

3.6 Implementation Plan

There are two tabs Your Weather Tab and Search Weather Tab. If you click on Your Weather Tap, it will ask you grant location access if your location is not stored in session memory. After granting access, it will show your current weather based on your location.

By clicking on Search Weather Tab, a search bar will appear on the screen using which we can get the weather of the desired city.




WEATHER APP
Your Weather
Search Weather
Morinda
light rain
20.89 °C
WINDSPEED
HUMIDITY
CLOUDS


WEATHER APP
Your Weather
Search Weather
Search for City...


WEATHER APP
Your Weather
Search Weather
delhi
Delhi
mist
22.05 °C
WINDSPEED
HUMIDITY
CLOUDS

CHAPTER 4

RESULT ANALYSIS AND VALIDATION

4.1 Implementation of plan

Implementing a weather forecasting website typically involves the following steps:

1. **Get access to weather data:** You will need to obtain access to weather data in order to provide forecasts. There are several weather data APIs available, such as OpenWeather, Weather Underground, and Dark Sky. You will need to sign up for an account and get an API key to use their services.

2. **Choose a programming language and framework:** We will need to choose a programming language and framework to build your website. Some popular options include JavaScript with Node.js, Python with Django or Flask, and Ruby on Rails.

3. **Build the front-end:** Once we have chosen a programming language and framework, we can start building the front-end of your website. This will involve designing the user interface and creating HTML, CSS, and JavaScript code to display weather data.

4. **Integrate weather data API:** We will need to integrate the weather data API with your website. This will involve making API calls to retrieve weather data and parsing the JSON or XML response to extract the relevant information.

5. **Display weather data:** Once we have retrieved the weather data, we can display it on your website. This may involve creating charts or graphs to visualize the data, or simply displaying the current conditions and forecast for a given location.

6. **Add Additional features:** Depending on your requirements, you may want to add additional features to your website, such as the ability to search for weather forecasts by zip code or city, or to provide weather alerts and notifications.

7. **Test and deploy:** Once you have completed your website, you will need to test it thoroughly to ensure that it works as expected. You can then deploy your website to a web hosting provider to make it accessible to the public.

WeatherOpen API is a free API that provides access to weather data for locations around the world. The API provides a range of weather data, including current conditions, hourly forecasts, daily forecasts, and historical weather data.

To use the WeatherOpen API, you will first need to sign up for an API key. You can do this by visiting the WeatherOpen website and creating an account. Once you have an API key, you can use it to make API requests and retrieve weather data.

To make an API request, you will need to use the appropriate URL and query parameters. For example, to retrieve the current weather conditions for a specific location, you would use the following URL:

http://api.weatherunlocked.com/api/current/{latitude},{longitude}?app_id={app_id}&app_key={app_key}

In this URL, you would replace `{latitude}` and `{longitude}` with the latitude and longitude coordinates of the location you want to retrieve weather data for. You would also replace `{app_id}` and `{app_key}` with your own WeatherOpen API app ID and app key.

When you make this API request, you will receive a JSON response containing the current weather conditions for the specified location. The response will include information such as the temperature, humidity, wind speed and direction, and weather conditions (e.g. sunny, cloudy, rainy).

You can also use the WeatherOpen API to retrieve hourly and daily forecasts, as well as historical weather data for a specific date range. The API documentation provides more information on th e available endpoints and query parameters.

Overall, the WeatherOpen API is a useful resource for developers who want to incorporate weather data into their applications. By signing up for an API key and making API requests, you can retrieve accurate and up-to-date weather information for any location around the world.

Writing down the code forr the website that include CSS and Javascript.

After the study portion is over, the time has come to actually develop the website. For this, we require a few languages, specifically PHP, CSS, and JS.

To develop dynamic page content, php is used. It can also gather form data and send and receive cookies. It can create, open, read, write, remove, and close files on the server. It has the ability to alter and add to your database's data.

