**CHAPTER 1**

**INTRODUCTION OF THE PROJECT**

* 1. **Introduction**

We will talk about a project called “Weather Web (A MERN Project)”. This project required the usage of several technologies and tools. This is simple, safe, and secure solution that requires little time. Creating a website that will serve as a solution to this problem. Weather forecasting is the prediction of the state of the atmosphere for a given location using the application of science and technology. This includes temperature, rain, cloudiness, wind speed, and humidity. Weather warnings are a special kind of short-range forecast carried out for the protection of human life.

So we are creating this project in **MERN (MongoDB, ExpressJs, ReactJs, NodeJs)** technology. Here this system will predict weather based on parameters such as temperature, humidity and wind. User will enter current temperature humidity and wind, System will take this parameter and will predict weather (rainfall in inches) from previous data in database. Because observations are so critical to weather prediction, an account of meteorological measurements and weather forecasting is a story in which ideas and [technology](https://www.britannica.com/technology/technology) are closely intertwined, with creative thinkers drawing new insights from available observations and pointing to the need for new or better measurements, and technology providing the means for making new observations and for processing the data derived from measurements. The basis for weather prediction started with the theories of the ancient Greek philosophers and continued with [Renaissance](https://www.britannica.com/event/Renaissance) scientists, the scientific revolution of the 17th and 18th centuries, and the theoretical models of 20th- and 21st-century atmospheric scientists and meteorologists. Likewise, it tells of the development of the “synoptic” idea—that of characterizing the weather over a large region at exactly the same time in order to organize information about prevailing conditions.

* 1. **History**

Greek philosophers had much to say about [meteorology](https://www.britannica.com/science/meteorology), and many who subsequently engaged in [weather](https://www.britannica.com/science/weather) forecasting no doubt made use of their ideas. Unfortunately, they probably made many bad forecasts, because [Aristotle](https://www.britannica.com/biography/Aristotle), who was the most influential, did not believe that [wind](https://www.britannica.com/science/wind) is air in motion. He did believe, however, that west winds are cold because they blow from the sunset. [Torricelli, Evangelista](https://cdn.britannica.com/21/195421-050-FC98CA10/Evangelista-Torricelli-physicist-Italian-mercury-barometer.jpg) The scientific study of meteorology did not develop until measuring instruments became available. Its beginning is commonly associated with the invention of the [mercury barometer](https://www.britannica.com/technology/mercury-barometer) by [Evangelista Torricelli](https://www.britannica.com/biography/Evangelista-Torricelli), an Italian physicist-mathematician, in the mid-17th century and the nearly [concurrent](https://www.merriam-webster.com/dictionary/concurrent) development of a reliable [thermometer](https://www.britannica.com/technology/thermometer). (Galileo had constructed an elementary form of gas thermometer in 1607, but it was defective; the efforts of many others finally resulted in a reasonably accurate liquid-in-glass device.)A succession of notable achievements by chemists and physicists of the 17th and 18th centuries contributed significantly to meteorological research. The formulation of the laws of gas [pressure](https://www.britannica.com/science/pressure), [temperature](https://www.britannica.com/science/temperature), and [density](https://www.britannica.com/science/density) by [Robert Boyle](https://www.britannica.com/biography/Robert-Boyle) and [Jacques-Alexandre-César Charles](https://www.britannica.com/biography/Jacques-Charles), the development of calculus by [Isaac Newton](https://www.britannica.com/biography/Isaac-Newton) and [Gottfried Wilhelm Leibniz](https://www.britannica.com/biography/Gottfried-Wilhelm-Leibniz), the development of the law of partial pressures of mixed gases by [John Dalton](https://www.britannica.com/biography/John-Dalton), and the formulation of the [doctrine](https://www.britannica.com/dictionary/doctrine) of [latent heat](https://www.britannica.com/science/latent-heat) (i.e., heat release by [condensation](https://www.britannica.com/science/condensation-phase-change) or freezing) by [Joseph Black](https://www.britannica.com/biography/Joseph-Black) are just a few of the major scientific breakthroughs of the period that made it possible to measure and better understand theretofore unknown aspects of the atmosphere and its behavior. During the 19th century, all of these brilliant ideas began to produce results in terms of useful [weather](https://www.britannica.com/science/weather-map) forecasts.

* **The emergence of synoptic forecasting methods:** An observant person who has learned nature’s signs can interpret the appearance of the sky, the wind, and other local effects and “foretell the weather.” A scientist can use instruments at one location to do so even more effectively. The modern approach to weather forecasting, however, can only be realized when many such observations are exchanged quickly by experts at various weather stations and entered on a synoptic weather map to depict the patterns of pressure, wind, temperature, clouds, and [precipitation](https://www.britannica.com/science/precipitation) at a specific time. Such a rapid exchange of weather data became [feasible](https://www.merriam-webster.com/dictionary/feasible) with the development of the electric [telegraph](https://www.britannica.com/technology/telegraph) in 1837 by [Samuel F.B. Morse](https://www.britannica.com/biography/Samuel-F-B-Morse) of the [United States](https://www.britannica.com/place/United-States). By 1849 [Joseph Henry](https://www.britannica.com/biography/Joseph-Henry) of the [Smithsonian Institution](https://www.britannica.com/topic/Smithsonian-Institution) in Washington, D.C., was plotting daily weather maps based on telegraphic reports, and in 1869 [Cleveland Abbe](https://www.britannica.com/biography/Cleveland-Abbe) at the Cincinnati Observatory began to provide regular weather forecasts using data received telegraphically.
* **Establishment of weather-station networks and services:** Routine production of synoptic weather maps became possible after networks of stations were organized to take measurements and report them to some type of central observatory. As early as 1814, U.S. Army Medical Corps personnel were ordered to record [weather](https://www.britannica.com/science/weather-bureau) data at their posts; this activity was subsequently expanded and made more systematic. Actual weather-station networks were established in the United States by [New York University](https://www.britannica.com/topic/New-York-University), the [Franklin Institute](https://www.britannica.com/topic/Franklin-Institute), and the Smithsonian Institution during the early decades of the 19th century. In Britain, [James Glaisher](https://www.britannica.com/biography/James-Glaisher) organized a similar network, as did [Christophorus H.D. Buys Ballot](https://www.britannica.com/biography/Christophorus-Buys-Ballot) in the Netherlands. Other such networks of weather stations were developed near Vienna, Paris, and [St. Petersburg](https://www.britannica.com/place/St-Petersburg-Russia). It was not long before national meteorological services were established on the Continent and in the United Kingdom. The first national weather service in the United States [commenced](https://www.britannica.com/dictionary/commenced) operations in 1871, with responsibility assigned to the [U.S. Army Signal Corps](https://www.britannica.com/topic/United-States-Army-Signal-Corps). The original purpose of the service was to provide storm warnings for the Atlantic and Gulf coasts and for the [Great Lakes](https://www.britannica.com/place/Great-Lakes). Within the next few decades, national meteorological services were established in such countries as Japan, India, and Brazil. The importance of international cooperation in weather prognostication was recognized by the directors of such national services. By 1880 they had formed the [International Meteorological Organization](https://www.britannica.com/topic/International-Meteorological-Organization) (IMO).
* **Progress during the early 20th century:** An important aspect of [weather](https://www.britannica.com/science/weather) prediction is to calculate the [atmospheric pressure](https://www.britannica.com/science/atmospheric-pressure) pattern—the positions of the highs and lows and their changes. Modern research has shown that sea-level [pressure](https://www.britannica.com/science/pressure) patterns respond to the motions of the upper-atmospheric winds, with their narrow, fast-moving jet streams and waves that [propagate](https://www.merriam-webster.com/dictionary/propagate) through the air and pass air through themselves. Frequent surprises and errors in estimating surface atmospheric pressure patterns undoubtedly caused 19th-century forecasters to seek information about the upper atmosphere for possible explanations. The British meteorologist [Glaisher](https://www.britannica.com/biography/James-Glaisher) made a series of ascents by [balloon](https://www.britannica.com/technology/balloon) during the 1860s, reaching an unprecedented height of nine kilometres. At about this time investigators on the Continent began using unmanned balloons to carry recording barographs, thermographs, and hygrographs to high altitudes. During the late 1890s meteorologists in both the [United States](https://www.britannica.com/place/United-States) and Europe used kites equipped with instruments to [probe](https://www.britannica.com/dictionary/probe) the atmosphere up to altitudes of about three kilometres. Notwithstanding these efforts, knowledge about the upper atmosphere remained very limited at the turn of the century. The situation was aggravated by the confusion created by observations from weather stations located on mountains or hilltops. Such observations often did not show what was expected, partly because so little was known about the upper atmosphere and partly because the mountains themselves affect measurements, producing results that are not representative of what would be found in the free atmosphere at the same altitude.
* **Modern trends and developments:** Once again technology provided the means with which to test the new scientific ideas and stimulate yet newer ones. During the late 1920s and ’30s, several groups of investigators (those headed by [Yrjö Väisälä](https://www.britannica.com/biography/Yrjo-Vaisala) of Finland and Pavel Aleksandrovich Malchanov of the [Soviet Union](https://www.britannica.com/place/Soviet-Union), for example) began using small radio transmitters with balloon-borne instruments, eliminating the need to recover the instruments and speeding up access to the upper-air data. These [radiosondes](https://www.britannica.com/technology/radiosonde), as they came to be called, gave rise to the upper-air observation networks that still exist today. Approximately 75 stations in the United States and more than 500 worldwide release, twice daily, balloons that reach heights of 30,000 metres or more. Observations of [temperature](https://www.britannica.com/science/temperature) and [relative humidity](https://www.britannica.com/science/relative-humidity) at various pressures are radioed back to the station from which the balloons are released as they ascend at a predetermined rate. The balloons also are tracked by radar and global positioning system (GPS) satellites to [ascertain](https://www.merriam-webster.com/dictionary/ascertain) the behaviour of winds from their drift. Forecasters are able to produce synoptic weather maps of the upper atmosphere twice each day on the basis of [radiosonde](https://www.britannica.com/technology/radiosonde) observations. While new methods of upper-air measurement have been developed, the primary synoptic clock times for producing upper-air maps are still the radiosonde-observation times—namely, 0000 (midnight) and 1200 (noon) [Greenwich Mean Time](https://www.britannica.com/science/Greenwich-Mean-Time) (GMT). Furthermore, modern computer-based forecasts use 0000 and 1200 GMT as the starting times from which they calculate the changes that are at the heart of modern forecasts. It is, in effect, the synoptic approach carried out in a different way, intimately linked to the radiosonde networks developed during the 1930s and ’40s.
  1. **Objective of the Project:**

MERN Weather Web weathercast involves predicting things like **cloud cover, rain or snow, wind speed, and temperature before they happen**. ... We forecast the weather by looking at current conditions, motion of air and clouds, historical patterns, pressure changes, and computer models.

* 1. **Problem Statement:** The determination of the state often atmosphere at the initial time is the task of observational meteorology This problem has not yet been solved to the extent that is necessary for rational forecasting. There are two major gaps in the observer. The first one is that only land stations participate in the daily programs of the weather services. Over the seas, which cover four-fifths of the earth's surface and must therefore exert an overwhelming influence, no observations are made for the purposes of current weather analysis. Furthennore, the observations that are used in current analysis are made only at the surface of the earth and all data pertaining to the state of the higher layers of the atmosphere are missing. But we already have the technical means that will enable us to fill these two gaps. With the help of wireless telegraphy, we will be able to include among the reporting stations the ships moving in fixed routes. And to judge by the great forward steps that have been made in recent years in the techniques of upper air soundings, it will be possible to obtain.
* A sufficiently accurate knowledge of the state of the atmosphere at the initial time.
* A sufficiently accurate knowledge of the laws according to which one state of the atmosphere develops from another.
  1. **Justification and Needs for the Weather Web:**

The proposed weather forecasting data model utilizes the kmeans unsupervised learning technique for performing the clustering on the entire training dataset. This clustering is performed for finding the pattern level pattern similarity among two instance data. The total number of 12 clusters is developed; these clusters are providing the observations of the data and their weather conditions. Using the extracted observations and available class labels the data is reorganized in terms of observation matrix and the transition matrix. These two different matrixes are provided into next phase namely Hidden Markov Model and the training are performed. The trained data model is used for prediction or the pattern recognition work. For making the prediction, using prepared trained data model. The system needs to provide the current weather conditions on the basis of that the new upcoming pattern of data is predicted.

* 3.6 HMM training The hidden Markov model is responsible to accepting these two matrixes as input and producing the learned model for prediction. The Hidden Markov model is trained in this phase for the given observational patterns and the transitional patterns.
* Current weather In order to predict the next weather condition or upcoming weather condition the system required to take input the just patterns of the weather conditions, based on the observation and transitional patterns the system generate the next possible pattern of weather condition.
* Predicted weather That is the final outcome of the proposed data model as the predictive outcome.
* Proposed algorithm The given section summarizes the entire process models into the procedural steps. Thus the section introduces the proposed algorithm for prediction.
  1. **Function of the Weather Web:**
* Weather Training Data The supervised algorithms are working on labelled data. Such kind of data most of the time found in structured format. This structured data has some pre-defined class labels that are representing the outcomes of the combination of attributes. Thus using the training data the algorithm learned on pre-defined patterns. In this presented work the Bhopal weather forecasting data for last five years are used. That training dataset contains the different weather attributes observations and the class labels as the weather conditions.
* **Pre-processing**: The pre-processing is a technique by which the data is refined, transformed and cleaned for improving the quality of the training data on which the data model is prepared for decision making or prediction. In this given technique the three major contributions is placed in this phase. Removal of attributes that are not fluctuating with the other patterns In this phase those attributes are removed from the training samples which are not fluctuating with the different data patterns. Thus this technique reduces the dimension of the dataset by which the memory consumption of the data analysis is reduces.
* **Removal of attributes:** that uniquely defined the instance data In this phase the data set is evaluated for removal of data that are performing the identity representation for the dataset objects.
* **Missing data handling**: In this phase the data is analyzed for finding the missing attributes in the data set. That also improves the quality of data for representation of accurate data model. Therefore those dataset objects are removed which are not completed or having the missing attributes.
* **K-means clustering**: In this phase the well refined and defined data is used for prepare the groups of the data which are simulating the similar behavioural patterns. Therefore the entire data is clustered according to the similar attributes in these grouping of data the data attributes are considered. The outcome of the k-means clustering is organized in two main matrixes given as.
  1. **Advantages of the Weather Web:**
* Weather forecasting is the prediction of the state of the atmosphere in a particular region over a period of 24 to 48 hours
* Weather forecasting is done almost daily by various authorities given the power to do so by the government, for example in Tanzania weather forecasting is performed by Tanzania meteorological authority.
* Military personnel benefit from **weather forecasting** as they can plan their military activities based on expected weather conditions. During the war the military can plan their battles by featuring in the expected**weather** condition to maximize the chance of winning the war.
* Weather forecasting enable people to plan and take precautions against various natural calamities such as[**flood**](https://geographypoint.com/2021/09/effects-of-floods-and-how-to-mitigate-them/) and typhoon so that to minimize their effects; bad weather such as torrent [**rainfall**](https://geographypoint.com/2015/07/three-3-types-of-rainfall/) or strong wind can destroy properties and lead to death therefore if there is prediction that bad weather is going to happen people can take precaution such as evacuating from affected areas or staying indoors.
* Weather forecasting enables farmers to adjust their [**farming**](https://geographypoint.com/2021/10/problems-facing-horticultural-farming-in-kenya/) activities to suit the expected weather condition. For example, if it is expected that there will be less rainfall in the future then farmers will prepare an [**irrigation**](https://geographypoint.com/2020/11/reasons-why-irrigation-is-carried-out-in-east-africa/)**system** to compensate for the shortfall.
* help surfers to know when huge waves are expected
* Weather forecasting greatly influences [**transport**](https://geographypoint.com/2021/01/effects-of-transport-on-the-environment/), especially in air and water. Aircraft take-off and landing can be affected by weather whereas storms and strong winds greatly affect water travel.
  1. **Disadvantages of Weather Web:**
* [weather](https://geographypoint.com/2022/12/10-factors-influencing-weather/) is extremely difficult to forecast correctly
* it is expensive to monitor so many variables from 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
* Confusing Terminology: The terminology used in weather forecasting can be confusing, making it difficult for some people to understand the predictions.
* Reliance on Technology: Weather forecasting relies heavily on technology, and if the technology fails or is unavailable, accurate predictions cannot be made.
* Limited Reach: Weather forecasts are not available for many remote or sparsely populated areas, making it difficult for people in these areas to prepare for severe weather.
* Model Limitations: Forecasting models can only make predictions based on existing data and are limited by the quality and quantity of that data.
* Limited Time Frame: Forecasts are usually only accurate for a short time frame, making it difficult to plan ahead.
  1. **Literature Review:**

