**Weather Forecast Website**

**A Project Report**

***Submitted by:***

**Sagar Rana (19EBKCS099)**

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

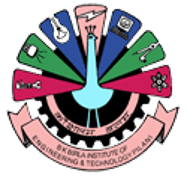
***of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

at



**B.K. BIRLA INSTITUTE OF ENGINEERING & TECHNOLOGY**

**PILANI, RAJASTHAN (INDIA) - 333031**

**(AFFILIATED TO RAJASTHAN TECHNICAL UNIVERSITY, KOTA, RAJASTHAN (INDIA))**

**MAY 2023**

**DECLARATION**

I hereby declare that the project entitled “weather forecast website” submitted for the B. Tech. (CSE) degree is my original work, and the project has not formed the basis for the award of any other degree, or any other similar title.

Sagar Rana

**Signature of the Student**

**Place:**

**Date:**

**CERTIFICATE**

This is to certify that the project titled “weather forecast website” is the Bonafide work carried out by Sagar Rana(19EBKCS099) a student of BTech (CSE) of B. K. Birla Institute of Engineering & Technology, Pilani affiliated to Rajasthan Technical University, Kota, Rajasthan (India) during the academic year 2019-2023, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Computer Science and Engineering) and that the project has not formed the basis for the award previously of any other degree, or any other similar title.

**Signature of the Guide**

**Place:**

**Date:**

**ACKNOWLEDGEMENTS**

I would like to express my deepest gratitude to my guide, Mr. Nachiket Sainis for his valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to us for completing this research work.

**ABSTRACT**

Weather forecasting is the application of science and technology to predict the conditions of the atmosphere for a given location and time. People have attempted to predict the weather informally for millenia and formally since the 19th century.

Weather forecasts are made by collecting quantitative data about the current state of the atmosphere, land, and ocean and using meteorology to project how the atmosphere will change at a given place.

Once calculated manually based mainly upon changes in barometric pressure, current weather conditions, and sky conditions or cloud cover, weather forecasting now relies on computer based model that take many atmospheric factors into account. Human input is still required to pick the best possible model to base the forecast upon, which involves pattern recognition skills, teleconnections, knowledge of model performance, and knowledge of model biases.

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. Hence, forecasts become less accurate as the difference between current time and the time for which the forecast is being made (the *range* of the forecast) increases. The use of ensembles and model consensus helps narrow the error and provide confidence in the forecast.

1.1 OBJECTIVE OF THE SYSTEM

This project will serve the following objectives:-

1. Provides the user with an easy and friendly interface

2. Provides the user with the temperature of a particular region

3. It will also show humidity, wind speed and cloud

1. INTRODUCTION

1.1.Objective of the System

1.2.Jusiﬁcation and need for the system

1.3.Advantage of the system

1.4 Previous work or related systems, how they

are used.

2. REQUIREMENT ANALYSIS

2.1 Analysis Study

2.2 User Requirements

2.3 Final Requirements

3. DESIGN OF THE SYSTEM

3.1 Hardware, Software requirements

3.2 System requirements

3.3 Design Requirement

3.4 DFDs

4. IMPLEMENTATION & CODING

4.1 Operating System

4.2. Languages

4.3 Coding

5.[Testing](#_heading=h.1jlao46)

5.1Software Testing Strategies . . . . . . . . . . . . . . . . . . .

[5.2](#_heading=h.xvir7l)Testing The Application . . . . . . . . . . . . . . . . . . . . . .

6. [Conclusion, and Future Work](#_heading=h.2250f4o)

[6.2Conclusion of the report](#_heading=h.4h042r0) . . . . . . . . . . . . . . .

**CHAPTER 1**

INTRODUCTION

Weather forecasting is the application of science and technology to predict the conditions of the atmosphere for a given location and time. People have attempted to predict the weather informally for millenia and formally since the 19th century.

Weather forecasts are made by collecting quantitative data about the current state of the atmosphere, land, and ocean and using meteorology to project how the atmosphere will change at a given place.

Once calculated manually based mainly upon changes in barometric pressure, current weather conditions, and sky conditions or cloud cover, weather forecasting now relies on computer based model that take many atmospheric factors into account. Human input is still required to pick the best possible model to base the forecast upon, which involves pattern recognition skills, teleconnections, knowledge of model performance, and knowledge of model biases.

1.1 OBJECTIVE OF THE SYSTEM

This project will serve the following objectives:-

1. Provides the user with an easy and friendly interface

2. Provides the user with the temperature of a particular region

3. It will also show humidity, wind speed and cloud

1.2 JUSTIFICATION AND NEED FOR THE SYSTEM

Weather is something everybody deals with, and accurate data about it like what is coming can help users to make informed decisions. With weather apps for iOS and Android, people can exactly know when to expect a change in the weather conditions. Weather apps can give urgent alerts too.

Undoubtedly, weather forecasting has come a long way, helping people to know about weather conditions. So, if you are in an area where weather frequently changes from sunny to torrential rain in a matter of minutes, thenwhat is the easiest way to make sure to be prepared? A suitable answer is a weather application.

1.3 Advantages of the system

1. Real-Time Data

One of the biggest advantages of weather monitoring systems and also the reason why people have been going in for weather stations is because of the ability to get their information in real-time.

2. Accurate Local Forecast

In reality, the meteorological department may be located far from your home and weather forecasts are made for regions, not a specific area. That’s a reason why in these instances, the weather predictions that they give are not always the most accurate.

3. Ease Of Use

Ease to use is definitely a big advantage of the weather monitoring system. Weather stations like all other weather devices are designed to be efficient and straightforward, therefore, everyone can use them.It is so convenient and comfortable for users to get the most accurate information in the simplest way possible.

1.4 Previous work or related systems; how they are used.

Before we begin a new system it is important to study the system that will be improved or replaced (if there is one). We need to analyze how this system uses hardware, software, network and people resources to convert data resources, such as transaction data, into information products.