CSS styling gives the website a professional appearance. Font size, colour, and style are among them.

CSS is a stylesheet language used to describe how a document produced in HTML or XML (including XML dialects like SVG, Math ML, or XHTML) is presented. CSS specifies how items should be shown in various media, including speech, paper, screens, and other media. One of the fundamental languages of the open web is CSS, which is standard across Web browsers in accordance with W3C specifications. In the past, different CSS standard components were developed simultaneously, allowing for the versioning of the most recent recommendations. You may be familiar with CSS 1, CSS 2, or perhaps CSS .

The primary sheet, JS, is where the app's functionality is detailed. The functioning of the buttons and other features that we add to the website is controlled by this sheet.

Java Script is a scripting or programming language that enables you to implement complex features on web pages. Whenever a web page displays dynamic content updates, interactive maps, animated 2D and 3D graphics, scrolling video jukeboxes, etc., you can bet that Java Script is likely involved. It is the third tier of the layer cake of common web technologies, of which HTML and CSS have already received extensive coverage.

Code

- **Html:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
    <link
     href="https://fonts.googleapis.com/css2?family=Merriweather+Sans:wght@300;400;500;60
     0;700&display=swap"
    rel="stylesheet" />
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <div class="wrapper">

        <h1>Weather App</h1>

        <div class="tab-container">
            <p class="tab" data-userWeather>Your Weather</p>
            <p class="tab" data-searchWeather>Search Weather</p>
        </div>

        <div class="weather-container">

            <!-- grant location container-->
            <div class="sub-container grant-location-container">
                <img src="./assets/location.png" width="80" height="80" loading="lazy">
                <p>Grant Location Access</p>
                <p>Allow Access to get weather Information</p>
```

```
        <button class="btn" data-grantAccess>Grant Access</button>
      </div>
      <!-- search form -> form-container-->
<form class="form-container" data-searchForm>
   <input placeholder="Search for City..." data-searchInput>
   <button class="btn" type="submit">
      <img src="./assets/search.png"  width="20" height="20" loading="lazy">
   </button>
</form>

<!--- loading screen container -->
<div class="sub-container loading-container">
   <img src="./assets/loading.gif" width="150" height="150">
   <p>Loading</p>
</div>

<!-- show weather info -->
<div class="sub-container user-info-container">

   <!--city name and Flag-->
   <div class="name">
      <p data-cityName></p>
      <img data-countryIcon>
   </div>

   <!-- weather descriptuion-->
   <p data-weatherDesc></p>
   <!--weather Icon-->
   <img data-weatherIcon>
   <!--temperature-->
   <p data-temp></p>

   <!--3 cards - parameters-->
   <div class="parameter-container">
   <!--card 1-->
      <div class="parameter">
         <img src="./assets/wind.png" >
         <p>windspeed</p>
         <p data-windspeed></p>
      </div>

      <!--card 2-->
```

```
            <div class="parameter">
                <img src="./assets/humidity.png" >
                <p>humidity</p>
                <p data-humidity></p>
            </div>

            <!--card 3-->
            <div class="parameter">
                <img src="./assets/cloud.png" >
                <p>Clouds</p>
                <p data-cloudiness></p>
            </div>
        </div>
    </div>
  </div>
</div>


    <script src="index.js"></script>
    </body>
    </html>
```

- **CSS:**

```
*,*::before,*::after {
    margin: 0;
   padding: 0;
   box-sizing: border-box;
   font-family: 'Merriweather Sans', sans-serif;
  }

:root {
        --colorDark1: #112D4E;
         --colorDark2: #3F72AF;
           --colorLight1: #DBE2EF;
            --colorLight2: #F9F7F7;
                                    }

 .wrapper{
 width:100vw;
 height:100vh;
```

```
        color: var(--colorLight2);
        background-image: linear-gradient(160deg, #112d4e 0%, #3f72af 100%);;
      }

        h1 {
       text-align: center;
        text-transform:uppercase;
        padding-top: 20px;
      }