Weather forecasting, as an important and indispensable procedure in people’s daily lives, evaluates the alteration happening in the current condition of the atmosphere. Big data analytics is the process of analyzing big data to extract the concealed patterns and applicable information that can yield better results. Nowadays, several parts of society are interested in big data, and the meteorological institute is not excluded. Therefore, big data analytics will give better results in weather forecasting and will help forecasters to forecast weather more accurately. In order to achieve this goal and to recommend favorable solutions, several big data techniques and technologies have been suggested to manage and analyze the huge volume of weather data from different resources. By employing big data analytics in weather forecasting, the challenges related to traditional data management techniques and technology can be solved. This paper tenders a systematic literature review method for big data analytic approaches in weather forecasting (published between 2014 and August 2020). A feasible taxonomy of the current reviewed papers is proposed as technique-based, technology-based, and hybrid approaches. Moreover, this paper presents a comparison of the aforementioned categories regarding accuracy, scalability, execution time, and other Quality of Service factors. The types of algorithms, measurement environments, modeling tools, and the advantages and disadvantages per paper are extracted. In addition, open issues and future trends are debated.

* 1. **Methodology:**

**1.10.1 Current System:**

Weather Training Data The supervised algorithms are working on labelled data. Such kind of data most of the time found in structured format. This structured data has some pre-defined class labels that are representing the outcomes of the combination of attributes. Thus using the training data the algorithm learned on pre-defined patterns. In this presented work the Bhopal weather forecasting data for last five years are used. That training dataset contains the different weather attributes observations and the class labels as the weather conditions.

* **Pre-processing**: The pre-processing is a technique by which the data is refined, transformed and cleaned for improving the quality of the training data on which the data model is prepared for decision making or prediction. In this given technique the three major contributions is placed in this phase. Removal of attributes that are not fluctuating with the other patterns In this phase those attributes are removed from the training samples which are not fluctuating with the different data patterns. Thus this technique reduces the dimension of the dataset by which the memory consumption of the data analysis is reduces.

Removal of attributes that uniquely defined the instance data In this phase the data set is evaluated for removal of data that are performing the identity representation for the dataset objects.

* **Missing data handling**: In this phase the data is analyzed for finding the missing attributes in the data set. That also improves the quality of data for representation of accurate data model. Therefore those dataset objects are removed which are not completed or having the missing attributes.
* **K-means clustering:** In this phase the well refined and defined data is used for prepare the groups of the data which are simulating the similar behavioural patterns. Therefore the entire data is clustered according to the similar attributes in these grouping of data the data attributes are considered. The outcome of the k-means clustering is organized in two main matrixes given as.

**1.10.2 Problem with Current System:**

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.

* 1. **Proposed System:**
* **Memory used:** The memory consumption shows the amount of main memory required to process the algorithm task. That is also called the space complexity of the system. That is also known as the space complexity in algorithm study. The figure 6 and table 6 shows the memory consumption of the system with increasing size of training dataset. The amount of data is given using the X axis and the Y axis shows the amount of consumed memory during experimentation with respective amount of data in terms of kilobytes. According to the experimented results the amount of memory is similar and not more fluctuating. But the respective comparison the proposed algorithm is efficient than the traditional approach of prediction. In order to investigate the difference in the memory consumption of the proposed and traditional technique the mean memory consumption is calculated and reported using the figure 7. In this diagram the X axis contains the methods name and the Y axis contains the mean memory consumption of the system.
* **Accuracy:** The accuracy of the predictive algorithm provides the amount of generated prediction is similar to actual outcomes. Therefore that can also be defines as the amount of correctly recognized patterns among the total samples produces to test.
* **Prediction Time:** The amount of time required to evaluate the data form making accurate prediction is termed here as the prediction time. That prediction time of both the system in comparative manner is demonstrated using figure 4 and the table 5. In this diagram the amount of data is given in the X axis and the Y axis shows the performance obtained in terms of milliseconds. According to observations the amount of time during prediction is not much fluctuated and not also affected by the amount of data to be process. The comparative results of the systems shows the effectiveness of the proposed technique that consumes less time for computing the predicted events as compared to traditional approach. Because during the time based data clustering reduces the amount of data to process. Thus the prediction is much frequent as compared to the traditional approach.
* **Proposed algorithm:** The given section summarizes the entire process models into the procedural steps. Thus the section introduces the proposed algorithm for prediction.

**CHAPTER 2**

**FEASIBILITY STUDY**

A test or review of the entire system plan is called system feasibility. Such an evaluation is required to describe the application area, as well as its length and complexity, to specify the extent of computerization, as well as proposed output and input formats, and to determine prospective advantages. The system analysis must determine if a technically feasible solution is practicable. The survey is enlarged to a more extensive research based on the results of the original examination. A feasibility study is the best system concept based on its influence on the organization, capacity to satisfy demands, and effective use of resources. The result of the feasibility study is a formal proposal. This is simply a report, a formal document detailing the nature and scope of the proposed system. After management reviews the proposal, it becomes a formal agreement that proves the way for actual design and implementation. The feasibility study is an investigation that results in a written document that: Defines the scope of the problem.

• Identifies the elements of the problem.

• Identifies the evaluative criteria.

• Possible criteria includes: The impact on the environment, safety and

• manufacture ability, the political climate; the possible difficulties in the design phase; and appraisal of the return (profit) on investment. Identifies possible alternative solutions.

• Evaluate each solution with the criteria.

The goal of the feasibility study is to discover possible solutions and to determine which of these appear to have promise and which are not feasible and why. In a feasibility study we need to concentrate on four primary areas of interest

# Technical Feasibility:

* Technical feasibility refers to the assessment of whether the proposed systemcan be implemented using the existing technology and infrastructure, or whether new technologies and infrastructure need to be developed. This evaluation involves an examination of the hardware, software, and other technical resources needed to build, operate, and maintain the proposed system. Hardware considerations include the assessment of the processing power.

• memory, storage capacity, and network connectivity required to support the system's operations. Software considerations involve identifying the necessary software components, programming languages, and development tools needed to build and operate the system. Technical feasibility also involves assessing the compatibility of the proposed system with existing software and hardware platforms.

* 1. **OPERATIONAL FEASIBILITY**:
* Operational feasibility is a measure of whether a proposed project or system can be effectively and efficiently implemented and operated within an organization's existing infrastructure, resources, and culture. It is one of the key factors that determine the success of a project, and is therefore an important consideration during the planning phase of any project.
* In the context of transitioning from a manual system to a computerized system, operational feasibility refers to the ability of the organization to manage and maintain the new system, and the availability of the necessary resources and expertise to do so. This includes the availability of high quality manpower, the platform and tools used to develop and operate the system, and the ease of use and understanding of the system by the users.
* When evaluating the operational feasibility of a project, it is important to consider factors such as the cost of implementing and maintaining the system, the availability of technical support, the level of training and expertise required for system operation, and the compatibility of the new system with existing business processes and systems.
  1. **BEHAVIOURAL FEASIBILITY:**
* Behavioral feasibility refers to the likelihood of users accepting and using a system as intended. It is an important aspect of feasibility analysis, which is typically conducted before a project is initiated, to determine whether the proposed system is practical, functional, and cost-effective.
* The aim of behavioral feasibility is to understand how users will interact with the system, and whether they will find it easy to use and accept it as a necessary tool. This involves training the users to get used to the system and ensuring that it meets their needs and expectations.
* The level of acceptance by the user is a critical factor in determining the success of the system. If users do not find the system useful or user-friendly, they are unlikely to adopt it, resulting in low usage and poor return on investment. Therefore, it is essential to design and develop systems that are easy to use, intuitive, and meet the needs of the users.
  1. **LEGAL FEASIBILITY:**
* Legal feasibility refers to the analysis of the legal aspects of a proposed project, which includes identifying and evaluating any legal barriers that may impact the implementation of the project. This feasibility study examines the legal requirements that need to be satisfied for a project to proceed legally and ethically.
* The legal feasibility study typically begins by identifying the relevant laws, regulations, and policies that govern the proposed project. This may involve a review of local, state, and federal laws, as well as industry-specific regulations, data protection acts, social media laws, intellectual property laws, and other legal requirements that may impact the project.
* During the legal feasibility study, the project team will assess whether the• proposed project complies with all relevant legal requirements. This may involve reviewing the project plan, scope, budget, and schedule, as well as any contracts, licenses, permits, or certifications required to complete the project. 15 The team may also evaluate any potential legal risks associated with the project and develop strategies to mitigate those risks.

Some of the key factors that are typically evaluated during a legal feasibility study include:

* **Legal compliance:** Whether the proposed project complies with all relevant laws, regulations, and policies.
* **Intellectual property:** Whether the project infringes on any existing patents, trademarks, or copyrights, or whether the project itself creates any new intellectual property rights.
* **Liability:** Whether the project could potentially expose the organization or individuals to legal liability.
* **Data protection and privacy:** Whether the project complies with data protection and privacy laws and regulations, including the storage, processing, and transmission of personal data.
* **Social media:** Whether the project complies with the applicable social media laws and regulations, including advertising, disclosure, and user privacy.
  1. **ECONOMICAL FEASIBILITY:**
* Economic feasibility refers to the ability of a proposed project or system to generate economic benefits that exceed its costs, making it a viable investment option. It involves evaluating the economic viability of a project or system by analyzing the benefits it is expected to generate against the cost of implementing and maintaining it.
* Economic feasibility is a critical aspect of any business decision-making process. The objective is to identify the financial risks and opportunities associated with a proposed project or system, and determine whether the expected benefits justify the required investment.
* The economic feasibility analysis involves several steps, including identifying the project's objectives, estimating the costs associated with implementing and maintaining the system, and estimating the expected benefits over the life of 16 the system. The analysis also considers factors such as the project's payback period, return on investment, and net present value.
* The cost/benefit analysis is a common method used to determine economic feasibility. It involves comparing the expected benefits of the system, such as increased revenue or reduced costs, against the costs of implementing and maintaining the system, including development, operating, and maintenance costs. If the benefits exceed the costs, the project is deemed economically feasible and can be considered for implementation.
  + 1. **Cost based study:**
* A cost-based study is an analysis of the expenses associated with a particular system or project. The purpose of this study is to identify and estimate the costs that will be incurred in the development, implementation, and maintenance of the system, and to determine the benefits that can be derived from the system.
* The cost-based study involves gathering data on the various costs involved in the project, including direct costs such as salaries, equipment, and materials, as well as indirect costs such as overhead, administrative expenses, and training. These costs are then analyzed to determine the total cost of the project and to identify areas where cost savings can be achieved.
  + 1. **Time based study:**
* Time-based study is a type of analysis that focuses on determining the time required to achieve a return on investment (ROI). This type of analysis is useful in evaluating the viability of a project and determining its profitability. In a time-based study, several factors are considered, including the benefits derived from the system the future value of the project, and the costs involved in developing and maintaining the application.
* One important factor that is considered in a time-based study is the benefits derived from the system. These benefits can be both tangible and 17 intangible. Tangible benefits are those that can be quantified in terms of money, such as increased revenue or reduced costs. Intangible benefits are those that cannot be quantified in terms of money, such as improved customer satisfaction or enhanced brand image. In a time-based study, both tangible and intangible benefits are considered to determine the overall value of the project.
* Another important factor that is considered in a time-based study is the future value of the project. This refers to the potential profitability of the project over time. The future value of a project is determined by estimating its revenue and costs over a certain period of time and then calculating the net present value (NPV) of those cash flows. A positive NPV indicates that the project is profitable, while a negative NPV indicates that the project is not profitable.
* The costs involved in developing and maintaining the application are also considered in a time-based study. These costs can include development costs, operating costs, and maintenance costs. Development costs refer to the expenses incurred in designing and building the application. Operating costs refer to the expenses incurred in running the application, such as server costs and employee salaries. Maintenance costs refer to the expenses incurred in maintaining the application over time, such as bug fixes and updates.
  1. **SCHEDULE FEASIBILITY:**
* Schedule feasibility refers to the ability of a project to be completed within its allotted time frame. It is an important aspect of project management because it helps determine whether a project is realistic and achievable within the given constraints of time and resources.
* In order to determine the schedule feasibility of a project, a project manager must consider a variety of factors, including the complexity of the project, the availability of resources, the level of technical expertise required, and the nature of the project's goals and objectives.

**CHAPTER 3**

**SOFTWARE REQUIREMENTS & SYSTEM**

**SPACIFICATION**

# 3.1 Software Requirement Specification:

A Software Requirements Specification (SRS) is a complete description of the behaviour of the system to be developed. It fully describes what the software will do and how it will be expected to perform. It includes a set of use cases that describe all the interactions the users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains nonfunctional (or supplementary) requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints). An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations. Parameters such as operating speed, response time, availability, portability, maintainability, footprint, security and speed of recovery from adverse events are evaluated.

**3.1.1 Developer responsibilities overview:**

The roles and responsibilities that people can assume in the project are based on merit. Everybody can help no matter what their role. Those who have been long term or valuable contributors to the project obtain the right to vote and commit directly to the source repository.

* **USER**

Users are the people who use the products of the Project. People in this role aren't contributing code, but they are using the products, reporting bugs, making feature requests, and such. This is by far the most important category of people as, without users, there is no reason for the Project. When a user starts to contribute code or documentation patches, they become a developer.

* **DEVELOPERS**

Developer are the people who write code or documentation patches or contribute positively to the project in other ways. A developer's contribution is always recognized. In source code, all developers who contribute to a source file may add their name to the list of authors for that file.

* **OMMITTERS**

Developers who give frequent and valuable contributions to a subproject of the Project can have their status promoted to that of a "Committer" for that subproject. A Committer has write access to the source code repository and gains voting rights allowing them to affect the future of the subproject.

**3.1.2 Information description:**

This section of the SRS provides a detail of the problem that software must solve. It should describe the core of the application i.e. THE DATA. The information the software is going to work is the most basic part of the software. The description of each data or information entity is described here. It also gives details of the relationship between the data elements of the software. The information description helps the software designers in their designing purpose.