Following are the problems associated with the previous project which led to the creation of the proposed project:-

1. Not user-friendly: The existing system is not user-friendly because the information like humidity cloud and wind etc are not in one place.

2. Not a good UI: The user interface of the previous systems are not that good.

**CHAPTER 2**

REQUIREMENT ANALYSIS

Functional Requirements

Functional requirements are the requirements that describe the functionalities of the system elements. It may involve functional user requirements or functional system requirements.

For example:

1.The operator shall be able to input the region to the system to view the desired weather parameters.

2.The system shall provide the following weather parameters: temperature, pressure, wind speed ,date / time and humidity.

2.1 ANALYSIS STUDY

1. Lower Installation Charges:

We neither require any high-configuration systems for the smooth running of the server program nor do we require any high-configuration systems for the smooth running of a client program. This application is designed with ease to support any ordinary system having an internet connection.

2. Secured and Reliable: The reliability of the system is to make sure the website does not go offline

3. Availability The availability of the system is that the website will be active on the Internet and people will be able to browse it.

2.2 Feasibility Study

All projects are feasible if they have unlimited resources and infinite time. But the development of software is plagued by the scarcity of resources and difficult delivery rates. It is necessary and prudent to evaluate the feasibility of a project at the earliest possible time. The three considerations are involved in

the feasibility analysis.

2.3 Technical Feasibility

Technical feasibility centres on the existing mobile system (hardware, software…etc) and to what extent it can support the proposed addition if the budget is a serious constraint, then the project is judged not feasible. The

technical feasibilities are an important role in our project because here we’re using HTML,CSS and JavaScript . It requires Visual Studio Code(software) to develop this application. A easily available software and easy to use.

2.4 Economical Feasibility

This procedure is to determine the benefits and savings that are expected from a candidate system and compare them with cost. If the benefits outweigh the cost then the decision is made to design and implement the system. Otherwise, further justification or alterations in proposed systems have to be made if it is having a chance of being approved. This is an ongoing effort that improves any feasibility costs spent on this project because here we’re using open-source environments.

2.5 Operational Feasibility

People are inherently resistant to change and mobiles have been known to facilitate change. There is no need of technical background is required to work on the application. All the information needed can be seen with just one click.

2.2 USER REQUIREMENTS

The system specifications that a user may want are as follows:

1. It should be easy to understand

2. Must be interactive

3. Should provide a good user interface

4. Security should be maintained

2.3 Final Requirements

**User Oriented:** A system should be more user friendly not from the technical point of view.

**Better GUI:** All the elements used in the system should be interactive in nature so that its look and feel are not so boring that the user could get bored while using it.

**Reliability:** The system should be reliable and fast in processing

Data security: Access to the organizational data is not to be granted to any unknown person who is not a part of the transaction.

**Confidentiality:** Whatever the user is providing to the organization, the user has the full rights to modify it and it could be not be accessed/modified without the user's permission

**Better Management of information:** All the information should be managed so that is the flow of the information is to be in the right track.

**Presentation:** The content that is to be presented to the user is to be presented in such a way that is self-explanatory to the user and he/she is satisfied with the data.

**Chapter 3**

DESIGN OF THE SYSTEM

3.1 Software requirements

| Platform | Platform Independent |
| --- | --- |
| The Operating System | Window 7 |
| Framework | Bootstrap |
| Front-End Tool | Google Chrome |
| API | OpenWeatherMap |

3.1 Hardware Requirements

| Processor | Intel Pentium IV 2.9 GHz Other |
| --- | --- |
| RAM | Minimum 4 GB |
| Graphics | Integrated graphics card |
| Hard Disk | Minimum 500 GB |

3.2 System Requirements

To know the detailed system requirements an SRS has to be prepared. Software requirement specification abbreviated as SRS is a means of translating the idea of files into a formal document. The main features of SRS include:

•Establishing the basis for an agreement between the client and the developer.

•Producing a reference for validation of the final product. SRS assist clients in determining if the

software meets their requirements.

Mainly there are six requirements which an SRS must satisfy.

(a) It should specify the external behaviour.

(b) It should specify the constraints.

(c) It should be easy to change.

(d) It should be a reference tool.

(e) It should record throughout the lifecycle.

(f) It should have the capacity to expect an undesired event.

**Functional Requirements**

Functional requirements are the requirements that describe the functionalities of the system elements. It may involve functional user requirements or functional system requirements.

For example:

1. The operator shall be able to input the region to the system to view the desired weather parameters.

2. The system shall provide the following weather parameters: temperature, pressure, wind speed & direction, rainfall, and humidity.\

3.3 Design Requirements

The main objectives of input design are:

(a) Controlling the amount of input.

(b) Keeping the process simple.

(c) The best thing in the input design is to achieve all the objectives mentioned in the simplest manner possible.

The main objectives of output design are:

(a) Identifying the specific outputs.

The primary goal of the system analysis is to improve the efficiency of the existing system. For that the study of specification of the requirements is very essential. For the development of the new system, a preliminary survey of the existing system will be conducted. Investigation done whether the upgradation

of the system into an application program could solve the problems and eradicate the inefficiency of the existing system.

**3.5 DATA FLOW DIAGRAM (DFD)**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. Often they are a preliminary step used to create an overview of