      .tab-container {
       width:90%;
       max-width: 550px;
      margin: 0 auto;
      margin-top: 4rem;
      display: flex;
      justify-content: space-between;
    }

      .tab{
     cursor: pointer;
    font-size: 0.875rem;
    letter-spacing: 1.75px;
     padding: 5px 8px;
   }

   .tab.current-tab{
    background-color: rgba(219, 226, 239, 0.5);
    border-radius: 4px;
   }

   .weather-container{
 margin-block:4rem;
}

  .btn{
  all: unset;
  /* appearance: none;
  border:none;
  color: white; */
  font-size: 0.85rem;
  text-transform: uppercase;
```

```
    border-radius: 5px;
    background-color: var(--colorDark2);
   cursor: pointer;
   padding: 10px 30px;
   margin-bottom: 10px;
 }

    .sub-container{
    display:flex;
    flex-direction:column;
    align-items: center;
   }

   .grant-location-container{
   display:none;
  }

  .grant-location-container.active{
  display:flex;
}

  .grant-location-container img{
   margin-bottom: 2rem;
  }

  .grant-location-container p:first-of-type{
  font-size: 1.75rem;
  font-weight: 600;
}

 .grant-location-container p:last-of-type{
 font-size:0.85rem;
 font-weight: 500;
 margin-top: 0.75rem;
 margin-bottom: 1.75rem;
 letter-spacing: 0.75px;
}

 .loading-container{
  display: none;
 }
```

```
    .loading-container.active{
     display: flex;
   }
.form-container input{
   all:unset;
   width: calc(100% - 80px);
   height:40px;
   padding: 0 20px;
   background-color:rgba(219, 226, 239, 0.5);
   border-radius: 10px;
}

.form-container input::placeholder{
   color: rgba(255, 255, 255, 0.7);
}

.form-container input:focus{
   outline: 3px solid rgba(255, 255, 255, 0.7);
}

.form-container .btn {
   padding:unset;
   width: 40px;
   height: 40px;
   display: flex;
   align-items: center;
   justify-content: center;
   border-radius: 100%;
   margin-bottom:1px;
}
```

- **Javascript:**

```
const userTab = document.querySelector("[data-userWeather]");
const searchTab = document.querySelector("[data-searchWeather]");
const userContainer = document.querySelector(".weather-container");

const grantAccessContainer = document.querySelector(".grant-location-container");
const searchForm = document.querySelector("[data-searchForm]");
const loadingScreen = document.querySelector(".loading-container");
const userInfoContainer = document.querySelector(".user-info-container");
```

```
//initially vairables need????

let oldTab = userTab;
const API_KEY = "d1845658f92b31c64bd94f06f7188c9c";
oldTab.classList.add("current-tab");
getfromSessionStorage();

function switchTab(newTab) {
    if(newTab != oldTab) {
        oldTab.classList.remove("current-tab");
        oldTab = newTab;
        oldTab.classList.add("current-tab");

        if(!searchForm.classList.contains("active")) {
            //kya search form wala container is invisible, if yes then make it visible
            userInfoContainer.classList.remove("active");
            grantAccessContainer.classList.remove("active");
            searchForm.classList.add("active");
        }
        else {
            //main pehle search wale tab pr tha, ab your weather tab visible karna h
            searchForm.classList.remove("active");
            userInfoContainer.classList.remove("active");
            //ab main your weather tab me aagya hu, toh weather bhi display karna poadega, so let's
     check local storage first
                //for coordinates, if we haved saved them there.
            getfromSessionStorage();
        }
    }
}

userTab.addEventListener("click", () => {
    //pass clicked tab as input paramter
    switchTab(userTab);
});

searchTab.addEventListener("click", () => {
    //pass clicked tab as input paramter
    switchTab(searchTab);
});
```