**3.1.3 Functional requirement:**

This section of the SRS describes the each function to solve the problem. It emphasizes on the core of the software on which the data will be processed-i.e. design constraints, and performance characteristics. The DFD or any other graphical diagram can also be added to describe the functionality of the system.

**3.1.4 Logical design:**

Logical design of an information system shows the major features and also how they are related to one another. The first step of the system design is to design logical design elements. This is the most creative and challenging phase and important too. Design of proposed system produces the details of the state how the system will meet the requirements identified during the system analysis that is, in the design phase we have to find how to solve the difficulties faced by the existing system. The logical design of the proposed system should include the details that contain how the solutions can be implemented. It also specifies how the database is to be built for storing and retrieving data, what kind of reports are to be created and what are the inputs to be given to the system. The logical design includes input design, output design, and database design and physical design.

**3.1.5 Input design:**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data into a usable form for processing data entry. The activity of putting data into the computer for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The system needs the data regarding the asset items, depreciation rates, asset transfer, and physical verification for various validation, checking, calculation and report generation. The error raising method is also included in the software, which helps to raise error message while wrong entry of input is done.

**3.1.6 Output design**:

Computer output is the most important and direct information source to the user. Output design is a process that involves designing necessary outputs in the form of reports that should be given to the users according to the requirements. Efficient, intelligible output design should improve the system's relationship with the user and help in decision making. Since the reports are directing referred by the management for taking decisions and to draw conclusions they must be 22 designed with almost care and the details in the reports must be simple, descriptive and clear to the user. Depending on the nature and future use of output required, they can be displayed on the monitor for immediate need and for obtaining the hardcopy. The options for the output reports are given in the appendix.

**3.1.7 Physical design:**

The term "physical design" typically refers to the process of designing the physical layout or architecture of a system or product. In the context of software development, physical design may refer to the design of the system's physical architecture or infrastructure, such as the hardware and network components that the software will run on. However, based on your description, it sounds like you might be referring to a specific phase of the software development life cycle (SDLC) that involves designing the software's functionality and creating the code that implements that functionality. This phase is commonly known as the "coding and implementation" phase or simply "implementation" phase. During the implementation phase, software developers typically take the software design specifications created during the previous phase (i.e. the "design" phase) and use them to create the actual software code. This typically involves breaking the system down into smaller, more manageable modules or components, and writing the code for each module according to its specific requirements and logic.

**3.1.8 Behavioural description**:

The behavioural description of software refers to a detailed explanation of the actions and activities that the software will perform in response to specific events or inputs. The purpose of this section is to provide a clear understanding of the expected behavior of the software from the perspective of its users or stakeholders. The behavioral description typically includes a list of events or inputs that the software will be able to receive, along with a description of the expected response or output for each event. These events could include user actions, such as button clicks or input entries, or system events, such as errors or notifications. For each event or input, the behavioral description should outline the specific operations that the software will perform in response. This may include calculations, data processing, database queries, or other actions necessary to fulfill the requirements of the software. The behavioral description should also include details about any constraints or limitations on the software's behaviour . For example, it may describe certain scenarios in which the software will produce an error or fail to perform as expected.

**3.1.9 Validation criteria:**

Validation criteria refer to the set of tests and metrics that are used to ensure that the software meets the requirements and specifications set out in the software requirements specification (SRS). These tests and metrics are designed to verify the functionality, performance, and behavior of the software, and to ensure that it meets the needs and expectations of the end-users. There are several types of validation criteria that can be used to test software, including functional testing, performance testing, usability testing, security testing, and compatibility testing. Functional testing involves testing the software to ensure that it meets the functional requirements outlined in the SRS. Performance testing involves testing the software's ability to perform under different conditions and loads, and to ensure that it meets the performance requirements outlined in the SRS. Usability testing involves testing the software's ease of use and user-friendliness, and to ensure that it meets the usability requirements outlined in the SRS. Security testing involves testing the software's ability to protect against unauthorized access and to ensure that it meets the security requirements outlined in the SRS. Compatibility testing involves testing the software's ability to work with different hardware and software configurations, and to ensure that it meets the compatibility requirements outlined in the SRS. 24 To ensure that the validation criteria are effective, they must be clearly defined and documented in the SRS. The validation criteria should also be realistic and achievable, and should be designed to provide a comprehensive assessment of the software's functionality, performance, and behavior. The testing should be carried out by qualified testers using appropriate tools and techniques, and the results of the testing should be documented and reported in a clear and concise manner.

**3.2 Technology Specific Requirements:**

**3.2.1 Hardware Requirements**:

Android mobile screen for the system users.

A smart phone with android to run the program on in the room with following specifications:-

* **Processor**: Intel Core 2 Duo+
* **RAM**: 2GB+
* **Storage**: 25MB+
* **Display**: For all Digital Devices
  + 1. **Software Requirements:**
* **Platform**: Platform Independent.
* **The Operating System:** Android, Windows and Mac.
* **Front End:** HTML, CSS, JavaScript, ReactJs.
* **Framework Used:** ExpressJs, NodeJs.
* **Databases:** MongoDB, Mongoose, MongoDB Compass.

**CHAPTER 4**

**TECHNOLOGICAL ENVIROMENT**

**4. 1 Hardware requirements:**

This project work is done on the windows 10 operating system. As operating system is a set of software tools designed to make it easy for people or programmers to make optimum use of the computer. Windows 10 is an operating system released by Microsoft. Windows 10 introduced a lot of changes, and many people found it difficult to use. Windows 10 is very similar to windows 8.1, but it addresses some of theproblems people had with windows 10. The new updating features of windows 10 are:

* **More lock screen and start screen options.**
* **Better search.**
* **3D printing support.**
* **Refreshed Xbox music.**
* **Dual screen with security.**

**4. 2 Language and Tools Used:**

* + 1. **JavaScript**
* JavaScript is a widely-used programming language that was first introduced in 1995 by Brendan Each while he was working at Netscape Communications Corporation. Initially, the language was created as a simple scripting language for web browsers, but it has since evolved to become a versatile language that can be used for a wide range of applications.
* One of the key features of JavaScript is that it is an interpreted language, which means that the code is executed without the need for a compilation step. This makes it easy to develop and test code quickly, which is one 26 reason why it is so popular for web development. Additionally, JavaScript is a high-level language, which means that it is abstracted from the machine-level details of the hardware it runs on. This makes it easier for programmers to write code without worrying about the underlying hardware.
* JavaScript is a dynamically-typed language, which means that data types are determined at runtime. This allows for more flexible programming, as variables can be re-assigned with different data types. The language is also garbage-collected, which means that the memory used by objects is automatically freed up when they are no longer needed. This makes it easier for programmers to write code without worrying about managing memory explicitly.
* JavaScript supports multiple programming paradigms, including structured, object-oriented, and functional programming. Structured programming is a procedural approach that focuses on breaking down code into smaller, more manageable chunks. Object-oriented programming is a paradigm that uses objects to encapsulate data and behavior, while functional programming emphasizes the use of functions to perform calculations and manipulate data.
* **Features of JavaScript**
* Easy to learn and use
* Interpreted Language
* Cross-Platform Language
* Free and Open Source
* Object Oriented Programming
* Large standard Library
* Dynamic Memory Allocation and etc.
  + 1. **Hypertext Markup Language (HTML):**
* Hypertext Markup Language, or HTML for short, is a standard markup language used for creating web pages and applications. It is a fundamental 27 technology used in web development, alongside Cascading Style Sheets (CSS) and JavaScript.
* HTML is a markup language, which means it is used to annotate text and add structure to content, rather than being a programming language that allows for more complex functionality. It uses a series of tags, or codes, to define different elements and structures within a web page, such as headings, paragraphs, images, and links.
* HTML is based on a tree-like structure, where each tag defines a specific part of the page, with the content within the tag representing the information that will be displayed to the user. Tags can be nested within each other to create more complex structures, such as lists, tables, and forms.
* One of the key benefits of HTML is its cross-platform compatibility,• meaning that it can be viewed on a variety of different devices and platforms, including desktop computers, laptops, tablets, and smartphones. This is achieved through the use of responsive design techniques, which allow web developers to create web pages that adapt and adjust their layout and content based on the device they are being viewed on.
  + 1. **Cascading Style Sheet (CSS)**
* CSS, short for Cascading Style Sheets, is a styling language used for web development. It is responsible for the visual presentation of a webpage and allows web developers to create visually appealing and user-friendly websites.
* CSS works by selecting HTML elements and defining their styles, such as the font, color, size, and layout. It also allows for the creation of complex visual effects like gradients, animations, and transitions.
* One of the key features of CSS is its cascading nature, where styles can be inherited from parent elements and overridden by child elements. This allows for a consistent look and feel throughout a website, while still allowing for customization and variation.
* CSS is also capable of responsive design, where a website can adapt its layout and styles based on the device and screen size of the user. This is 28 crucial in today's mobile-first world, where the majority of internet usage is on smart phones and tablets.
  + 1. **ReactJS**
* ReactJS is an open-source JavaScript library that is used for building user interfaces. It was developed by Facebook and has gained immense popularity among web developers due to its flexibility, reusability, and ease of use. ReactJS allows developers to build complex UIs with ease, making it one of the most preferred tools for building web applications.
* One of the most significant advantages of ReactJS is its ability to create reusable UI components. These components are modular and can be used multiple times within an application, making development faster and more efficient. Additionally, ReactJS uses a virtual DOM (Document Object Model) which allows it to update the UI quickly and efficiently, providing a seamless user experience.
* ReactJS is also highly flexible and can be used with other libraries or frameworks. This means that developers can use ReactJS to build small components within a larger application or use it as the primary library for building an entire application. This flexibility has made ReactJS a popular choice for building single-page applications (SPAs) and progressive web applications (PWAs).
* Another feature that makes ReactJS popular is its ability to handle data seamlessly. With ReactJS, data can be passed down through the component tree using props, making it easy to manage and update data throughout an application. This feature ensures that data is always up-to-date and eliminates the need for developers to write complex code to manage data.
  + 1. **NodeJS**
* Node.js is an open-source server-side runtime environment built on the Chrome V8 JavaScript engine. It allows developers to build fast, scalable, and highly network applications. Node.js was created by Ryan Dahl in 2009 and has since grown to become one of the most popular JavaScript frameworks for server-side development.
* Node.js is highly scalable and can handle large amounts of concurrent connections. This is because it uses an event-driven, non-blocking I/O model, which allows it to handle multiple requests simultaneously without waiting for one request to finish before moving on to the next. This makes it an ideal choice for building real-time applications such as chat applications, online gaming, and collaboration tools.
* Another benefit of Node.js is that it allows developers to use JavaScript both on the front-end and back-end of their applications. This means that developers can use the same language and programming paradigms throughout their entire stack, which can make development more efficient and easier to maintain.
* Node.js also has a vast ecosystem of libraries, modules, and tools that developers can use to build their applications. The Node Package Manager (NPM) is a centralized repository for Node.js packages, which makes it easy for developers to find and install the packages they need for their projects.
* Some of the popular frameworks built on top of Node.js include Express, NestJS, and Meteor. These frameworks provide additional features and functionality to help developers build complex applications more quickly.
  + 1. **ExpressJS**
* ExpressJS is a popular web framework for building robust and scalable web applications using Node.js. It is open-source, lightweight, and provides a simple, flexible, and powerful set of features for creating web applications and APIs.
* One of the main advantages of using ExpressJS is its simplicity. With its minimalistic and unopinionated design, ExpressJS allows developers to build web applications quickly and easily, without imposing a rigid structure or specific coding conventions. This makes it easy to customize and extend the framework to suit your specific needs and preferences.
* ExpressJS provides a range of features that are essential for building modern web applications. It supports routing, middleware, templating, and more, making it a versatile framework that can handle a wide range of tasks. ExpressJS is also highly modular, which means that developers can choose which components to use and how to integrate them into their application.
* ExpressJS is also known for its robustness and scalability. It provides a solid foundation for building complex web applications that can handle high traffic and large amounts of data. It also integrates well with other Node.js modules and packages, making it easy to add additional functionality as needed.
* One of the key features of ExpressJS is its middleware architecture. Middleware functions can be used to modify request and response objects, perform authentication and authorization checks, handle errors, and more. This makes it easy to add custom logic and functionality to your application without cluttering up your main codebase.
  + 1. **MongoDB**
* MongoDB is a popular NoSQL database that was first released in 2009. It is designed to handle large amounts of unstructured and semi-structured data, making it ideal for use in modern web applications, mobile applications, and IoT (Internet of Things) systems. Unlike traditional relational databases, MongoDB is a document database, which means that it stores data in flexible JSON-like documents rather than rigid tables.
* One of the key benefits of using MongoDB is its flexible data model. MongoDB allows you to store data in a variety of different structures, including arrays, nested objects, and key-value pairs. This makes it easy to work with unstructured data, and to store data in a way that makes sense for your specific use case.
* Another advantage of using MongoDB is its full indexing support.• MongoDB provides a wide range of indexing options, including simple and compound indexes, geospatial indexes, and text indexes. This allows you to optimize your queries for the specific needs of your application, and to ensure that your queries run quickly and efficiently.
* MongoDB also provides built-in replication and sharding capabilities,• which make it easy to scale your application as your data grows. With MongoDB, you can replicate your data across multiple nodes for high availability, and shard your data across multiple clusters for horizontal scaling.

**CHAPTER 5**

**DESIGN**

Design is a meaningful engineering representation of something that is to be built. It can be traced to a customer’s requirements and at the same time assessed for quality against asset of predefined criteria for the good design. Thus it is the blue print of the system. In the software engineering context design concentrates on three factors, data, architecture, interfaces and components. The data and architectural design focuses on patterns as they apply to the application to be built. At the interface level human ergonomics often dictate our design approach. At the component level a programming approach leads to the effective data and procedural designs. Design begins with requirements model and is processed to transform it into four levels of design details; data structure, system architecture, interface representation and the component level. Finally a design specification is produced. This specification consists of design models that describe data, architecture, interfaces and components. At each stage, software design work products are renewed for clarity, correctness, completeness and consistency with the requirements and with one another.

**5.1 ER Diagram**

* Data models are tools used in analysis to describe the data requirements and assumptions in the system from a top-down perspective. It is a major data modeling tool & will help organizing the data in our project into entities & define the relationship between the entities. They also set the stage for the design of databases later on in the SDLC.
* The Entity-Relationship model is a data model for high-level descriptions of conceptual data models, and it provides a graphical notation for representing such data models in the form of entity-relationship diagrams. Such data models are typicallyused in the first stage of information-system design; they are used, for example, to describe information needs and/or the type of information that is to be stored in the database during the requirements 33 analysis phase of software development.
* An entity-relationship (ER) diagram is a type of visual representation of the data model that describes the entities and relationships between them. The ER diagram shows the structure of the database, including tables, columns, keys, and relationships between tables. It is a powerful tool for designing databases as it helps to visualize the relationships between different tables, and it can also be used to create a database schema.
* In an ER diagram, entities are represented by rectangles, and the relationships between entities are represented by lines connecting the rectangles. The relationships between entities can be of different types, such as one-to-one, one-to-many, or many-to-many. One-to-one relationships indicate that one instance of an entity is related to only one instance of another entity.
* An ER diagram is often used in the initial stages of software development to help with requirements analysis, and it can be used throughout the design and implementation phases. The diagram helps to identify the data requirements of the system and can be used to communicate these requirements to stakeholders. It can also help to identify potential issues or conflicts in the data model before implementation, saving time and resources.