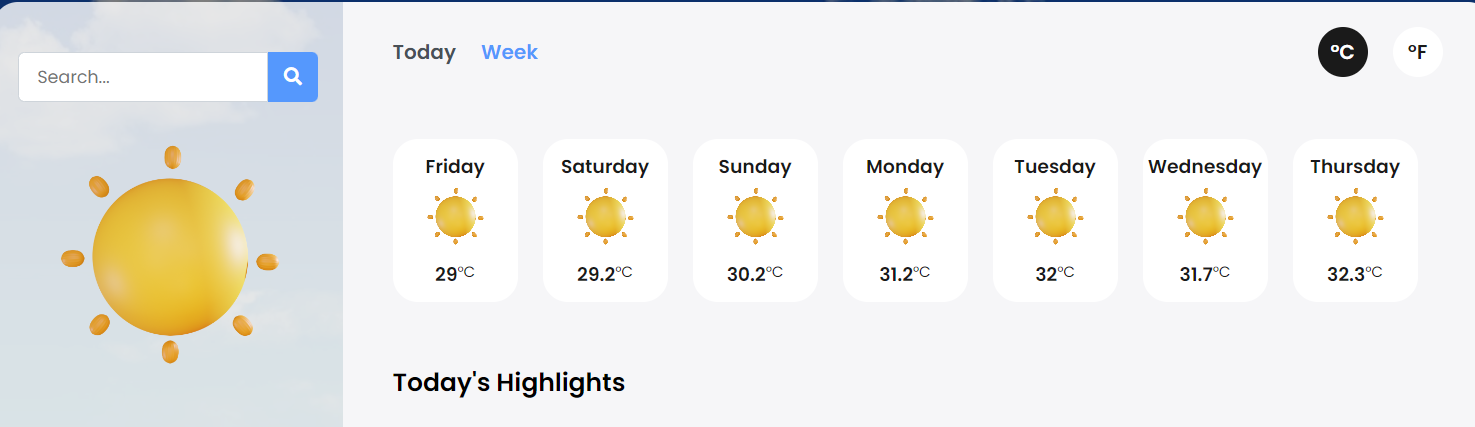
the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

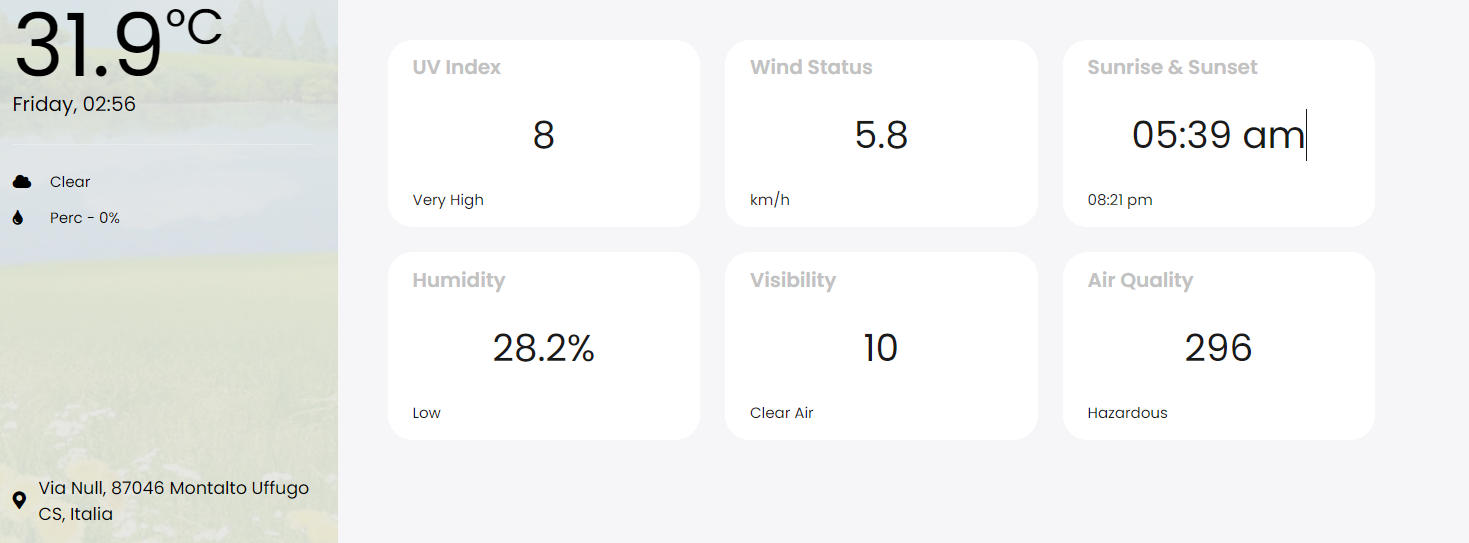
A DFD shows what kinds of information will be input into and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the

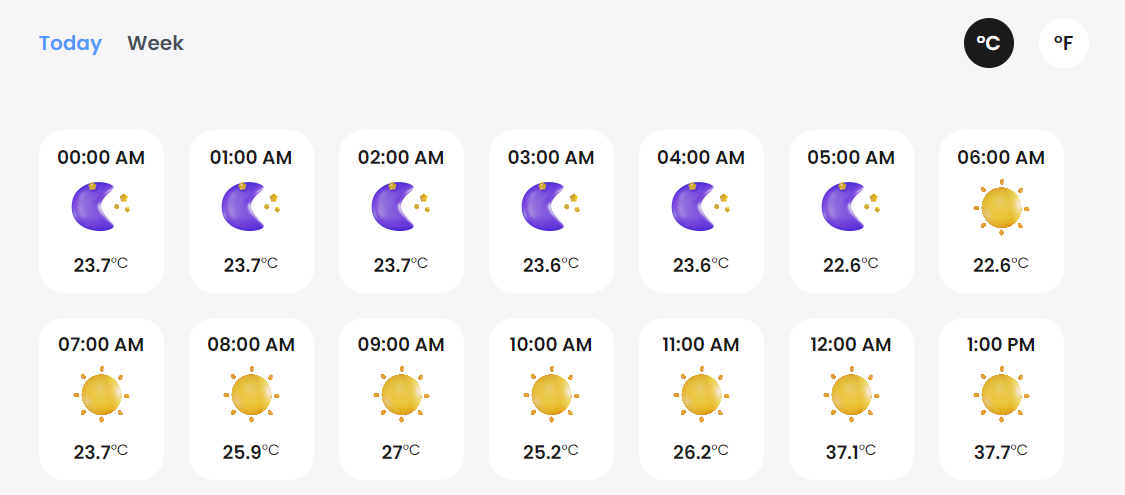
timing of processes, or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart

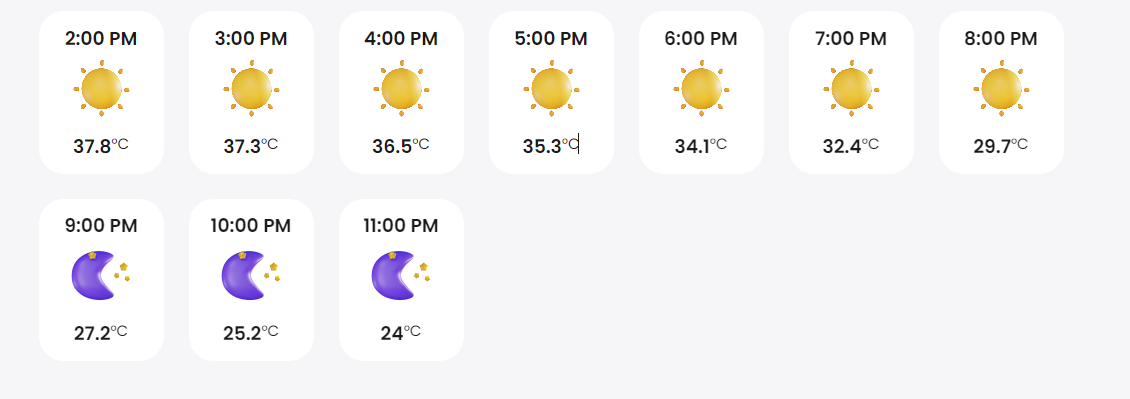
**CHAPTER 4**

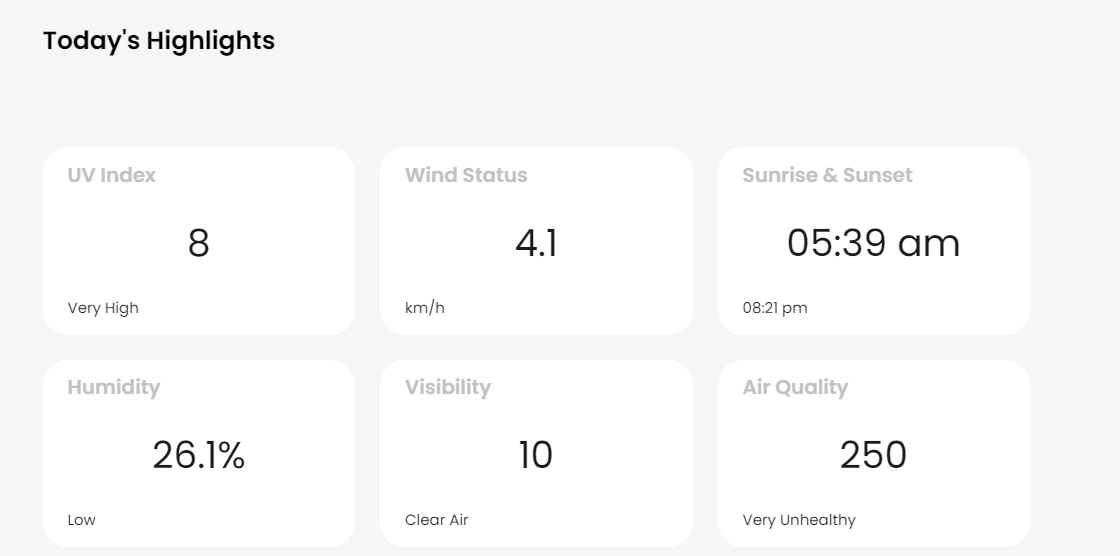
IMPLEMENTATION AND CODING

****









**CODING**

HTML

<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>

<meta charset="utf-8" />

<title>Weather App</title>

<link rel="stylesheet" href="style.css" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<script

src="https://kit.fontawesome.com/64d58efce2.js"

crossorigin="anonymous"

></script>

</head>

<body>

<div class="wrapper">

<div class="sidebar">

<div>

<form class="search" id="search">

<input type="text" id="query" placeholder="Search..." />

<button><i class="fas fa-search"></i></button>

</form>

<div class="weather-icon">

<img id="icon" src="icons/sun/4.png" alt="" />

</div>

<div class="temperature">

<h1 id="temp">0</h1>

<span class="temp-unit">°C</span>

</div>

<div class="date-time">

<p id="date-time">Monday, 12:00</p>

</div>

<div class="divider"></div>

<div class="condition-rain">

<div class="condition">

<i class="fas fa-cloud"></i>

<p id="condition">condition</p>

</div>

<div class="rain">

<i class="fas fa-tint"></i>

<p id="rain">perc - 0%</p>

</div>

</div>

</div>

<div class="location">

<div class="location-icon">

<i class="fas fa-map-marker-alt"></i>

</div>

<div class="location-text">

<p id="location">location</p>

</div>

</div>

</div>

<div class="main">

<nav>

<ul class="options">

<button class="hourly">today</button>

<button class="week active">week</button>

</ul>

<ul class="options units">

<button class="celcius active">°C</button>

<button class="fahrenheit">°F</button>

</ul>

</nav>

<div class="cards" id="weather-cards"></div>

<div class="highlights">

<h2 class="heading">today's highlights</h2>

<div class="cards">

<div class="card2">

<h4 class="card-heading">UV Index</h4>

<div class="content">

<p class="uv-index">0</p>

<p class="uv-text">Low</p>

</div>

</div>

<div class="card2">

<h4 class="card-heading">Wind Status</h4>

<div class="content">

<p class="wind-speed">0</p>

<p>km/h</p>

</div>

</div>

<div class="card2">

<h4 class="card-heading">Sunrise & Sunset</h4>

<div class="content">

<p class="sun-rise">0</p>

<p class="sun-set">0</p>

</div>

</div>

<div class="card2">

<h4 class="card-heading">Humidity</h4>

<div class="content">

<p class="humidity">0</p>

<p class="humidity-status">Normal</p>

</div>

</div>

<div class="card2">

<h4 class="card-heading">Visibility</h4>

<div class="content">

<p class="visibilty">0</p>

<p class="visibilty-status">Normal</p>

</div>

</div>

<div class="card2">

<h4 class="card-heading">Air Quality</h4>

<div class="content">

<p class="air-quality">0</p>

<p class="air-quality-status">Normal</p>

</div>

</div>

</div>

</div>

</div>

</div>

<script type="module" src="script.js"></script>

</body>

</html>

CSS

@importurl("[https://fonts.googleapis.com/css2?family=Poppins:wght@ 400;50](https://fonts.googleapis.com/css2?family=Poppins:wght@400;500); 600&display=swap");