```
//check if cordinates are already present in session storage
function getfromSessionStorage() {
    const localCoordinates = sessionStorage.getItem("user-coordinates");
    if(!localCoordinates) {
        //agar local coordinates nahi mile
        grantAccessContainer.classList.add("active");
    }
    else {
        const coordinates = JSON.parse(localCoordinates);
        fetchUserWeatherInfo(coordinates);
    }

}

async function fetchUserWeatherInfo(coordinates) {
    const {lat, lon} = coordinates;
    // make grantcontainer invisible
    grantAccessContainer.classList.remove("active");
    //make loader visible
const userTab = document.querySelector("[data-userWeather]");
const searchTab = document.querySelector("[data-searchWeather]");
const userContainer = document.querySelector(".weather-container");

const grantAccessContainer = document.querySelector(".grant-location-container");
const searchForm = document.querySelector("[data-searchForm]");
const loadingScreen = document.querySelector(".loading-container");
const userInfoContainer = document.querySelector(".user-info-container");

//initially vairables need????

let oldTab = userTab;
const API_KEY = "d1845658f92b31c64bd94f06f7188c9c";
oldTab.classList.add("current-tab");
getfromSessionStorage();

function switchTab(newTab) {
    if(newTab != oldTab) {
        oldTab.classList.remove("current-tab");
        oldTab = newTab;
        oldTab.classList.add("current-tab");

        if(!searchForm.classList.contains("active")) {
```

```
            //kya search form wala container is invisible, if yes then make it visible
            userInfoContainer.classList.remove("active");
            grantAccessContainer.classList.remove("active");
            searchForm.classList.add("active");
        }
        else {
            //main pehle search wale tab pr tha, ab your weather tab visible karna h
            searchForm.classList.remove("active");
            userInfoContainer.classList.remove("active");
            //ab main your weather tab me aagya hu, toh weather bhi display karna poadega, so let's
     check local storage first
            //for coordinates, if we haved saved them there.
            getfromSessionStorage();
        }
    }
}

userTab.addEventListener("click", () => {
    //pass clicked tab as input paramter
    switchTab(userTab);
});

searchTab.addEventListener("click", () => {
    //pass clicked tab as input paramter
    switchTab(searchTab);
});

//check if cordinates are already present in session storage
function getfromSessionStorage() {
    const localCoordinates = sessionStorage.getItem("user-coordinates");
    if(!localCoordinates) {
        //agar local coordinates nahi mile
        grantAccessContainer.classList.add("active");
    }
    else {
        const coordinates = JSON.parse(localCoordinates);
        fetchUserWeatherInfo(coordinates);
    }

}

async function fetchUserWeatherInfo(coordinates) {
```

```
    const {lat, lon} = coordinates;
    // make grantcontainer invisible
    grantAccessContainer.classList.remove("active");
    //make loader visible
    loadingScreen.classList.add("active");

    //API CALL
    try {
        const response = await fetch(

     `https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&appid=${API_K
     EY}&units=metric`
          );
        const  data = await response.json();

        loadingScreen.classList.remove("active");
        userInfoContainer.classList.add("active");
        renderWeatherInfo(data);
    }
    catch(err) {
        loadingScreen.classList.remove("active");
        //HW

    }

}

function renderWeatherInfo(weatherInfo) {
    //fistly, we have to fethc the elements

    const cityName = document.querySelector("[data-cityName]");
    const countryIcon = document.querySelector("[data-countryIcon]");
    const desc = document.querySelector("[data-weatherDesc]");
    const weatherIcon = document.querySelector("[data-weatherIcon]");
    const temp = document.querySelector("[data-temp]");
    const windspeed = document.querySelector("[data-windspeed]");
    const humidity = document.querySelector("[data-humidity]");
    const cloudiness = document.querySelector("[data-cloudiness]");

const userTab = document.querySelector("[data-userWeather]");
const searchTab = document.querySelector("[data-searchWeather]");
const userContainer = document.querySelector(".weather-container");
```