**5.1.1 Basic Elements in ER Model**

* **Entity** - a class of real world objects having common characteristics and properties about which we wish to record information.
* **Relationship** - an association among two or more entities.
* **Occurrence** - instance of a relationship is the collective instances of the relatedentities.
* **Degree** - number of entities associated in the relationship (binary, ternary,other n-ray).
* **Connectivity** - one-to-one, one-to-many, many-to-many.
* **Existence dependency (constraint**) - optional/mandatory.
* **Attribute** - a characteristic of an entity or relationship.
* **Identifier** - uniquely determines an instance of an entity.
* **Identity dependence** - when a portion of an identifier is inherited from anotherentity.
* **Multi-valued** - same attribute having many values for one entity.
* **Surrogate** - system created and controlled unique key.
* **Cardinality** - Cardinality defines the number of occurrences of one entity for asingle occurrence of related entity.

**5.1.2 Symbols Used for ER Diagram**

Rectangle are used for showing the entities in the data base

Ovals are used to represent the attributes of a relation.

Rhombus is used for representing the relationship b/t entity

Multi valued attributes are depicted by double ellipse

Derived attributes are depicted by dashed ellipse

**5.1.3 ER Diagram**

Sign Up

Details

Verify

Log In

***Figure 5.1 ER Diagram for Signup And Login Page***

Home Page

Add Location

Search

Hit APIs

Result

***Figure 5.2 Search Panel for weather***

**5. 2 Data Flow Diagram**

* A data flow diagram (DFD) is a powerful tool used to model the flow of data through an information system. It's a graphical representation that illustrates the movement of data and the processes that take place within a system. This diagram can be used as a preliminary step to give an overview of a system, which can later be elaborated and analyzed in greater detail.
* Data flow diagrams can be used for various purposes, including the visualization of data processing for structured design. This is particularly useful in software development, where developers can use DFDs to design systems that are efficient, reliable, and secure.
* DFDs typically consist of four main components: processes, data stores, data flows, and external entities. Processes represent the various functions that are performed within the system, while data stores represent the locations where data is stored. Data flows are the paths that data takes as it moves through the system, and external entities represent the sources and destinations of the data.
* DFDs are also useful for identifying potential bottlenecks and areas for improvement within a system. By analyzing the flow of data through a system, designers can identify areas where data may be delayed or lost, as well as areas where data security may be compromised. This information can be used to optimize the system and ensure that it functions as efficiently and effectively as possible.

**5.2.1 Level-0 DFD**

Search

***Figure 5.3 Level-0 DFD***

**5.2.2 Level-1 DFD**

HomePage

SignUp/login

Verification

Weather Dashboard

Add location

Search

Hit Rest APIs

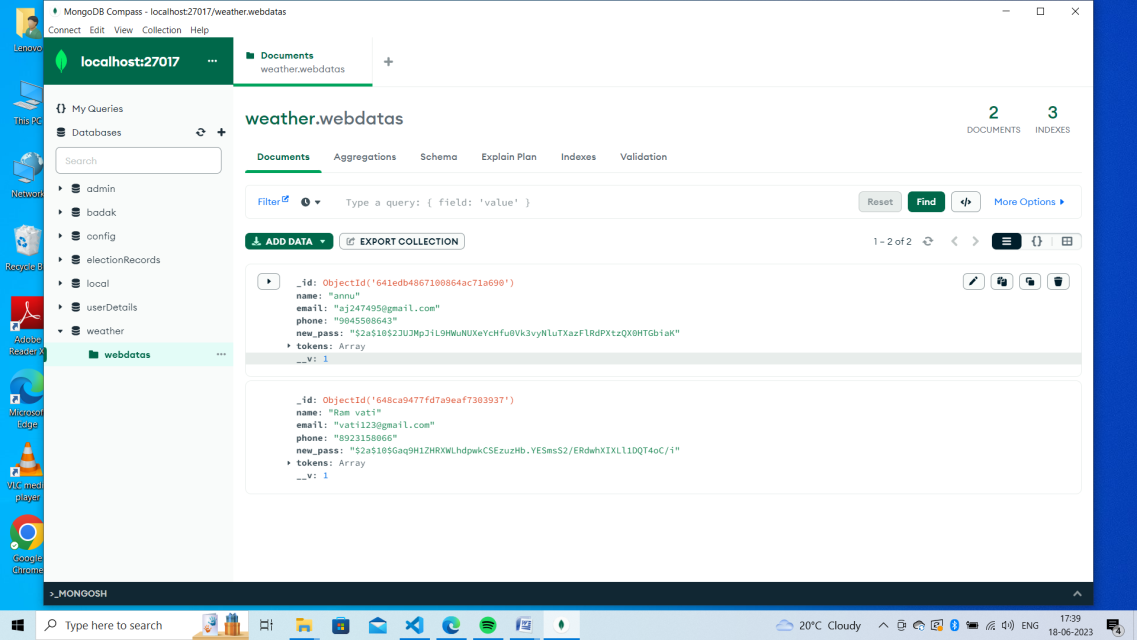
Result

Logout

***Figure 5.4 Level-1 DFD***

**5.3 Collections**

* A collection in MongoDB is a logical grouping of document fields that are stored together in a single database. Each document within a collection can have its own unique set of fields and values, but all documents within a collection have a similar structure.
* In terms of relational databases, a collection in MongoDB is equivalent to a table. However, there are some differences between the two. For example, in a table, each row has the same columns, but in a collection, each document can have its own set of fields.
* Collections are an essential part of the MongoDB data model, as they provide a way to organize related data in a flexible and scalable manner. They allow developers to store and retrieve data in a way that makes sense for their particular application, without having to adhere to a rigid schema. It's worth noting that collections in MongoDB do not enforce a schema, which means that documents within a collection can have different fields and data types. This flexibility allows developers to easily add or remove fields as their application evolves, without having to worry about altering a rigid schema.
* Collections also have a unique identifier known as an Object ID, which is automatically generated by MongoDB when a document is inserted into a collection. This identifier allows developers to uniquely identify and retrieve individual documents within a collection, making it easy to update or delete specific data points as needed.

**5.3.1 User Details**

**CHAPTER 6**

**CODING OF THE PROJECT**

**6. 1 Structure of the project directory**

* In a typical web development project like Internet voting, the project directory plays a crucial role in organizing all the necessary files and resources needed to build and deploy the application. The structure of the project directory can vary depending on the specific needs of the project, but there are some common practices that are typically followed.
* The Internet voting project directory can be divided into several main directories, each with its specific purpose. These directories include: Client directory: This directory contains all the client-side files, including the front-end code written in React, HTML, CSS, and JavaScript. It includes folders like components, public, and src, where the code for the user interface is written.
* Server directory: This directory contains all the server-side files, including the back-end code written in Node.js, and any other tools or frameworks that are used on the server side, like Express or MongoDB. It includes folders like routes, controllers, and models where the code for server-side functionality is written. Config directory: This directory contains all the configuration files, including environment variables, authentication keys, and any other settings that are required to run the application.
* Public directory: This directory contains all the static files that are served to the client, such as images, fonts, and other media.
* Test directory: This directory contains all the test cases that are used to test the• application, including unit tests, integration tests, and end-to-end tests.
* Documentation directory: This directory contains all the documentation related to the project, including user manuals, developer guides, and API documentation.
* In addition to these main directories, the project directory may also contain other files and directories that are specific to the project, such as a package.json file that contains all the dependencies needed to run the project, a README file that provides information about the project, or a .gitignore file that specifies which files should be excluded from version control.
* Overall, the structure of the project directory plays a crucial role in ensuring that the project is well-organized, maintainable, and scalable. By following best practices and maintaining a consistent structure, developers can work more efficiently and effectively, and ensure that the project is successful in meeting its goals.

expressWeb

public

src

Images

Index.html

Index.css

dbs

templates

Middleware

Index.js

………

………

………

………

***Figure 6.1 Directory of the Project***

**6.2 Coding of the Project**

**6.2.1 index.html (Home page)**

<!DOCTYPE html>

<html lang="en">

    <head>

        <title>WeatherWeb.in</title>

        <link rel="styleSheet" href="index.css">

    </head>

    <body>

       <div class="header">

           <div class="logo"><p>B Weather</p></div>

           <div class="home"><a href="index">Home</a></div>

           <div class="weather"><a href="weather">Weather</a></div>

           <div class="about"><a href="about">About</a></div>

           <div class="login"><a href="login">Login</a></div>

       </div>

       <div class="container">

           <div class="child\_1">

               <div class="par\_1">Welcome to Weather Web</div>

               <div class="par\_2">Get The Letest <span>Weather</span> Info Of Your City</div>

               <button><a href="weather">Check Now</a></button>

            </div>

           <div class="child\_2">

               <div class="sub\_child\_1">

                <div class="img\_1">

                    <img src="images/1.png" width="100%" height="100%">

                </div>

                <div class="img\_2">

                    <img src="images/2.png" width="100%" height="100%">

                </div>

               </div>

               <div class="sub\_child\_2">

                <img src="images/4.png" width="80%" height="100%">

               </div>

           </div>

           <div class="footer">

            <h5>All Copy Right Are Reserved</h5>

        </div>

       </div>

    </body>

   <script>

       const sub\_child\_1 = document.querySelector(".sub\_child\_1");

       const sub\_child\_2 = document.querySelector(".sub\_child\_2");

       const left\_to\_right  = () =>{

           setTimeout(() => {

               sub\_child\_1.style.transform = "translateX(-105%)"

           },2000);

           setTimeout(() => {

               sub\_child\_1.style.transform = "translateX(-215%)"

           },8000);

       }

               setTimeout(() => {

               sub\_child\_2.style.transform = "translate(-105%)"

           },11000);

        //   window.addEventListener('load',left\_to\_right);

        left\_to\_right();

   </script>

</html>

**6.2.2 index.css (Home page)**

\*{

    padding:0px;

    margin:0px;

    box-sizing:border-box;

    background-size:cover;

    font-family:verdana;

}

.header{

    display:flex;

    /\* border:1px solid red; \*/

    position:relative;

    flex-direction:row;

    height:70px;

    padding-left:100px;

    padding-right:100px;

    width:100%;

    background:#073b4c;

    font-size:17px;

    text-align:center;

    color:white

}

.header div a{

    text-decoration: none;

    color: #fff;

}

.logo{

    position:absolute;

    width:10%;

    /\* border:1px solid red; \*/

    height:100%;

    font-size:20px;

    text-align:center;

    color:white;

    padding-top:10px;

    cursor:pointer;

}

.logo p:first-letter{

    color: #8ecae6;

    font-size:30px;

}

.home{

    position:absolute;

    right:400px;

    width:5%;

    top:22px;

    cursor:pointer;

 }

.weather{

    position:absolute;

    right:300px;

    width:5%;

    top:22px;

    cursor:pointer;

 }

.about{

   position:absolute;

   right:200px;

   width:5%;

   top:22px;

   cursor:pointer;

}

.login{

    position:absolute;

    right:100px;

    width:5%;

    top:22px;

    cursor:pointer;

 }

.container{

    display:relative;

    /\* border:1px solid red; \*/

    width:90%;

    margin-left:5%;

    height:680px;

}

.child\_1{

    position:absolute;

    width:50%;

    /\* border:1px solid green; \*/

    height:40%;

    top:300px

}

.child\_1 a{

    text-decoration:none;

    color:#000;

}

.child\_2{

    position:absolute;

    /\* border:1px solid red; \*/

    width:40%;

    height:50%;

    right:5%;

    top:200px;

    overflow:hidden;

}

.child\_1 .par\_1{

    /\* border: 1px solid red; \*/

    font-size:15px;

    width:30%;

    color: #6b705c;

}

.par\_2{

    /\* border:1px solid black; \*/

    width:75%;

    font-size:25px;

    margin-top:20px;

}

.par\_2 span{

    color: #219ebc;

}

.child\_1 button{

    font-size:15px;

    margin-top:40px;

    border-radius:30px;

    padding:4px;

    border:none;

    background:#b7b7a4;

    box-shadow: 2px 2px 5px grey;

    color:#212529;

    cursor: pointer;

}

.child\_1 button:hover{

    box-shadow: 2px 2px 10px #a8dadc;

}

.sub\_child\_1{

    position:absolute;

    width:90%;

    left:100%;

    height:100%;

    transition-property: all;

    transition-duration: 2s;

    transition-timing-function: linear;

}

.img\_1{

    width:35%;

    height:35%;

}

.img\_2{

    width:65%;

    height:80%;

    margin-left:35%;

    margin-top:-70px;

}

.sub\_child\_2{

    position:absolute;

    width:90%;

    left:100%;

    height:100%;

    transition-property: all;

    transition-duration: 2s;

    transition-timing-function: linear;

}

.footer{

    position:fixed;

    /\* border:1px solid red; \*/

    width:70%;

    height:50px;

    font-size:verdana;

    bottom:30px;

    left:15%;

    text-align:center;

    line-height:50px;

    border-radius:8px;

    background:#caf0f8;

    color:#264653;

    box-shadow:2px 2px 10px grey;

}

**6.2.3 login.hbs (login page)**

**<!doctype html>**

**<html lang="en">**

**<head>**

**<title>weather.Co</title>**

**</head>**

**<body>**

**{{>header}}**

**<div class="main\_container">**

**<div class="child\_1">**

**<form action="/weather" method="post">**

**<div class="img\_1">**

**<img src="https://www.transparentpng.com/thumb/user/gray-user-profile-icon-png-fP8Q1P.png" alt="gray user profile icon png @transparentpng.com">**

**</div>**

**<input type="email" name="email" placeholder="Enter Email Address" class="email">**

**<input type="password" name="pass" placeholder="Enter Password" class="password">**

**<input type="submit" name="btn" value="Login" class="btn">**

**<span class="fPass"><a href="forgetPass">forget password</a></span>**

**</form>**

**</div>**

**<div class="child\_2">**

**<img src="images/10.png" width="100%" height="60%" style="margin-top: 25%;">**

**</div>**

**</div>**

**<span class="register">**

**<a href="register">Register</a>**

**</span>**

**</body>**

**<style>**

**.main\_container{**

**position:relative;**

**width:50%;**

**height:500px;**

**top:100px;**

**left:25%;**

**display:grid;**

**grid-template-columns:repeat(2,1fr);**

**box-shadow: 2px 2px 10px grey;**

**background:#073b4c;**

**border-radius: 5px;**

**}**

**form{**

**display: block;**

**background:#e8e8e4;**

**width:90%;**

**height:71%;**

**margin-top:15%;**

**margin-left:10%;**

**border-radius: 2px;**

**}**

**form input{**

**border:none;**

**outline-color: #8ecae6;**

**width:90%;**

**height:50px;**

**margin-left: 5%;**

**margin-top:20px;**

**font-family: Verdana, Geneva, Tahoma, sans-serif;**

**padding:10px;**

**}**

**.img\_1{**

**width:30%;**

**height:100px;**

**border-radius:40%;**

**margin-left:35%**

**}**

**.child\_1 img{**

**width:95%;**

**height:95%;**

**margin-top: 10px;**

**}**

**.btn{**

**background:#8ecae6;**

**}**

**.fPass{**

**position:absolute;**

**display:block;**

**font-size:13px;**

**text-decoration:block;**

**left:60px;**

**top:375px;**

**}**

**.register{**

**font-size: 20px;**

**width:130px;**

**height:50px;**

**font-family: Verdana, Geneva, Tahoma, sans-serif;**

**display: block;**

**position:absolute;**

**background:#073b4c;**

**right:60px;**

**top:90%;**

**border-radius:5px;**

**text-align: center;**

**line-height:45px;**

**transition-property:all;**

**transition-timing-function: linear;**

**transition-duration:0.5s;**

**}**

**.register:hover{**

**transform: translateY(-10px);**

**}**

**.register a{**

**color:white;**

**text-decoration: none;**

**}**

**</style>**

**</html>**

**6.2.4 register.hbs (Sign Up Page)**

**<!doctype html>**

**<html lang="en">**

**<head>**

**<title>weather.Co</title>**

**</head>**

**<body>**

**{{>header}}**

**<div class="main\_container">**

**<div class="img\_1">**

**<img src="https://www.transparentpng.com/thumb/user/add-user-male-transparent-icon--6vjB7f.png" alt="add user male transparent icon @transparentpng.com">**