:root {

--primary-color: #5598fd;

}

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: "Poppins", sans-serif;

}

body {

display: flex;

justify-content: center;

min-height: 100vh;

min-width: 1000px;

padding: 50px;

background: var(--primary-color);

background-image: linear-gradient(rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)),

url("./images/cd.jpg");

background-size: cover;

background-position: center;

transition: background-image 0.3s ease;

}

img {

width: 100%;

}

.wrapper {

display: flex;

width: 1200px;

min-width: 900px;

border-radius: 20px;

overflow: hidden;

}

.sidebar {

width: 30%;

min-width: 250px;

padding: 20px;

background: rgba(255, 255, 255, 0.815);

display: flex;

flex-direction: column;

justify-content: space-between;

}

.search {

display: flex;

align-items: center;

justify-content: space-between;

margin-bottom: 30px;

margin-top: 20px;

position: relative;

}

.search input {

width: 100%;

height: 40px;

border: 1px solid #ced4da;

border-top-left-radius: 5px;

border-bottom-left-radius: 5px;

padding: 0 15px;

font-size: 14px;

color: #495057;

}

.search input:focus {

outline: none;

border: 1px solid var(--primary-color);

}

.search button {

min-width: 40px;

height: 40px;

border: none;

border-top-right-radius: 5px;

border-bottom-right-radius: 5px;

background: var(--primary-color);

color: #fff;

font-size: 14px;

cursor: pointer;

}

.search ul {

max-height: 300px;

overflow-y: auto;

position: absolute;

width: 100%;

top: 40px;

border-radius: 5px;

transition: all 0.3s ease;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

background-color: #fff;

}

.search ul li {

padding: 10px 15px;

border-bottom: 1px solid #f1f1f1;

cursor: pointer;

text-transform: capitalize;

}

.search ul li:last-child {

border-bottom: none;

}

.search ul li:hover {

background-color: #f1f1f1;

}

.search ul li.active {

background-color: #f1f1f1;

}

.weather-icon {

width: 100%;

height: 150px;

text-align: center;

margin-top: 20px;

margin-bottom: 100px;

}

.weather-icon #icon {

width: 80%;

object-fit: cover;

}

.temperature {

display: flex;

}

.temperature #temp {

font-size: 70px;

font-weight: 100;

line-height: 1;

}

.temperature span {

font-size: 40px;

margin-top: -10px;

display: block;

}

.divider {

width: 100%;

height: 1px;

background: #e9ecef;

margin: 20px 0;

}

.condition-rain {

font-size: 12px;

text-transform: capitalize;

}

.condition-rain div {

display: flex;

align-items: center;

gap: 10px;

margin-bottom: 10px;

}

.condition-rain div i {

width: 20px;

}

.location {

display: flex;

align-items: center;

font-size: 14px;

gap: 10px;

margin-top: 10px;

}

.main {

width: 100%;

min-width: 400px;

padding: 20px 40px;

background-color: #f6f6f8;

position: relative;

padding-bottom: 90px;

}

.main nav {

display: flex;

align-items: center;

justify-content: space-between;

}

.main nav .options {

display: flex;

gap: 20px;

align-items: center;

}

.main nav .options button {

border: none;

background: none;

font-size: 16px;

font-weight: 600;

color: #495057;

cursor: pointer;

text-transform: capitalize;

}

.main nav .options button.active {

color: var(--primary-color);

}

.main nav .units button {

width: 40px;

height: 40px;

border-radius: 50%;

color: #1a1a1a;

background-color: #fff;

}

.main nav .units button.active {

color: #fff;

background-color: #1a1a1a;

}

.main .cards {

display: flex;

flex-wrap: wrap;

gap: 20px;

margin-top: 50px;

}

.cards .card {

width: 100px;

height: 130px;

border-radius: 20px;

color: #1a1a1a;

background-color: #fff;

text-align: center;

padding: 10px 0;

display: flex;

flex-direction: column;

justify-content: space-between;

}

.card h2 {

font-size: 15px;

font-weight: 600;

}

.card .card-icon {

width: 50%;

margin: 0 auto;

}

.card .day-temp {

font-size: 12px;

display: flex;

justify-content: center;

display: flex;

}

.highlights {

display: flex;

flex-wrap: wrap;

gap: 20px;

margin-top: 50px;

}

.highlights .heading {

width: 100%;

font-size: 20px;

font-weight: 600;

text-transform: capitalize;

}

.card2 {

width: 250px;

height: 150px;

border-radius: 20px;

color: #1a1a1a;

background-color: #fff;

padding: 10px 20px;

display: flex;

flex-direction: column;

}

.card2 .card-heading {

color: #c2c2c2;

}

.card2 .content {

margin-top: 20px;

}

.card2 .content p:first-child {

text-align: center;

font-size: 30px;

}

.card2 .content p:nth-child(2) {

font-size: 12px;

margin-top: 20px;

text-align: left;

}

.credits {

text-align: center;

font-size: 12px;

color: #c2c2c2;

position: absolute;

bottom: 30px;

left: 50%;

transform: translateX(-50%);