```
const grantAccessContainer = document.querySelector(".grant-location-container");
const searchForm = document.querySelector("[data-searchForm]");
const loadingScreen = document.querySelector(".loading-container");
const userInfoContainer = document.querySelector(".user-info-container");

//initially vairables need????

let oldTab = userTab;
const API_KEY = "d1845658f92b31c64bd94f06f7188c9c";
oldTab.classList.add("current-tab");
getfromSessionStorage();

function switchTab(newTab) {
    if(newTab != oldTab) {
        oldTab.classList.remove("current-tab");
        oldTab = newTab;
        oldTab.classList.add("current-tab");

        if(!searchForm.classList.contains("active")) {
            //kya search form wala container is invisible, if yes then make it visible
            userInfoContainer.classList.remove("active");
            grantAccessContainer.classList.remove("active");
            searchForm.classList.add("active");
        }
        else {
            //main pehle search wale tab pr tha, ab your weather tab visible karna h
            searchForm.classList.remove("active");
            userInfoContainer.classList.remove("active");
            //ab main your weather tab me aagya hu, toh weather bhi display karna poadega, so let's
     check local storage first
            //for coordinates, if we haved saved them there.
            getfromSessionStorage();
        }
    }
}

userTab.addEventListener("click", () => {
    //pass clicked tab as input paramter
    switchTab(userTab);
});
```

```
searchTab.addEventListener("click", () => {
    //pass clicked tab as input paramter
    switchTab(searchTab);
});

//check if cordinates are already present in session storage
function getfromSessionStorage() {
    const localCoordinates = sessionStorage.getItem("user-coordinates");
    if(!localCoordinates) {
        //agar local coordinates nahi mile
        grantAccessContainer.classList.add("active");
    }
    else {
        const coordinates = JSON.parse(localCoordinates);
        fetchUserWeatherInfo(coordinates);
    }

}

async function fetchUserWeatherInfo(coordinates) {
    const {lat, lon} = coordinates;
    // make grantcontainer invisible
    grantAccessContainer.classList.remove("active");
    //make loader visible
    loadingScreen.classList.add("active");

    //API CALL
    try {
        const response = await fetch(

     `https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&appid=${API_K
     EY}&units=metric`
          );
        const  data = await response.json();

        loadingScreen.classList.remove("active");
        userInfoContainer.classList.add("active");
        renderWeatherInfo(data);
    }
    catch(err) {
        loadingScreen.classList.remove("active");
        //HW
```

```
    }

}

function renderWeatherInfo(weatherInfo) {
    //fistly, we have to fethc the elements

    const cityName = document.querySelector("[data-cityName]");
    const countryIcon = document.querySelector("[data-countryIcon]");
    const desc = document.querySelector("[data-weatherDesc]");
    const weatherIcon = document.querySelector("[data-weatherIcon]");
    const temp = document.querySelector("[data-temp]");
    const windspeed = document.querySelector("[data-windspeed]");
    const humidity = document.querySelector("[data-humidity]");
    const cloudiness = document.querySelector("[data-cloudiness]");

    console.log(weatherInfo);

    //fetch values from weatherINfo object and put it UI elements
    cityName.innerText = weatherInfo?.name;
    countryIcon.src =
     `https://flagcdn.com/144x108/${weatherInfo?.sys?.country.toLowerCase()}.png`;
    desc.innerText = weatherInfo?.weather?.[0]?.description;
    weatherIcon.src =
     `http://openweathermap.org/img/w/${weatherInfo?.weather?.[0]?.icon}.png`;
    temp.innerText = `${weatherInfo?.main?.temp} °C`;
    windspeed.innerText = `${weatherInfo?.wind?.speed} m/s`;
    humidity.innerText = `${weatherInfo?.main?.humidity}%`;
    cloudiness.innerText = `${weatherInfo?.clouds?.all}%`;
}
function getLocation() {
    if(navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition);
    }
    else {
        //HW - show an alert for no gelolocation support available
    }
}
function showPosition(position) {
    const userCoordinates = {
        lat: position.coords.latitude,
```