**</div>**

**<form action="/login" method="post">**

**<input type="text" name="name" placeholder="Enter Username" class="name" required="required">**

**<input type="email" name="email" placeholder="Enter Email Address" class="email" required="required">**

**<input type="number" name="phone" placeholder="Enter Mobile Number" class="number" required="required">**

**<input type="password" name="new\_pass" placeholder="Create Password" class="new\_pass" required="required">**

**<input type="password" name="com\_pass" placeholder="Confirm Password" class="com\_pass" required="required">**

**<input type="submit" name="btn" value="Register" class="btn">**

**</form>**

**</div>**

**</body>**

**<style>**

**.main\_container{**

**position:relative;**

**display: block;**

**width:30%;**

**height:500px;**

**background:#e8e8e4;**

**left:35%;**

**top:100px;**

**box-shadow: 2px 2px 10px 2px;**

**}**

**.img\_1{**

**position:absolute;**

**width:30%;**

**height:20%;**

**left:35%;**

**top:8%;**

**}**

**.img\_1 img{**

**width:100%;**

**height:100%**

**}**

**form{**

**display: block;**

**position:absolute;**

**top:28%;**

**height:72%;**

**}**

**form input{**

**width:90%;**

**height:40px;**

**margin-left:5%;**

**margin-top:15px;**

**padding:10px;**

**font-family: Verdana, Geneva, Tahoma, sans-serif;**

**border:none;**

**outline-color:#8ecae6 ;**

**}**

**.btn{**

**background:#8ecae6;**

**font-family: Verdana, Geneva, Tahoma, sans-serif;**

**}**

**</style>**

**</html>**

**6.2.5 weather.hbs (Weather Page)**

<!doctype html>

<html>

    <head>

        <title>weatherWeb.in</title>

        <script src="https://use.fontawesome.com/83b0b88390.js"></script>

    </head>

  <body>

  {{>header}}

  <form id="submit" class="src\_tmp">

      <input placeholder="Enter City Name..." id="inputField">

      <input type="button" id="submit\_btn" value="Serach">

  </form>

  <div class="container">

     <div class="date\_time">

        <div class="time" style="padding-left: 10px;"><p>17:53:12</p></div>

        <div class="date"><p>Son|Nov|17|2021</p></div>

     </div>

     <div class="city\_and\_cuntory">

        <p id="city\_name">Enter City Name And Get Today Weather Report</p>

     </div>

     <div class="weather\_report">

        <h1 class="weatherStatus"style="padding-top:40px; text-align:center;">27.40 ℃</h1>

        <span class="weatherIcon"style="color:#ffff;"><img src='images/9.png' width='100%' height='100%'></span>

     </div>

 </div>

 {{>footer}}

  </body>

  <style>

      \*{

          padding:0px;

          margin:0px;

          box-sizing:border-box;

          background-size:cover;

          font-family:Verdana,,Tahoma,sans-serif;

      }

      .src\_tmp{

          position: relative;

          display: block;

          width:40%;

          height:50px;

        /\*  border:1px solid red;\*/

          left:30%;

          top:50px;

      }

      #inputField{

          position:absolute;

          width:70%;

          height:100%;

          left:10px;

          border-radius:5px;

          border:2px solid black;

          padding:5px;

          color:black;

      }

      #submit\_btn{

          position: absolute;

          right:10px;

          width:25%;

          height:100%;

          padding:10px;

          font-size:20px;

          border-radius:10px;

          background:#073b4c;

          border:none;

          color: #ffff;

      }

      .container{

           position:relative;

           display:grid;

           width:40%;

           height:300px;

           left:30%;

           top:120px;

           background:#081c15;

           border-radius:8px;

           box-shadow: 5px 5px 7px #081c15;

      }

      .date\_time{

          position:absolute;

          width:100%;

          height:60px;

          border-radius: 8px;

          background:#001219;

          display: grid;

          grid-template-columns: 25% 25%;

          grid-column-gap: 50%;

      }

      .date\_time div{

          color:#FFFF;

          padding-top:15%;

      }

      .city\_and\_cuntory{

          position:absolute;

          display: grid;

          height:50px;

          top:80px;

          color:grey;

          font-size:22px;

          left:20px;

          font-size:15px;

        }

      .city\_and\_cuntory span{

          display: block;

          height:50px;

          padding:5px;

          line-height:30px;

        }

        .weather\_report{

            position:absolute;

            display: grid;

            width:615px;

            height:150px;

            top:150px;

            grid-template-columns: 60% 30%;

        }

        .weather\_report h1{

            font-size:45px;

            color:#FFF;

        }

        .weatherStatus{

            font-size: 40px;

        }

        .weather\_report span {

            height:150px;

        }

  </style>

  <script>

      const date          = document.querySelector(".date");

      const time          = document.querySelector(".time");

      const weatherStatus = document.querySelector(".weatherStatus");

      const weatherIcon   = document.querySelector(".weatherIcon");

      const btn           = document.getElementById("submit\_btn");

      const city\_name     = document.getElementById("city\_name");

      const cityName      = document.getElementById("inputField");

      // console.log(cityName)

  // Api call through the async mathode

    const getInfo = async(event) =>{

         event.preventDefault();

         if(cityName.value ===""){

             city\_name.innerText = "Please Enter The Valid City Name..."

         }else{

           try{

              let url = "https://api.openweathermap.org/data/2.5/weather?q="+cityName.value+"&appid=d08b23284e6883e7922aa2308be29afb";

              const response = await fetch(url);

              //change data json to object

              const data     = await response.json();

              const arrData  = [data]

              //console.log(arrData[0].name)

              city\_name.innerText= arrData[0].name+","+arrData[0].sys.country;

              const kelvin= arrData[0].main.temp;

              const celcius= Math.trunc(kelvin-273.15);

              weatherStatus.innerText = celcius+"℃";

              const weatherIcon\_1 = arrData[0].weather[0].main;

              //console.log(weatherIcon\_1)

              if(weatherIcon\_1 == "Sunny"){

                  weatherIcon.innerHTML = "<img src='images/9.png' width='100%' height='100%'>"

              }else if(weatherIcon\_1 == "Clouds"){

                  weatherIcon.innerHTML = "<img src='images/8.png' width='100%' height='100%'>"

              }else if(weatherIcon\_1 == "Clear"){

                  weatherIcon.innerHTML = "<img src='images/9.png' width='100%' height='100%'>"

              }else{

                  weatherIcon.innerHTML = "<img src='images/7.png' width='100%' height='100%'>"

              }

           }

              catch{

                console.log(error)

                }

            }

        }

     btn.addEventListener("click",getInfo);

      const currentTime = new Date();

       setTimeout(() =>{

             time.innerText = currentTime.getHours()+":"+currentTime.getMinutes();

      },1000);

      const setDay   = ["Sun","Mon","Tue","Wed","Thu","Fri","Set"];

      const setMonth = ["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"];

      const day   = currentTime.getDay();

      const month = currentTime.getMonth();

      date.innerText = setDay[day]+"|"+setMonth[month]+"|"+currentTime.getDate()+"|"+currentTime.getFullYear();

     </script>

</html>

<!doctype html>

<html lang="en">

   <head>

       <title>weather.Co</title>

   </head>

   <body>

       {{>header}}

       <form action="/name" method="patch" class="f\_pass">

            <input type="password" name="pass\_1" required="required" class="email" placeholder="Enter new password">

            <input type="password" name="pass\_2" required="required" class="email" placeholder="Confirm password">

            <input type="submit" value="Save" class="sub">

        </form>

   </body>

   <style>

        .f\_pass{

            position:absolute;

            display:block;

            width:20%;

            height:180px;

            background:#e8e8e4;

            left:40%;

            top:270px;

        }

        .f\_pass input{

            margin-top:20px;

            width:90%;

            margin-left:5%;

            line-height:20px;

            height:30px;

            outline-color:#8ecae6;

            padding:10px;

            font-size:10px;

        }

        .f\_pass .sub{

            width:100px;

            font-size:10px;

            margin-left:100px;

            padding:0px;

        }

    </style>

</html>

<!DOCTYPE html>

<html lang="en">

    <head>

        <title>WeatherWeb.in</title>

    </head>

    <body>

       {{>header}}

       <div class="container">

           <div class="child\_1">

               <div class="par\_1">Welcome to Weather Web</div>

               <div class="par\_2">I Am A <span> Mern Stack Devloper </span> Fresher And Still Pursuing In B.tech</div>

               <button><a href="https://www.instagram.com/badal\_k09/">Explore</a></button>

            </div>

           <div class="child\_2">

               <img src="images/5.png" width="70%" height="100%">

          </div>

          {{>footer}}

       </div>

    </body>

    <style>

        \*{

    padding:0px;

    margin:0px;

    box-sizing:border-box;

    background-size:cover;

    font-family:verdana;

}

.container{

    display:relative;

    /\* border:1px solid red; \*/

    width:90%;

    margin-left:5%;

    height:680px;

}

.child\_2{

    position:absolute;

    width:40%;

    /\* border:1px solid green; \*/

    height:50%;

    top:200px;

    padding-left: 10%;

}

.child\_1{

    position:absolute;

    /\* border:1px solid red; \*/

    width:50%;

    height:40%;

    right:5%;

    top:300px;

    overflow:hidden;

    padding-left:20px;

}

.child\_1 a{

    text-decoration: none;

    color:#000;

}

.child\_1 .par\_1{

    /\* border: 1px solid red; \*/

    font-size:15px;

    width:30%;

    color: #6b705c;

}

.par\_2{

    width:72%;

    font-size:25px;

    margin-top:20px;

    line-height: 35px;

    /\* text-align:center \*/

}

.par\_2 span{

    color: #219ebc;

}

.child\_1 button{

    font-size:15px;

    /\* margin-left:18px; \*/

    margin-top:40px;

    border-radius:30px;

    padding:7px;

    border:none;

    background:#b7b7a4;

    box-shadow: 2px 2px 5px grey;

    color:#212529;

    cursor: pointer;

}

.child\_1 button:hover{

    box-shadow: 2px 2px 10px #a8dadc;

}

 </style>

 </html>

<!doctype html>

<html lang="en">

    <head>

        <title>Weather.Co</title>

    </head>

    <body>

        {{>header}}

        <form action="/userDetails" method="post" class="f\_pass">

            <input type="email" name="email" required="required" class="email" placeholder="Enter Email">

            <input type="number" name="number" required="required" class="email" placeholder="enter Mobile Number">

            <input type="submit" value="Change Password" class="sub">

        </form>

    </body>

    <style>

        .f\_pass{

            position:absolute;

            display:block;

            width:20%;

            height:180px;

            background:#e8e8e4;

            left:40%;

            top:270px;

        }

        .f\_pass input{

            margin-top:20px;

            width:90%;

            margin-left:5%;

            line-height:20px;

            height:30px;

            outline-color:#8ecae6;

            padding:10px;

            font-size:10px;

        }

        .f\_pass .sub{

            width:100px;

            font-size:10px;

            margin-left:100px;

            padding:0px;

        }

    </style>

</html>

**6.2.6 Index.js**

const { log } = require("console");

const express = require("express");

const app     = express();

const hbs     = require("hbs");

const port    = process.env.PORT ||10000;

const path    = require("path");

const bcryptjs  =require("bcryptjs");

const jwt = require("jsonwebtoken");

const cookieParser = require("cookie-parser");

const auth = require("../src/middleware/auth");

// console.log(auth)

const partialsPath  = path.join(\_\_dirname,"./templates/partials");

const template\_path = path.join(\_\_dirname,"./templates/views");

// console.log(template\_path)

const staticPath    = path.join(\_\_dirname,"../public");

// console.log(staticPath);

 app.set("view engine","hbs");

 app.set('views',template\_path);

 app.use(cookieParser());

 app.use(express.static(staticPath));

 hbs.registerPartials(partialsPath);

 require("./dbs/models/conn");

 app.use(express.json());

 app.use(express.urlencoded({extended:false}));

 const webdata = require("./dbs/models/data");

 const register = require("../../registration/src/dbs/models/regis");

//  console.log(webdata);

app.get("/home",(req,res) =>{

    res.render("index");

});

app.get("/about",(req,res) =>{

    res.render("about");

});

app.get("/weather",auth,(req,res) =>{

    res.render("weather");

});

app.get("/login",(req,res)=>{

    res.render("login");

});

app.get("/register",(req,res)=>{

    res.render("register");

});

app.post("/login",async(req,res)=>{

    try {

        const pass\_1 = req.body.new\_pass;

        const pass\_2 = req.body.com\_pass;

        const email = req.body.email;

        // const emailData = await webdata.findOne({email})

        //  if(emailData.email === email){

            //  res.send("email address already exist");

        //  }else

          if(pass\_1 === pass\_2){

             const userData = new webdata({

                 name:req.body.name,

                 email:req.body.email,

                 phone:req.body.phone,

                 new\_pass:req.body.new\_pass

             });

             const user = await userData.save();

            //  console.log(user)

              res.status(201).render("login");

         }else{

             res.send("password are matched...")

         }

    }catch(e){

        res.status(404).send(e)

    }

});

app.post("/weather",async(req,res)=>{

    try {

        const email = req.body.email;

        const pass = req.body.pass;

        const user = await webdata.findOne({email});

        const matchPass = await bcryptjs.compare(pass,user.new\_pass);

        if(matchPass){

             const token = await user.generateAuthToken();

             res.cookie("jwt",token,{

               expires:new Date(Date.now()+500000),

               htmlOnly:true,

           });

             res.render("weather")

        }else{

             res.send("invalid login details");

        }

     }catch(error){

        res.status(404).send(error );

    }

});

app.get("/forgetPass",(req,res)=>{

    res.render("forgetPass");

});

app.post("/userDetails",async(req,res)=>{

    try{

    const email = req.body.email;

    const number = req.body.number;

    const user = await webdata.findOne({email});

       if(user.phone==number){

           res.status(200).render("userDetails");

       }else{

           res.send("Pls Enter valid email or phone No...")

       }

   }catch(e){

    res.status(404).send(e)

   }

});

app.get("\*",(req,res) =>{

    res.render("404");

});

// const token = async()=>{

//        const to =  jwt.sign({\_id:"sdjoisjdsojdsddojohdofifhfs"},"bafdalkumarsinghansdiaadeceloperandaprohrammert");

//        console.log(to)

// }

// token()

app.listen(port,()=>{

    console.log("Your Port No. Is "+port+" And Local Server Is Created");

});

// palindrom Number program....