}

Java Script

const temp = document.getElementById("temp"),

date = document.getElementById("date-time"),

condition = document.getElementById("condition"),

rain = document.getElementById("rain"),

mainIcon = document.getElementById("icon"),

currentLocation = document.getElementById("location"),

uvIndex = document.querySelector(".uv-index"),

uvText = document.querySelector(".uv-text"),

windSpeed = document.querySelector(".wind-speed"),

sunRise = document.querySelector(".sun-rise"),

sunSet = document.querySelector(".sun-set"),

humidity = document.querySelector(".humidity"),

visibilty = document.querySelector(".visibilty"),

humidityStatus = document.querySelector(".humidity-status"),

airQuality = document.querySelector(".air-quality"),

airQualityStatus = document.querySelector(".air-quality-status"),

visibilityStatus = document.querySelector(".visibilty-status"),

searchForm = document.querySelector("#search"),

search = document.querySelector("#query"),

celciusBtn = document.querySelector(".celcius"),

fahrenheitBtn = document.querySelector(".fahrenheit"),

tempUnit = document.querySelectorAll(".temp-unit"),

hourlyBtn = document.querySelector(".hourly"),

weekBtn = document.querySelector(".week"),

weatherCards = document.querySelector("#weather-cards");

let currentCity = "";

let currentUnit = "c";

let hourlyorWeek = "week";

// function to get date and time

function getDateTime() {

let now = new Date(),

hour = now.getHours(),

minute = now.getMinutes();

let days = [

"Sunday",

"Monday",

"Tuesday",

"Wednesday",

"Thursday",

"Friday",

"Saturday",

];

hour = hour % 12;

if (hour < 10) {

hour = "0" + hour;

}

if (minute < 10) {

minute = "0" + minute;

}

let dayString = days[now.getDay()];

return `${dayString}, ${hour}:${minute}`;

}

date.innerText = getDateTime();

setInterval(() => {

date.innerText = getDateTime();

}, 1000);

function getPublicIp() {

fetch("https://geolocation-db.com/json/", {

method: "GET",

headers: {},

})

.then((response) => response.json())

.then((data) => {

currentCity = data.city;

getWeatherData(data.city, currentUnit, hourlyorWeek);

})

.catch((err) => {

console.error(err);

});

}

getPublicIp();

function getWeatherData(city, unit, hourlyorWeek) {

fetch(

`https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/${city}?unitGroup=metric&key=EJ6UBL2JEQGYB3AA4ENASN62J&contentType=json`,

{

method: "GET",

headers: {},

}

)

.then((response) => response.json())

.then((data) => {

let today = data.currentConditions;

if (unit === "c") {

temp.innerText = today.temp;

} else {

temp.innerText = celciusToFahrenheit(today.temp);

}

currentLocation.innerText = data.resolvedAddress;

condition.innerText = today.conditions;

rain.innerText = "Perc - " + today.precip + "%";

uvIndex.innerText = today.uvindex;

windSpeed.innerText = today.windspeed;

measureUvIndex(today.uvindex);

mainIcon.src = getIcon(today.icon);

changeBackground(today.icon);

humidity.innerText = today.humidity + "%";

updateHumidityStatus(today.humidity);

visibilty.innerText = today.visibility;

updateVisibiltyStatus(today.visibility);

airQuality.innerText = today.winddir;

updateAirQualityStatus(today.winddir);

if (hourlyorWeek === "hourly") {

updateForecast(data.days[0].hours, unit, "day");

} else {

updateForecast(data.days, unit, "week");

}

sunRise.innerText = covertTimeTo12HourFormat(today.sunrise);

sunSet.innerText = covertTimeTo12HourFormat(today.sunset);

})

.catch((err) => {

alert("City not found in our database");

});

}

function updateForecast(data, unit, type) {

weatherCards.innerHTML = "";

let day = 0;

let numCards = 0;

if (type === "day") {

numCards = 24;

} else {

numCards = 7;

}

for (let i = 0; i < numCards; i++) {

let card = document.createElement("div");

card.classList.add("card");

let dayName = getHour(data[day].datetime);

if (type === "week") {

dayName = getDayName(data[day].datetime);

}

let dayTemp = data[day].temp;

if (unit === "f") {

dayTemp = celciusToFahrenheit(data[day].temp);

}

let iconCondition = data[day].icon;

let iconSrc = getIcon(iconCondition);

let tempUnit = "°C";

if (unit === "f") {

tempUnit = "°F";

}

card.innerHTML = `

<h2 class="day-name">${dayName}</h2>

<div class="card-icon">

<img src="${iconSrc}" class="day-icon" alt="" />

</div>

<div class="day-temp">

<h2 class="temp">${dayTemp}</h2>

<span class="temp-unit">${tempUnit}</span>

</div>

`;

weatherCards.appendChild(card);

day++;

}

}

function getIcon(condition) {

if (condition === "partly-cloudy-day") {

return "icons/sun/27.png";

} else if (condition === "partly-cloudy-night") {

return "icons/moon/15.png";

} else if (condition === "rain") {

return "icons/rain/39.png";

} else if (condition === "clear-day") {

return "icons/sun/26.png";

} else if (condition === "clear-night") {

return "icons/moon/10.png";

} else {

return "icons/sun/26.png";

}

}

function changeBackground(condition) {

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

let bg = "";

if (condition === "partly-cloudy-day") {

bg = "images/pc.jpg";

} else if (condition === "partly-cloudy-night") {

bg = "images/pcn.jpg";

} else if (condition === "rain") {

bg = "images/rain.jpg";

} else if (condition === "clear-day") {

bg = "images/cd.jpg";

} else if (condition === "clear-night") {

bg = "images/cn.jpg";

} else {

bg = "images/pc.jpg";

}

body.style.backgroundImage = `linear-gradient( rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5) ),url(${bg})`;

}

function getHour(time) {

let hour = time.split(":")[0];

let min = time.split(":")[1];

if (hour > 12) {

hour = hour - 12;

return `${hour}:${min} PM`;

} else {

return `${hour}:${min} AM`;