```
    lon: position.coords.longitude,
    }
    sessionStorage.setItem("user-coordinates", JSON.stringify(userCoordinates));
    fetchUserWeatherInfo(userCoordinates);
}
const grantAccessButton = document.querySelector("[data-grantAccess]");
grantAccessButton.addEventListener("click", getLocation);
const searchInput = document.querySelector("[data-searchInput]");
searchForm.addEventListener("submit", (e) => {
    e.preventDefault();
    let cityName = searchInput.value;

    if(cityName === "")
        return;
    else
        fetchSearchWeatherInfo(cityName);
})

async function fetchSearchWeatherInfo(city) {
    loadingScreen.classList.add("active");
    userInfoContainer.classList.remove("active");
    grantAccessContainer.classList.remove("active");

    try {
        const response = await fetch(

     `https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${API_KEY}&units=
     metric`
          );
        const data = await response.json();
        loadingScreen.classList.remove("active");
        userInfoContainer.classList.add("active");
        renderWeatherInfo(data);
    }
    catch(err) {
//hW
    }
}
```

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In conclusion, creating a weather forecasting application using HTML, CSS, and JavaScript can be a rewarding project. By combining these three technologies, you can create a user-friendly and interactive application that provides real-time weather information to users.

HTML will be used to structure the application and define the co ntent. You can create different sections to display the weather details such as temperature, humidity, wind speed, and precipitation. Additionally, you can include images or icons to represent the weather conditions, such as sun, clouds, or rain.

CSS will allow you to style the application and make it visually appealing. You can define the layout, fonts, colors, and animations to enhance the overall user experience. With CSS, you can create responsive designs that adapt to different screen sizes and ensure your application looks good on both desktop and mobile devices.

JavaScript is the programming language that will bring interactivity and functionality to your weather application. You can use JavaScript to fetch weather data from an API (Application Programming Interface) such as OpenWeatherMap or Weatherbit. With the retrieved data, you can dynamically update the HTML elements to display the current weather conditions and forecast.

To enhance the user experience, you can implement additional features such as location detection to automatically display the weather for t he user's current location, a search functionality to allow users to search for weather in specific cities, and the ability to switch between different units of measurement (e.g., Celsius and Fahrenheit).

Overall, building a weather forecasting applicatio n using HTML, CSS, and JavaScript allows you to combine the power of these technologies to create an interactive and visually appealing application that provides real -time weather information to users. It can be an excellent opportunity to apply and enhanc e your web development skills.

5.2 Future Scope

Building a weather forecasting application using HTML, CSS, and JavaScript has a promising future scope. Weather forecasting is a vital aspect of daily life for many people, and as technology advances, there is an increasing demand for more accurate and accessible weather information. Here are some aspects to consider regarding the future scope of such an application:

- **User Engagement:** Weather applications with intuitive and user-friendly interfaces tend to attract a large user base. By leveraging HTML, CSS, and JavaScript, you can create visually appealing and interactive designs that enhance user engagement. Consider incorporating features such as real-time weather updates, customizable settings, and user preferences to improve the user experience.

- **Mobile compatibility:** With the rising popularity of smartphones and tablets, mobile compatibility is crucial. Designing your weather application to be responsive and mobile-friendly ensures that users can access weather information on the go. Employ responsive design techniques using HTML, CSS media queries, and JavaScript libraries or frameworks to create a seamless experience across different devices**.**