// const fs = require("prompt-sync");

// const prompt = fs();

// const palindrom = () =>{

//        value = prompt("Enter The Number = ");

//        var palindromNo = value.toString().split("").reverse().join("");

//                 if(palindromNo.endsWith("-")){

//                   palindromNo = parseInt(palindromNo);

//                   palindromNo = ("-").concat(palindromNo)

//                   if(palindromNo == value){

//                       return palindromNo+" This Number Is palindrom..."

//                          }else{

//                       return palindromNo+" This Number Is not Palindrom..."

//                          }

//                         }else if(palindromNo == value){

//                           return palindromNo+" This number Is Palindrom..."

//                         }else{

//                            return palindromNo+" This Number Is Not palindron..."

//                         }

//                     }

//  console.log(palindrom())

// factorial program...........

// var fact = prompt("Pls Enter The Factorial No = ");

//  for(i = fact-1; i>=1; i-- ){

//      var fact  = fact\*i;

//  }

//  console.log(fact)

// fabonachi series program....

//   var fab = 10;

//   for(i = 0; i<=10; i++){

//       var x = 0;

//       var y = 1;

//     fab = x+y;

//      x=y

//      y = fab

//      console.log(fab)

//   }

app.get("/home",(req,res) =>{

    res.render("index");

});

app.get("/about",(req,res) =>{

    res.render("about");

});

app.get("/weather",auth,(req,res) =>{

    res.render("weather");

});

app.get("/login",(req,res)=>{

    res.render("login");

});

app.get("/register",(req,res)=>{

    res.render("register");

});

app.post("/login",async(req,res)=>{

    try {

        const pass\_1 = req.body.new\_pass;

        const pass\_2 = req.body.com\_pass;

        const email = req.body.email;

        // const emailData = await webdata.findOne({email})

        //  if(emailData.email === email){

            //  res.send("email address already exist");

        //  }else

          if(pass\_1 === pass\_2){

             const userData = new webdata({

                 name:req.body.name,

                 email:req.body.email,

                 phone:req.body.phone,

                 new\_pass:req.body.new\_pass

             });

             const user = await userData.save();

            //  console.log(user)

              res.status(201).render("login");

         }else{

             res.send("password are matched...")

         }

    }catch(e){

        res.status(404).send(e)

    }

});

app.post("/weather",async(req,res)=>{

    try {

        const email = req.body.email;

        const pass = req.body.pass;

        const user = await webdata.findOne({email});

        const matchPass = await bcryptjs.compare(pass,user.new\_pass);

        if(matchPass){

             const token = await user.generateAuthToken();

             res.cookie("jwt",token,{

               expires:new Date(Date.now()+500000),

               htmlOnly:true,

           });

             res.render("weather")

        }else{

             res.send("invalid login details");

        }

     }catch(error){

        res.status(404).send(error );

    }

});

app.get("/forgetPass",(req,res)=>{

    res.render("forgetPass");

});

app.post("/userDetails",async(req,res)=>{

    try{

    const email = req.body.email;

    const number = req.body.number;

    const user = await webdata.findOne({email});

       if(user.phone==number){

           res.status(200).render("userDetails");

       }else{

           res.send("Pls Enter valid email or phone No...")

       }

   }catch(e){

    res.status(404).send(e)

   }

});

app.get("/home",(req,res) =>{

    res.render("index");

});

app.get("/about",(req,res) =>{

    res.render("about");

});

app.get("/weather",auth,(req,res) =>{

    res.render("weather");

});

app.get("/login",(req,res)=>{

    res.render("login");

});

app.get("/register",(req,res)=>{

    res.render("register");

});

app.post("/login",async(req,res)=>{

    try {

        const pass\_1 = req.body.new\_pass;

        const pass\_2 = req.body.com\_pass;

        const email = req.body.email;

        // const emailData = await webdata.findOne({email})

        //  if(emailData.email === email){

            //  res.send("email address already exist");

        //  }else

          if(pass\_1 === pass\_2){

             const userData = new webdata({

                 name:req.body.name,

                 email:req.body.email,

                 phone:req.body.phone,

                 new\_pass:req.body.new\_pass

             });

             const user = await userData.save();

            //  console.log(user)

              res.status(201).render("login");

         }else{

             res.send("password are matched...")

         }

    }catch(e){

        res.status(404).send(e)

    }

});

app.post("/weather",async(req,res)=>{

    try {

        const email = req.body.email;

        const pass = req.body.pass;

        const user = await webdata.findOne({email});

        const matchPass = await bcryptjs.compare(pass,user.new\_pass);

        if(matchPass){

             const token = await user.generateAuthToken();

             res.cookie("jwt",token,{

               expires:new Date(Date.now()+500000),

               htmlOnly:true,

           });

             res.render("weather")

        }else{

             res.send("invalid login details");

        }

     }catch(error){

        res.status(404).send(error );

    }

});

app.get("/forgetPass",(req,res)=>{

    res.render("forgetPass");

});

app.post("/userDetails",async(req,res)=>{

    try{

    const email = req.body.email;

    const number = req.body.number;

    const user = await webdata.findOne({email});

       if(user.phone==number){

           res.status(200).render("userDetails");

       }else{

           res.send("Pls Enter valid email or phone No...")

       }

   }catch(e){

    res.status(404).send(e)

   }

});

app.get("/home",(req,res) =>{

    res.render("index");

});

app.get("/about",(req,res) =>{

    res.render("about");

});

app.get("/weather",auth,(req,res) =>{

    res.render("weather");

});

app.get("/login",(req,res)=>{

    res.render("login");

});

app.get("/register",(req,res)=>{

    res.render("register");

});

app.post("/login",async(req,res)=>{

    try {

        const pass\_1 = req.body.new\_pass;

        const pass\_2 = req.body.com\_pass;

        const email = req.body.email;

        // const emailData = await webdata.findOne({email})

        //  if(emailData.email === email){

            //  res.send("email address already exist");

        //  }else

          if(pass\_1 === pass\_2){

             const userData = new webdata({

                 name:req.body.name,

                 email:req.body.email,

                 phone:req.body.phone,

                 new\_pass:req.body.new\_pass

             });

             const user = await userData.save();

            //  console.log(user)

              res.status(201).render("login");

         }else{

             res.send("password are matched...")

         }

    }catch(e){

        res.status(404).send(e)

    }

});

app.post("/weather",async(req,res)=>{

    try {

        const email = req.body.email;

        const pass = req.body.pass;

        const user = await webdata.findOne({email});

        const matchPass = await bcryptjs.compare(pass,user.new\_pass);

        if(matchPass){

             const token = await user.generateAuthToken();

             res.cookie("jwt",token,{

               expires:new Date(Date.now()+500000),

               htmlOnly:true,

           });

             res.render("weather")

        }else{

             res.send("invalid login details");

        }

     }catch(error){

        res.status(404).send(error );

    }

});

app.get("/forgetPass",(req,res)=>{

    res.render("forgetPass");

});

app.post("/userDetails",async(req,res)=>{

    try{

    const email = req.body.email;

    const number = req.body.number;

    const user = await webdata.findOne({email});

       if(user.phone==number){

           res.status(200).render("userDetails");

       }else{

           res.send("Pls Enter valid email or phone No...")

       }

   }catch(e){

    res.status(404).send(e)

   }

});

app.get("/home",(req,res) =>{

    res.render("index");

});

app.get("/about",(req,res) =>{

    res.render("about");

});

app.get("/weather",auth,(req,res) =>{

    res.render("weather");

});

app.get("/login",(req,res)=>{

    res.render("login");

});

app.get("/register",(req,res)=>{

    res.render("register");

});

app.post("/login",async(req,res)=>{

    try {

        const pass\_1 = req.body.new\_pass;

        const pass\_2 = req.body.com\_pass;

        const email = req.body.email;

        // const emailData = await webdata.findOne({email})

        //  if(emailData.email === email){

            //  res.send("email address already exist");

        //  }else

          if(pass\_1 === pass\_2){

             const userData = new webdata({

                 name:req.body.name,

                 email:req.body.email,

                 phone:req.body.phone,

                 new\_pass:req.body.new\_pass

             });

             const user = await userData.save();

            //  console.log(user)

              res.status(201).render("login");

         }else{

             res.send("password are matched...")

         }

    }catch(e){

        res.status(404).send(e)

    }

});

app.post("/weather",async(req,res)=>{

    try {

        const email = req.body.email;

        const pass = req.body.pass;

        const user = await webdata.findOne({email});

        const matchPass = await bcryptjs.compare(pass,user.new\_pass);

        if(matchPass){

             const token = await user.generateAuthToken();

             res.cookie("jwt",token,{

               expires:new Date(Date.now()+500000),

               htmlOnly:true,

           });

             res.render("weather")

        }else{

             res.send("invalid login details");

        }

     }catch(error){

        res.status(404).send(error );

    }

});

app.get("/forgetPass",(req,res)=>{

    res.render("forgetPass");

});

app.post("/userDetails",async(req,res)=>{

    try{

    const email = req.body.email;

    const number = req.body.number;

    const user = await webdata.findOne({email});

       if(user.phone==number){

           res.status(200).render("userDetails");

       }else{

           res.send("Pls Enter valid email or phone No...")

       }

   }catch(e){

    res.status(404).send(e)

   }

});

app.get("/home",(req,res) =>{

    res.render("index");

});

app.get("/about",(req,res) =>{

    res.render("about");

});

app.get("/weather",auth,(req,res) =>{

    res.render("weather");

});

app.get("/login",(req,res)=>{

    res.render("login");

});

app.get("/register",(req,res)=>{

    res.render("register");

});

app.post("/login",async(req,res)=>{

    try {

        const pass\_1 = req.body.new\_pass;

        const pass\_2 = req.body.com\_pass;

        const email = req.body.email;

        // const emailData = await webdata.findOne({email})

        //  if(emailData.email === email){

            //  res.send("email address already exist");

        //  }else

          if(pass\_1 === pass\_2){

             const userData = new webdata({

                 name:req.body.name,

                 email:req.body.email,

                 phone:req.body.phone,

                 new\_pass:req.body.new\_pass

             });

             const user = await userData.save();

            //  console.log(user)

              res.status(201).render("login");

         }else{

             res.send("password are matched...")

         }

    }catch(e){

        res.status(404).send(e)

    }

});

app.post("/weather",async(req,res)=>{

    try {

        const email = req.body.email;

        const pass = req.body.pass;

        const user = await webdata.findOne({email});

        const matchPass = await bcryptjs.compare(pass,user.new\_pass);

        if(matchPass){

             const token = await user.generateAuthToken();

             res.cookie("jwt",token,{

               expires:new Date(Date.now()+500000),

               htmlOnly:true,

           });

             res.render("weather")

        }else{

             res.send("invalid login details");

        }

     }catch(error){

        res.status(404).send(error );

    }

});

app.get("/forgetPass",(req,res)=>{

    res.render("forgetPass");

});

app.post("/userDetails",async(req,res)=>{

    try{

    const email = req.body.email;

    const number = req.body.number;

    const user = await webdata.findOne({email});

       if(user.phone==number){

           res.status(200).render("userDetails");

       }else{

           res.send("Pls Enter valid email or phone No...")

       }

   }catch(e){

    res.status(404).send(e)

   }

});

app.get("/home",(req,res) =>{

    res.render("index");

});

app.get("/about",(req,res) =>{

    res.render("about");

});

app.get("/weather",auth,(req,res) =>{

    res.render("weather");

});

app.get("/login",(req,res)=>{

    res.render("login");

});

app.get("/register",(req,res)=>{

    res.render("register");

});

app.post("/login",async(req,res)=>{

    try {

        const pass\_1 = req.body.new\_pass;

        const pass\_2 = req.body.com\_pass;

        const email = req.body.email;

        // const emailData = await webdata.findOne({email})

        //  if(emailData.email === email){

            //  res.send("email address already exist");

        //  }else

          if(pass\_1 === pass\_2){

             const userData = new webdata({

                 name:req.body.name,

                 email:req.body.email,

                 phone:req.body.phone,

                 new\_pass:req.body.new\_pass

             });

             const user = await userData.save();

            //  console.log(user)

              res.status(201).render("login");

         }else{

             res.send("password are matched...")

         }

    }catch(e){

        res.status(404).send(e)

    }

});

app.post("/weather",async(req,res)=>{

    try {

        const email = req.body.email;

        const pass = req.body.pass;

        const user = await webdata.findOne({email});

        const matchPass = await bcryptjs.compare(pass,user.new\_pass);

        if(matchPass){

             const token = await user.generateAuthToken();

             res.cookie("jwt",token,{

               expires:new Date(Date.now()+500000),

               htmlOnly:true,

           });

             res.render("weather")

        }else{

             res.send("invalid login details");

        }

     }catch(error){

        res.status(404).send(error );

    }

});

app.get("/forgetPass",(req,res)=>{

    res.render("forgetPass");

});

app.post("/userDetails",async(req,res)=>{

    try{

    const email = req.body.email;

    const number = req.body.number;

    const user = await webdata.findOne({email});

       if(user.phone==number){

           res.status(200).render("userDetails");

       }else{

           res.send("Pls Enter valid email or phone No...")

       }

   }catch(e){

    res.status(404).send(e)

   }

});

app.get("/home",(req,res) =>{

    res.render("index");

});

app.get("/about",(req,res) =>{

    res.render("about");

});

app.get("/weather",auth,(req,res) =>{

    res.render("weather");

});

app.get("/login",(req,res)=>{

    res.render("login");

});

app.get("/register",(req,res)=>{

    res.render("register");

});

app.post("/login",async(req,res)=>{

    try {

        const pass\_1 = req.body.new\_pass;

        const pass\_2 = req.body.com\_pass;

        const email = req.body.email;

        // const emailData = await webdata.findOne({email})

        //  if(emailData.email === email){

            //  res.send("email address already exist");

        //  }else

          if(pass\_1 === pass\_2){

             const userData = new webdata({

                 name:req.body.name,

                 email:req.body.email,

                 phone:req.body.phone,

                 new\_pass:req.body.new\_pass

             });

             const user = await userData.save();

            //  console.log(user)

              res.status(201).render("login");

         }else{

             res.send("password are matched...")

         }

    }catch(e){

        res.status(404).send(e)

    }

});

app.post("/weather",async(req,res)=>{

    try {

        const email = req.body.email;

        const pass = req.body.pass;

        const user = await webdata.findOne({email});

        const matchPass = await bcryptjs.compare(pass,user.new\_pass);

        if(matchPass){

             const token = await user.generateAuthToken();

             res.cookie("jwt",token,{

               expires:new Date(Date.now()+500000),

               htmlOnly:true,

           });

             res.render("weather")

        }else{

             res.send("invalid login details");

        }

     }catch(error){

        res.status(404).send(error );

    }

});

app.get("/forgetPass",(req,res)=>{

    res.render("forgetPass");

});

app.post("/userDetails",async(req,res)=>{

    try{

    const email = req.body.email;

    const number = req.body.number;

    const user = await webdata.findOne({email});

       if(user.phone==number){

           res.status(200).render("userDetails");

       }else{

           res.send("Pls Enter valid email or phone No...")