}

}

function covertTimeTo12HourFormat(time) {

let hour = time.split(":")[0];

let minute = time.split(":")[1];

let ampm = hour >= 12 ? "pm" : "am";

hour = hour % 12;

hour = hour ? hour : 12;

hour = hour < 10 ? "0" + hour : hour;

minute = minute < 10 ? "0" + minute : minute;

let strTime = hour + ":" + minute + " " + ampm;

return strTime;

}

function getDayName(date) {

let day = new Date(date);

let days = [

"Sunday",

"Monday",

"Tuesday",

"Wednesday",

"Thursday",

"Friday",

"Saturday",

];

return days[day.getDay()];

}

function measureUvIndex(uvIndex) {

if (uvIndex <= 2) {

uvText.innerText = "Low";

} else if (uvIndex <= 5) {

uvText.innerText = "Moderate";

} else if (uvIndex <= 7) {

uvText.innerText = "High";

} else if (uvIndex <= 10) {

uvText.innerText = "Very High";

} else {

uvText.innerText = "Extreme";

}

}

function updateHumidityStatus(humidity) {

if (humidity <= 30) {

humidityStatus.innerText = "Low";

} else if (humidity <= 60) {

humidityStatus.innerText = "Moderate";

} else {

humidityStatus.innerText = "High";

}

}

function updateVisibiltyStatus(visibility) {

if (visibility <= 0.03) {

visibilityStatus.innerText = "Dense Fog";

} else if (visibility <= 0.16) {

visibilityStatus.innerText = "Moderate Fog";

} else if (visibility <= 0.35) {

visibilityStatus.innerText = "Light Fog";

} else if (visibility <= 1.13) {

visibilityStatus.innerText = "Very Light Fog";

} else if (visibility <= 2.16) {

visibilityStatus.innerText = "Light Mist";

} else if (visibility <= 5.4) {

visibilityStatus.innerText = "Very Light Mist";

} else if (visibility <= 10.8) {

visibilityStatus.innerText = "Clear Air";

} else {

visibilityStatus.innerText = "Very Clear Air";

}

}

function updateAirQualityStatus(airquality) {

if (airquality <= 50) {

airQualityStatus.innerText = "Good";

} else if (airquality <= 100) {

airQualityStatus.innerText = "Moderate";

} else if (airquality <= 150) {

airQualityStatus.innerText = "Unhealthy for Sensitive Groups";

} else if (airquality <= 200) {

airQualityStatus.innerText = "Unhealthy";

} else if (airquality <= 250) {

airQualityStatus.innerText = "Very Unhealthy";

} else {

airQualityStatus.innerText = "Hazardous";

}

}

searchForm.addEventListener("submit", (e) => {

e.preventDefault();

let location = search.value;

if (location) {

currentCity = location;

getWeatherData(location, currentUnit, hourlyorWeek);

}

});

function celciusToFahrenheit(temp) {

return ((temp \* 9) / 5 + 32).toFixed(1);

}

var currentFocus;

search.addEventListener("input", function (e) {

removeSuggestions();

var a,

b,

i,

val = this.value;

if (!val) {

return false;

}

currentFocus = -1;

a = document.createElement("ul");

a.setAttribute("id", "suggestions");

this.parentNode.appendChild(a);

for (i = 0; i < cities.length; i++) {

if (

cities[i].name.substr(0, val.length).toUpperCase() == val.toUpperCase()

) {

b = document.createElement("li");

b.innerHTML =

"<strong>" + cities[i].name.substr(0, val.length) + "</strong>";

b.innerHTML += cities[i].name.substr(val.length);

b.innerHTML += "<input type='hidden' value='" + cities[i].name + "'>";

b.addEventListener("click", function (e) {

search.value = this.getElementsByTagName("input")[0].value;

removeSuggestions();

});

a.appendChild(b);

}

}

});

search.addEventListener("keydown", function (e) {

var x = document.getElementById("suggestions");

if (x) x = x.getElementsByTagName("li");

if (e.keyCode == 40) {

currentFocus++;

addActive(x);

} else if (e.keyCode == 38) {

currentFocus--;

addActive(x);

}

if (e.keyCode == 13) {

e.preventDefault();

if (currentFocus > -1) {

if (x) x[currentFocus].click();

}

}

});

function addActive(x) {

if (!x) return false;

removeActive(x);

if (currentFocus >= x.length) currentFocus = 0;

if (currentFocus < 0) currentFocus = x.length - 1;

x[currentFocus].classList.add("active");

}

function removeActive(x) {

for (var i = 0; i < x.length; i++) {

x[i].classList.remove("active");

}

}

function removeSuggestions() {

var x = document.getElementById("suggestions");

if (x) x.parentNode.removeChild(x);

}

fahrenheitBtn.addEventListener("click", () => {

changeUnit("f");

});

celciusBtn.addEventListener("click", () => {

changeUnit("c");

});

function changeUnit(unit) {

if (currentUnit !== unit) {

currentUnit = unit;

tempUnit.forEach((elem) => {

elem.innerText = `°${unit.toUpperCase()}`;

});

if (unit === "c") {

celciusBtn.classList.add("active");

fahrenheitBtn.classList.remove("active");

} else {

celciusBtn.classList.remove("active");

fahrenheitBtn.classList.add("active");

}

getWeatherData(currentCity, currentUnit, hourlyorWeek);

}

}

hourlyBtn.addEventListener("click", () => {

changeTimeSpan("hourly");

});

weekBtn.addEventListener("click", () => {

changeTimeSpan("week");

});

function changeTimeSpan(unit) {

if (hourlyorWeek !== unit) {

hourlyorWeek = unit;

if (unit === "hourly") {

hourlyBtn.classList.add("active");

weekBtn.classList.remove("active");

} else {

hourlyBtn.classList.remove("active");

weekBtn.classList.add("active");

}

getWeatherData(currentCity, currentUnit, hourlyor

**CHAPTER 5**

TESTING & TEST RESULTS

Testing is a process to show the correctness of the program. Testing is needed to show completeness, t improve the quality of the software and to provide the maintenance aid. Some testing standards are therefore necessary reduce the testing costs and operation time. Testing software extends throughout the coding phase and it represents the ultimate review of configurations, design and coding. Based on the way the software reacts to this application testing, we can decide whether the configuration has been built is study or not. All components of an application are tested, as the failure to do so many results in a series of bugs after the software is put to use.