- **Integration With APIs:** Weather forecasting applications typically rely on data from external sources, such as weather APIs. JavaScript enables you to fetch and process data from these APIs, allowing you to display accurate and up-to-date weather information to users. As the availability and accuracy of weather APIs improve, your application can benefit from more precise forecasts and additional data points.

- Data Visualization: Weather data visualization is an essential aspect of forecasting applications. HTML, CSS, and JavaScript can be used to create interactive charts, graphs, and maps to represent weather patterns, temperature variations, rainfall predictions, and other meteorological data. Implementing data visualization techniques effectively can enhance the user's understanding of weather forecasts.

- **Geolocation and Personalization:** Integrating geolocation functionality into your application can automatically provide localized weather information to users based on their location. By personalizing weather forecasts and notifications, users can receive rel evant and timely updates specific to their preferences, such as severe weather alerts or specific temperature ranges.

- **Machine Learning and AI:** The future of weather forecasting involves incorporating machine learning and AI techniques to improve accuracy and predictability. You can use JavaScript libraries or frameworks to implement machine learning algorithms for weather prediction or to analyze historical weather data. By continuously updating and refining your forecasting models, you can offer more pre cise and reliable predictions to users.

- **Integration with Smart Devices:** As the Internet of Things (IoT) expands, there is an increasing integration of weather data with smart devices. Your weather application can leverage HTML, CSS, and JavaScript to interact with IoT devices such as smart thermostats, sprinkler systems, or connected cars, allowing users to automate actions based on weather conditions.

- **Social and Community Features:** Weather applications often have social and community features that allow users to share weather updates, photos, or personal observations. By incorporating social media integration, users can engage with others who are experiencing similar weather conditions or share information with their networks, fostering a sense of community and enhancing the application's usefulness.

Overall, the future scope of a weather forecasting application built using HTML, CSS, and JavaScript is promising. By leveraging the advancements in web technologies and incorporating emerging trends, you can create an engaging, accurate, and personalized weather application that meets the evolving needs of users.

References:

[2] Vesna Rankovic´, Aleksandar Novakovic´ ,Nenad Grujovic, Dejan Divac,Nikola Milivojevic´,"Predicting piezometric water level in dams via artificial neural networks",Springer-Verlag London 2013,Neural Computing & Applications DOI 10.1007/s00521-012- 1334-2 DOI 10.1007/s00521-012-1334-.

[3] Deepak Ranjan Nayak , Amitav Mahapatra, Pranati Mishra, "A Survey on Rainfall Prediction using Artificial Neural Network", International Journal of Computer Applications (0975 – 8887),Volume 72, No.16, ,pp 32 - 40,June 2013.

[4] Amanpreet Kaur, Harpreet Singh, "Artificial Neural Networks in Forecasting Minimum Temperature", International Journal of Electronics and Communication Technology, Vol. 2, Issue 3, pp 101-105, Sept. 2011

[5] Enireddy.Vamsidhar, K.V.S.R.P.Varma P.Sankara Rao, Ravikanth satapati, "Predictionof Rainfall Using Back propagation Neural Network Model", International Journal on Computer Science and Engineering, Volume 02, No. 04,pp 1119-1121,2010.

[6] Weerasinghe H.D.P.,. Premaratne H.L and Sonnadara D.U.J., 2010, "Performance of neural networks in forecasting daily precipitation using multiple sources", J.Natn.Sci.Foundation Sri Lanka, vol. 38(3), pp. 163-170.

[7] Karmakar Sanjeev et al., 2008, "Development of Artificial Neural Network Models for Long-Range Meteorological Parameters Pattern Recognition over the Smaller Scale Geographical Region", IEEE Computer Society, Washington, DC, USA., 8-10 Dec. 2008, pp.1 – 6.

[8] Mohsen Hayati and Zahra Mohebi, "Temperature Forecasting Based on Neural Network Approach", World Applied Sciences Journal ,Volume 2(6), pp 613- 620,2007

[9] Toth E. *, Brath A., Montanari A., 2000, "Comparison of short-term rainfall prediction

[10]models for real-time flood forecasting", Journal of Hydrology, Elsevier, 239, pp. 132- 147.

[10]Basu Sujit & Andharia H I, 1992, "The Chaotic time series of Indian Monsoon rainfall and its prediction", Proc. Indian Acad. Sci., Vol. 101, No. 1, pp 27-34.