       }

   }catch(e){

    res.status(404).send(e)

   }

});

app.get("/home",(req,res) =>{

    res.render("index");

});

app.get("/about",(req,res) =>{

    res.render("about");

});

app.get("/weather",auth,(req,res) =>{

    res.render("weather");

});

app.get("/login",(req,res)=>{

    res.render("login");

});

app.get("/register",(req,res)=>{

    res.render("register");

});

app.post("/login",async(req,res)=>{

    try {

        const pass\_1 = req.body.new\_pass;

        const pass\_2 = req.body.com\_pass;

        const email = req.body.email;

        // const emailData = await webdata.findOne({email})

        //  if(emailData.email === email){

            //  res.send("email address already exist");

        //  }else

          if(pass\_1 === pass\_2){

             const userData = new webdata({

                 name:req.body.name,

                 email:req.body.email,

                 phone:req.body.phone,

                 new\_pass:req.body.new\_pass

             });

             const user = await userData.save();

            //  console.log(user)

              res.status(201).render("login");

         }else{

             res.send("password are matched...")

         }

    }catch(e){

        res.status(404).send(e)

    }

});

app.post("/weather",async(req,res)=>{

    try {

        const email = req.body.email;

        const pass = req.body.pass;

        const user = await webdata.findOne({email});

        const matchPass = await bcryptjs.compare(pass,user.new\_pass);

        if(matchPass){

             const token = await user.generateAuthToken();

             res.cookie("jwt",token,{

               expires:new Date(Date.now()+500000),

               htmlOnly:true,

           });

             res.render("weather")

        }else{

             res.send("invalid login details");

        }

     }catch(error){

        res.status(404).send(error );

    }

});

app.get("/forgetPass",(req,res)=>{

    res.render("forgetPass");

});

app.post("/userDetails",async(req,res)=>{

    try{

    const email = req.body.email;

    const number = req.body.number;

    const user = await webdata.findOne({email});

       if(user.phone==number){

           res.status(200).render("userDetails");

       }else{

           res.send("Pls Enter valid email or phone No...")

       }

   }catch(e){

    res.status(404).send(e)

   }

});

app.get("/home",(req,res) =>{

    res.render("index");

});

app.get("/about",(req,res) =>{

    res.render("about");

});

app.get("/weather",auth,(req,res) =>{

    res.render("weather");

});

app.get("/login",(req,res)=>{

    res.render("login");

});

app.get("/register",(req,res)=>{

    res.render("register");

});

app.post("/login",async(req,res)=>{

    try {

        const pass\_1 = req.body.new\_pass;

        const pass\_2 = req.body.com\_pass;

        const email = req.body.email;

        // const emailData = await webdata.findOne({email})

        //  if(emailData.email === email){

            //  res.send("email address already exist");

        //  }else

          if(pass\_1 === pass\_2){

             const userData = new webdata({

                 name:req.body.name,

                 email:req.body.email,

                 phone:req.body.phone,

                 new\_pass:req.body.new\_pass

             });

             const user = await userData.save();

            //  console.log(user)

              res.status(201).render("login");

         }else{

             res.send("password are matched...")

         }

    }catch(e){

        res.status(404).send(e)

    }

});

app.post("/weather",async(req,res)=>{

    try {

        const email = req.body.email;

        const pass = req.body.pass;

        const user = await webdata.findOne({email});

        const matchPass = await bcryptjs.compare(pass,user.new\_pass);

        if(matchPass){

             const token = await user.generateAuthToken();

             res.cookie("jwt",token,{

               expires:new Date(Date.now()+500000),

               htmlOnly:true,

           });

             res.render("weather")

        }else{

             res.send("invalid login details");

        }

     }catch(error){

        res.status(404).send(error );

    }

});

app.get("/forgetPass",(req,res)=>{

    res.render("forgetPass");

});

app.post("/userDetails",async(req,res)=>{

    try{

    const email = req.body.email;

    const number = req.body.number;

    const user = await webdata.findOne({email});

       if(user.phone==number){

           res.status(200).render("userDetails");

       }else{

           res.send("Pls Enter valid email or phone No...")

       }

   }catch(e){

    res.status(404).send(e)

}

**CHAPTER 7**

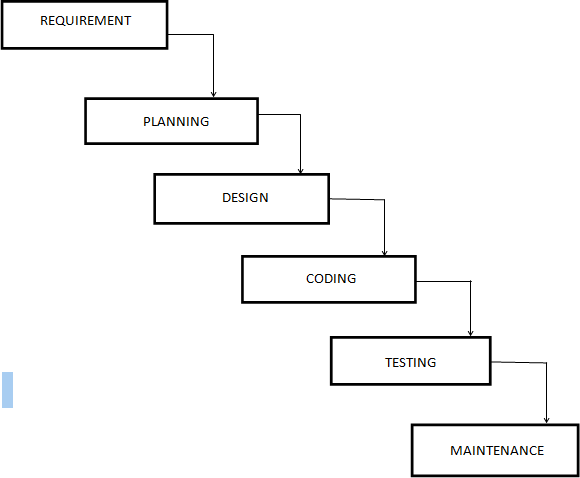
**TESTING OF THE PROJECT**

**7.1 Testing**

Testing is a critical phase in the software development life cycle that ensures that the software is reliable, efficient, and performs as expected. It is a systematic process of identifying and reporting defects or errors in a software product or application. The main purpose of testing is to identify and fix defects before the product or application is released to end-users.

Testing can be classified into two types: manual testing and automated testing. Manual testing is performed by a human tester who manually executes test cases and verifies the output. Automated testing, on the other hand, is performed by using tools and software to execute test cases and compare actual results with expected results. There are various types of testing that are performed during the software development life cycle.

Some of the commonly used testing types are unit testing, integration testing, system testing, acceptance testing, and regression testing. Each type of testing is designed to test specific aspects of the software application.

**7.2 Testing Architecture**

***Figure 7.1 Software Development Life cycle***

**7. 3 Types of Testing**

**7.3.1 Unit Testing**

* Unit testing is a crucial step in the software development process where developers write test cases to verify the functionality of individual units or modules of the source code. The main objective of unit testing is to ensure that each component of the software performs as expected, adheres to the design specifications, and meets the overall quality standards of the project.
* During unit testing, developers use different sets of test data than the ones used by the quality assurance team. This helps them to identify potential bugs and issues at an early stage, enabling them to rectify them before handing over the code to the testing team. By isolating each part of the program, developers 80 can ensure that the individual components work as intended and do not adversely affect other parts of the code.
* The process of unit testing typically involves writing test cases that cover all possible scenarios and edge cases of a particular unit. Developers use specialized tools and frameworks to automate the testing process and ensure that the tests run smoothly and efficiently. They also perform code reviews and code analysis to identify any potential issues that may affect the performance and functionality of the software.

**7.3.2 Integration Testing**

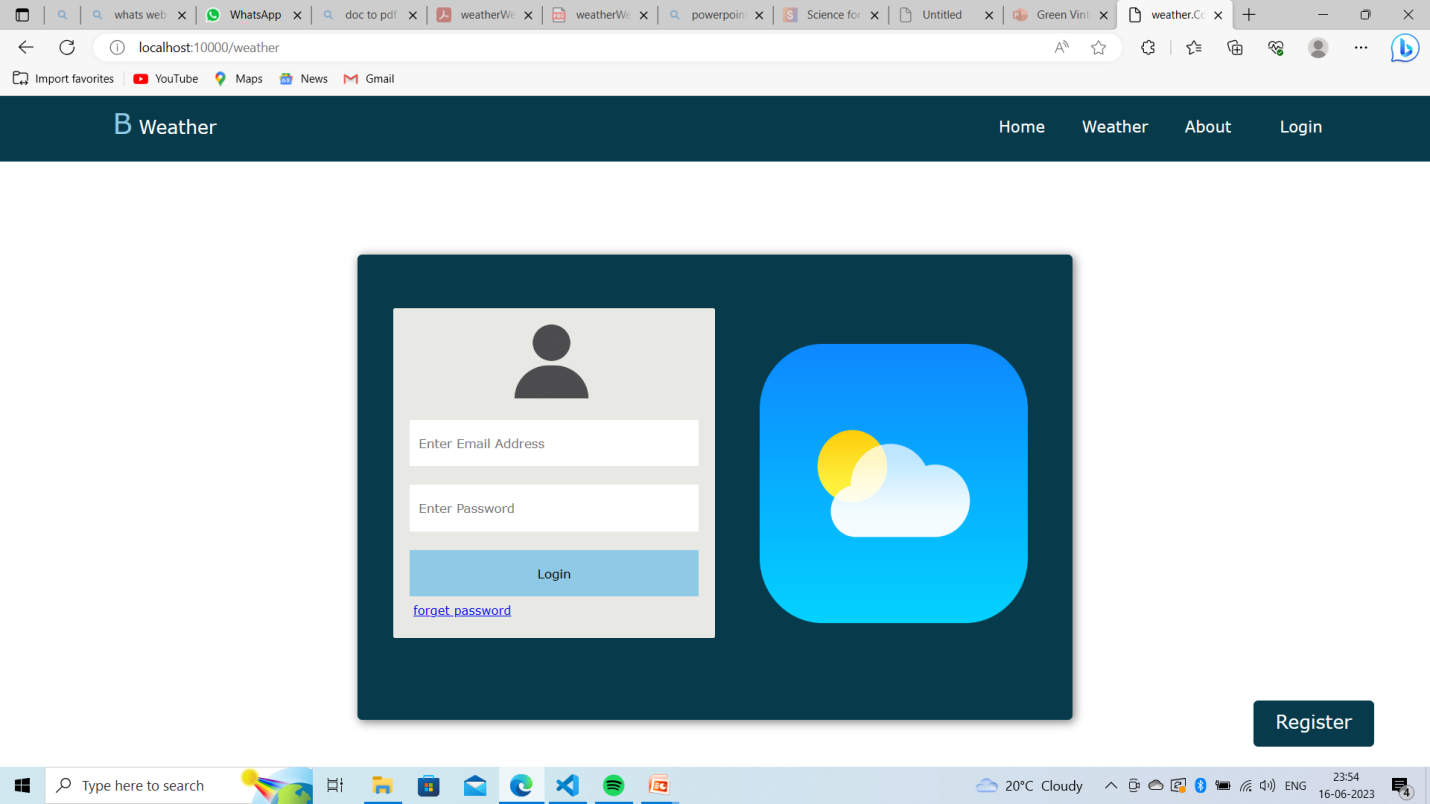
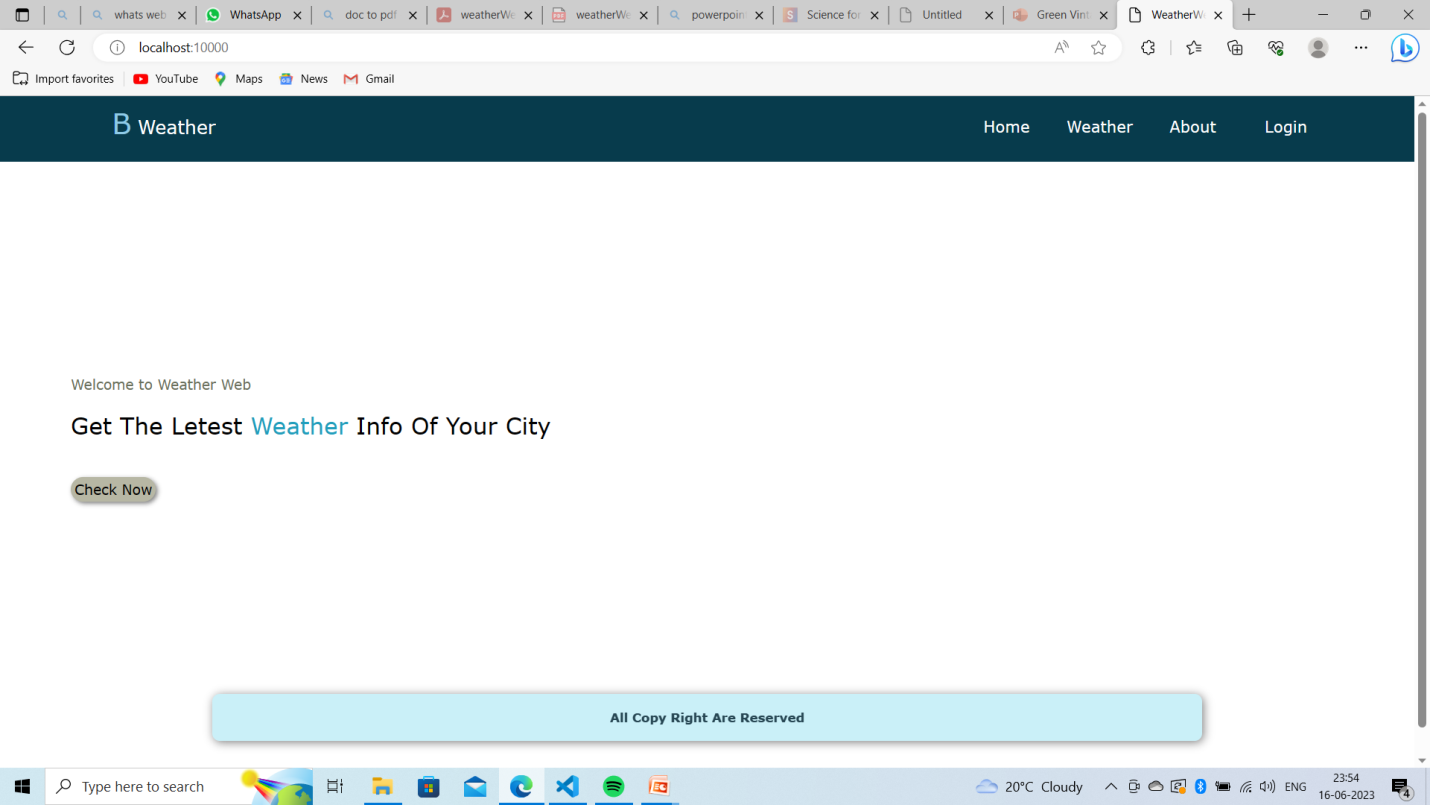
* Integration testing is an important aspect of software development that aims to verify the interaction between different modules of the software system. It is a type of testing that is performed after unit testing and before system testing. The purpose of integration testing is to ensure that all the individual modules of the software system work together as intended and that there are no integration issues or conflicts that could affect the overall functionality of the system.
* During integration testing, developers test the integrated modules of the software system to ensure that they function correctly and meet the system requirements. The testing team verifies that the modules interact seamlessly with each other and that there are no compatibility issues. Integration testing can be performed in different ways, such as top-down testing, bottom-up testing, and incremental testing.
* In top-down testing, the testing team starts testing from the highest level of the software system and gradually moves down to the lower levels. This type of testing is useful for detecting any issues in the higher-level modules of the system. In contrast, bottom-up testing starts from the lower-level modules and gradually moves up to the higher levels. This testing method is useful for detecting issues in the lower-level modules and verifying their integration with the higher-level modules.

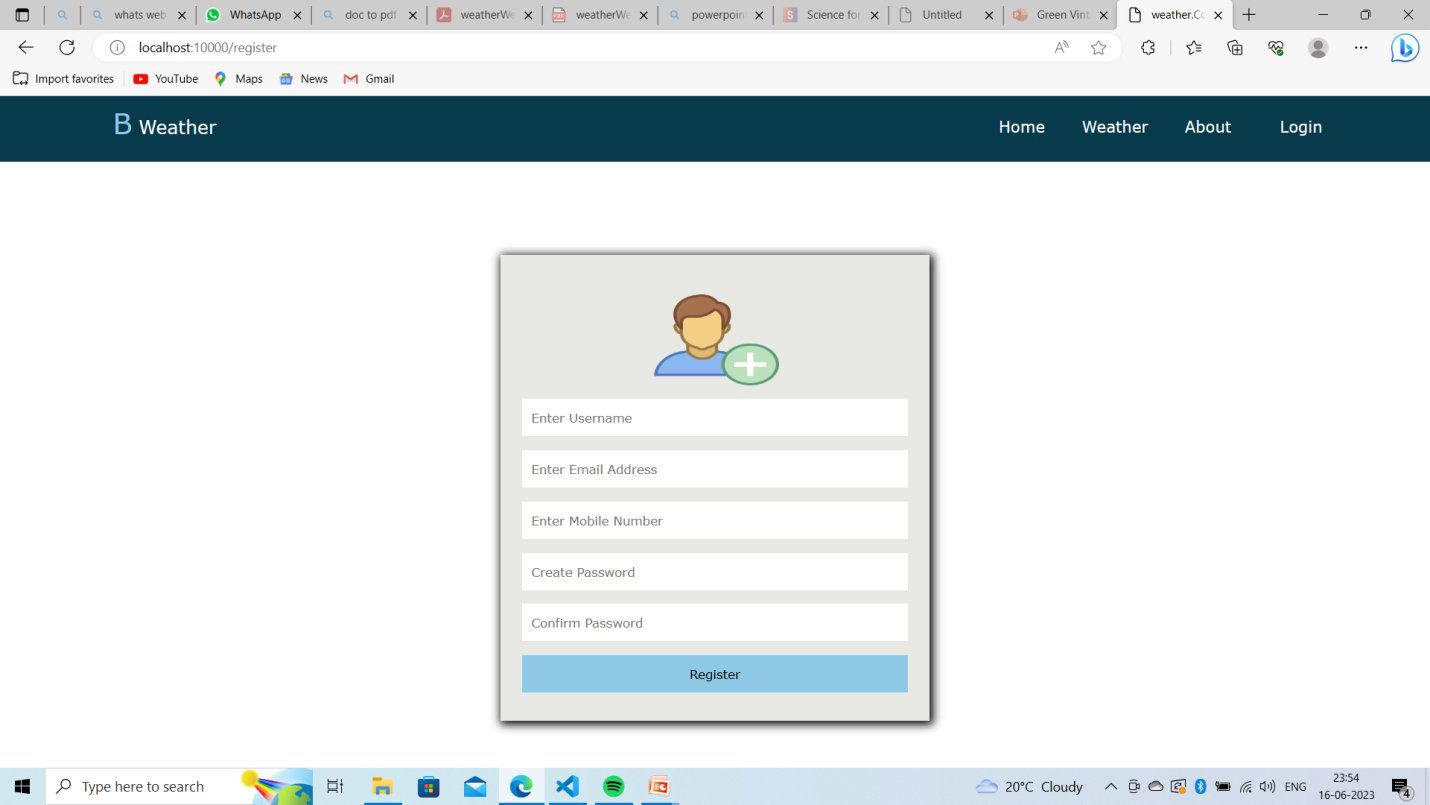
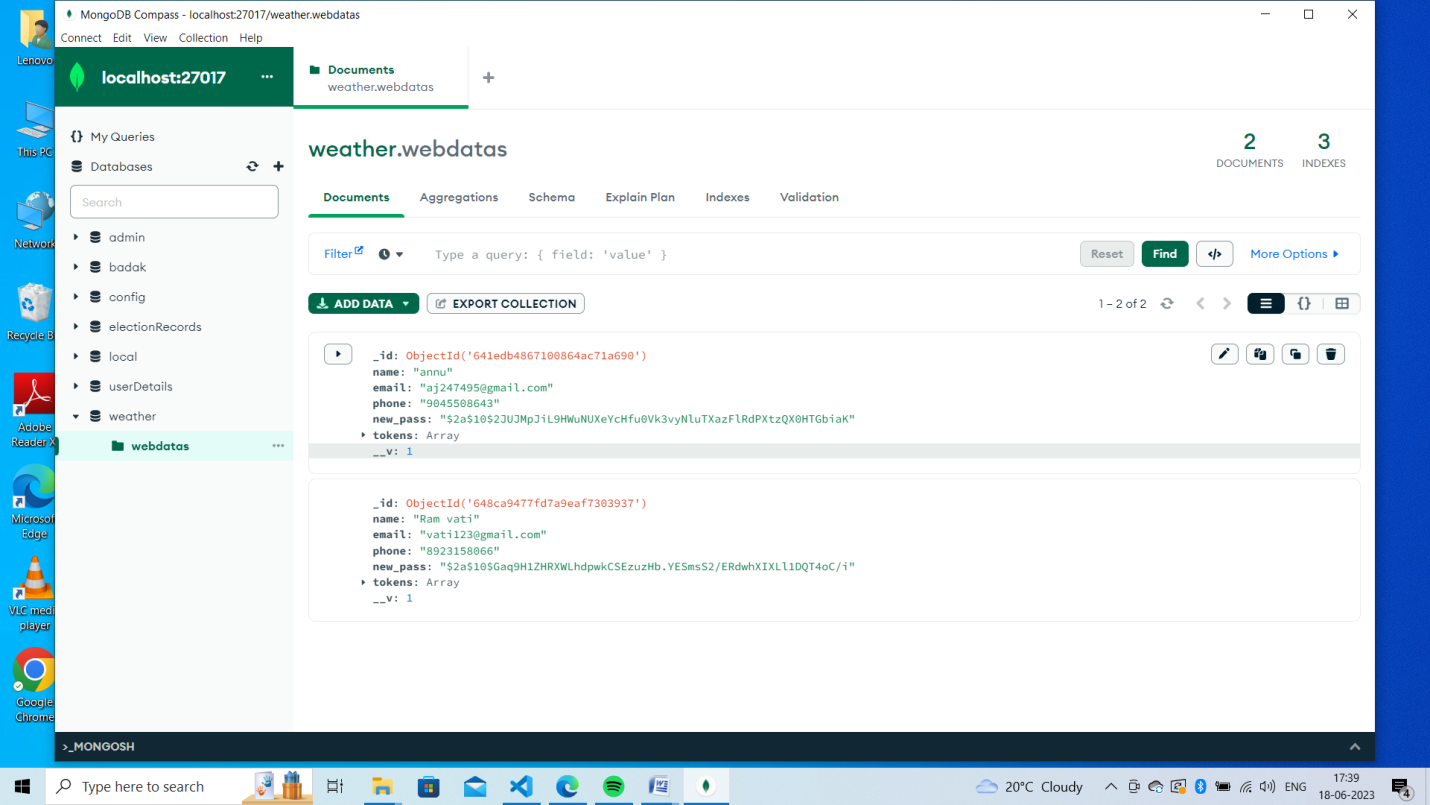
**7.3.3 System Testing**

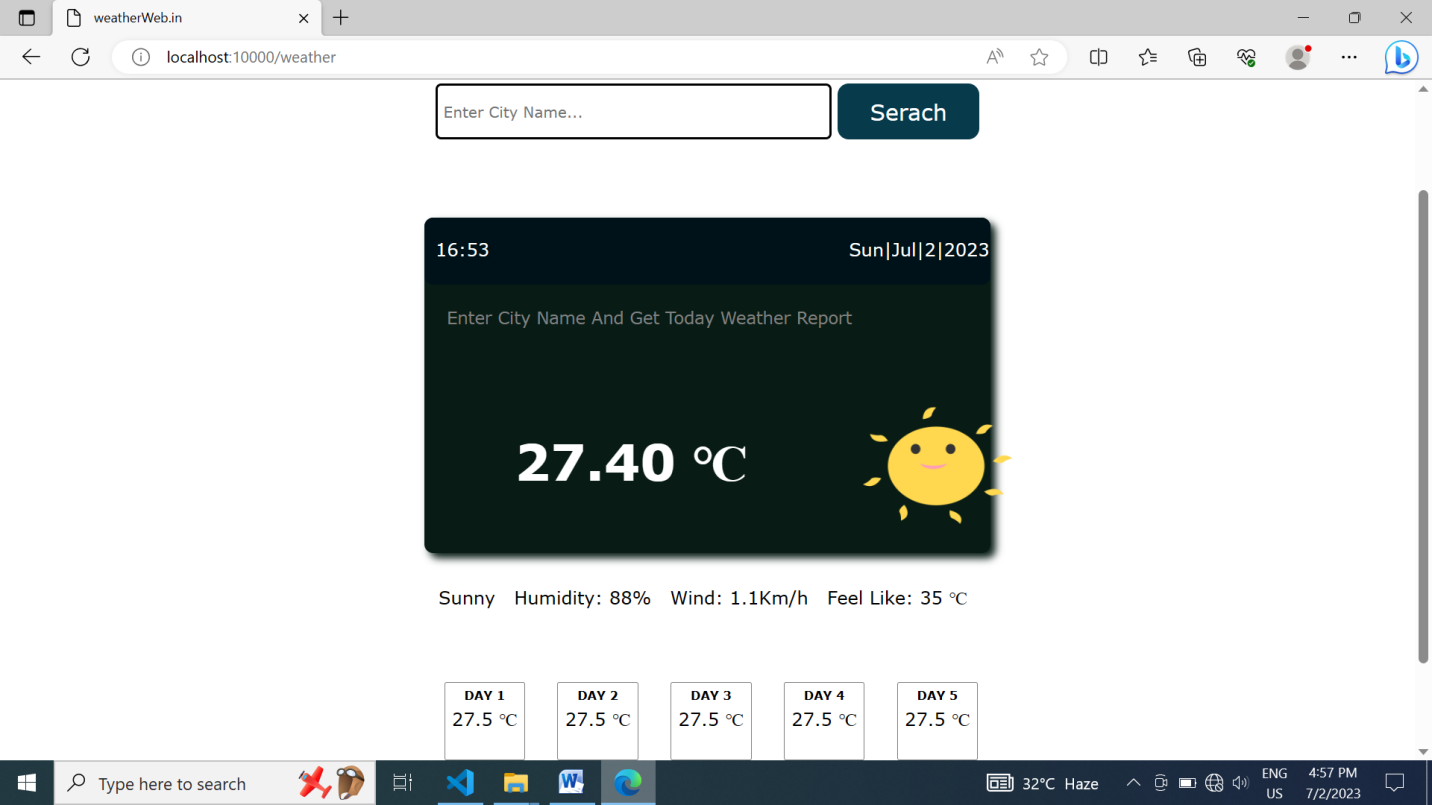
* System testing is a crucial phase in the software development life cycle, which follows unit and integration testing. In this phase, the entire software program is tested in conjunction with the expected hardware, network, and other associated parts that work together. The objective of system testing is to ensure that the software program is performing as expected in its intended environment.
* During system testing, both functional and non functional requirements of the system are tested. Functional testing ensures that the software program performs its intended functions, while non functional testing checks the software's performance, reliability, scalability, and security. The goal of system testing is to identify defects and bugs that could affect the system's performance and functionality.
* In system testing, the software program is tested in a simulated or real environment that mimics the end-users' environment. This testing environment includes the necessary hardware, software, and network configurations required for the software to operate correctly. The tests are conducted using various techniques, such as black-box testing, white-box testing, regression testing, and performance testing.

**CHAPTER 8**

**SNAPSHOTS**

****





**CHAPTER 9**

**CONCLUSION & FUTURE SCOPE**

**9.1 Conclusion**

The proposed work is intended to find the solution for accurate weather data modeling and prediction using the historical data. Therefore the data mining technique is studied for developing such kind of data model. The data mining techniques analyse the data of some pre-defined pattern and extract the significant on the data. Using the extracted patterns from the data the model takes training and prepared for classifying or predicting the similar patterns of associated class labels. The classification of the data supported by supervised technique of data mining. The key issue in weather prediction is to accomplish the relationship among the class labels and the attributes which are used for predicting the weather conditions. Therefore using the available accurate techniques a new data model is developed for weather forecasting. The proposed weather forecasting data model utilizes the kmeans unsupervised learning technique for performing the clustering on the entire training dataset. This clustering is performed for finding the pattern level pattern similarity among two instance data. The total number of 12 clusters is developed; these clusters are providing the observations of the data and their weather conditions. Using the extracted observations and available class labels the data is reorganized in terms of observation matrix and the transition matrix. These two different matrixes are provided into next phase namely Hidden Markov Model and the training are performed. The trained data model is used for prediction or the pattern recognition work. For making the prediction, using prepared trained data model. The system needs to provide the current weather conditions on the basis of that the new upcoming pattern of data is predicted.

**9.2 Future Scope**

The key aim for improving the classification and prediction performance for the traditional weather prediction model is designed and developed in this work. The implemented technique is efficient and accurate for weather prediction but some limitations of the model is also observed thus in near future need to be review before use of the proposed technique. Some of the key extensions of the works are as follows:

* Need to improve the training time: the training time of the proposed data model is increases with the amount of data to be trained.
* Need to be increasing the training samples for collecting the training data: as the performance of prediction is increases with the amount of data thus huge data can solve the issues of incorrect classification or prediction.

**REFRENCES**

* [***https://github.com/android***](https://github.com/android)
* [***https://mongodb.com***](https://mongodb.com)
* [***https://npmjs.com***](https://npmjs.com)
* [***https://mogoose.com***](https://mogoose.com)
* [***https://w3schools.com***](https://w3schools.com)
* [***https://acuweather.com***](https://acuweather.com)
* [***https://scribd.com/documentation***](https://scribd.com/documentation)

[1] A. Ramos-Soto, A. Bugar´ın, S. Barro, and J. Taboada, “Linguistic Descriptions for Automatic Generation of Textual Short-Term Weather Forecasts on Real Prediction Data”, 1063-6706 (c) 2013 IEEE’

[2] Data Mining: What is Data Mining? [http://www.anderson.ucla.edu/faculty/jason. frand/teacher/technologies/palace/datamining.htm](http://www.anderson.ucla.edu/faculty/jason.%20frand/teacher/technologies/palace/datamining.htm)

[3] Data Mining - Applications & Trends, http://www.tutorialspoint.com/data\_mining/dm\_applicat ions\_trends.htm

[4] Mahak Chowdhary, Shrutika Suri and Mansi Bhutani, “Comparative Study of Intrusion Detection System”, 2014, IJCSE All Rights Reserved, Volume-2, Issue-4

[5] Mrs. Pradnya Muley, Dr. Anniruddha Joshi, “Application of Data Mining Techniques for Customer Segmentation in Real Time Business Intelligence”, International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163, Issue 4, Volume 2 (April 2015)

[6] Ghazaleh Khodabandelou, Charlotte Hug, Rebecca Deneckere, Camille Salinesi, “Supervised vs. Unsupervised Learning for Intentional Process Model Discovery”, Business Process Modeling, Development, and Support (BPMDS), Jun 2014, Thessalonique, Greece. pp.1-15, 2014

[7] Importance of Predictive Analytics in Business, http://www.orchestrate.com/blog/importance-ofpredictive-analytics-in-business.

[8] David A. Dickey, N. Carolina State U., Raleigh, NC, “Introduction to Predictive Modeling with Examples”, Statistics and Data Analysis, SAS Global Forum 2012

[9] Hand, Manilla, & Smyth, “Descriptive Modeling”, http://www.stat.columbia.edu/~madigan/DM08/descript ive.ppt.pdf [10] K.Jayavani, “STATISTICAL CLASSIFICATION IN MACHINE INTELLEGENT”, ISRJournals and Publications, Volume: 1 Issue: 1 18-Jul-2014, I