**Blackbox Testing:**

Blackbox testing also called behavioral testing, focuses on the functional requirements of software. This application testing approach enables the software engineer to derive the input conditions from the application and will fully exercise all requirements for a program. Blackbox testing attempts to find the errors like

* Incorrect or missing functions
* Interface errors
* Errors in data structures or external database access
* Behavior or performance errors
* Initialization and termination errors

In Blackbox testing software is exercised over a full range of inputs and outputs are observed for correctness.

**White Box Testing:**

Whitebox testing is also called Glassbox testing is a test case design control; structure of the procedural design to derive test cases using Whitebox testing method, the software engineer can derive the test cases that guarantee the report all independent paths within the module have been exercised at least once. Exercise all logic decisions on their true or false sides. Execute all loops at their boundaries and within their operational bounds. Exercise internal data structure to ensure their validity.

**5. SOFTWARE TESTING STRATEGIES**

Testing involves

∙ Unit testing

∙ Integration testing

∙ Acceptance testing

The first level of test is unit testing. The purpose of unit testing is to ensure the application is fully tested. The second step is integration testing. In this application individual program units or programs are integrated and tested as a complete system to ensure the software requirements are met. Acceptance Testing involves planning and the execution of various types of tests in order to demonstrate the implemented software system satisfies the requirements. Finally our project meets the requirements after going through all the levels of testing.

**DESCRIPTION:**

1. Login/Registration Screen:

- A screen where users can log in or register to access the e tournament application.

- Includes fields for entering username/email and password.

2. Dashboard/Home Screen:

- The main screen after logging in, providing an overview of important information.

- Displays relevant statistics, such as the number of events and the event joined by the user.

- Can include quick access buttons for common actions or tasks.

3. Joined Tournaments Screen:

- Shows the tournaments that user has joined.

- Displays a list of available events and their details.

4. FAQ Screen:

- Provides instructions for the application.

- Commonly asked questions.

.

5. Profile Screen:

- Provides a platform to view the profile.

- Includes option to update or logout the profile.

6. Policy Screen:

- Displays the privacy policy.

- Also includes refund policy and terms and conditions.

These are just a few examples of screens or components that you might find in a e tournament application. You can use this description as a starting point to create visual representations or wireframes of the application using design tools like Adobe XD, Sketch, or Figma.

**STEPS INVOLVED:**

1. Project Setup:

- Set up a new Android project in Android Studio.

- Define the project structure and dependencies.

2. User Authentication:

- Implement a user registration and login system.

- Store user credentials securely (e.g., using Firebase Authentication).

3. Database Integration:

- Integrate Firebase Realtime Database or Firestore to store and retrieve events-related data.

- Design the database schema to store information such as events, joined events, profile update etc.

4. User Interface:

- Create the necessary activities, fragments, and layouts for various screens.

- Design the user interface components using XML layout files (e.g., login screen, dashboard, event manager, profile dashboard etc).

5. Event Manager:

- Implement functionality for joined events .

- Allow administrators to manage event availability, maintain schedules etc.

6. Fee Management:

- Develop features for managing fees and winning price.

- Allow administrators to record and track fee payments and winners price pool.

7. Notice Board:

- Implement a notice board feature to display important event and announcements.

- Allow administrators to create, edit, and delete notices.

8. Reports and Analytics:

- Develop features to generate reports and analytics related to tournament occupancy, fee collections, maintenance request trends, etc.

9. Testing and Debugging:

- Conduct comprehensive testing to ensure the application functions as intended.

- Identify and fix any bugs or issues that arise during testing.

10. Deployment and Release:

- Prepare the application for release by signing and generating APK files.

- Publish the application on the Google Play Store (if desired).

**CHAPTER 6**

**CONCLUSION, AND FUTURE WORK**

* 1. **Conclusion of the report**

Now a day’s there is a big demand of different types of applications, which is because IT has become the main part of our New World. There is a big need of different applications. People want application for every specific task from work to entertainment. We have developed the application “Weather WebApp” which works easy on any given web browser.

The application has been tested and found to be working as per the given criteria. It can be safely concluded that the application possesses a highly efficient UI system and is working properly and meeting to all the requirements of the user. The application gives the user maximum flexibility in the types of touch and other device movements.

* 1. **Future Work**

The field of Android development is constantly evolving, and there are several areas of future work that developers can focus on to stay up-to-date and take advantage of emerging trends and technologies. Here are some potential areas for future work in Android development:

1. Kotlin Multiplatform Mobile (KMM): Kotlin Multiplatform Mobile is gaining traction as a cross-platform development solution. Exploring KMM and leveraging its potential for code sharing between Android and iOS platforms can be a valuable future endeavor.

2. Jetpack Compose: Jetpack Compose is a modern UI toolkit for building native Android UIs declaratively. As it continues to mature, developers can invest time in learning and adopting Compose to create more efficient, flexible, and interactive user interfaces.

3. Augmented Reality (AR) and Virtual Reality (VR): With the increasing popularity of AR and VR applications, exploring frameworks like ARCore and Google VR SDK can open up new avenues for creating immersive experiences on Android devices.

4. Internet of Things (IoT): Android devices can act as gateways or controllers for IoT devices. Developers can explore integrating Android applications with IoT technologies and protocols such as Bluetooth, MQTT, and CoAP to build smart home or industrial automation solutions.

5. Machine Learning (ML) and Artificial Intelligence (AI): Integrating ML and AI capabilities into Android applications is a promising area. TensorFlow Lite, ML Kit, and other frameworks can be used to implement features like image recognition, natural language processing, and predictive analytics.

6. Wearable Devices: As wearable devices gain popularity, developing applications specifically for smartwatches, fitness trackers, and other wearables can offer unique user experiences and tap into the growing market.

7. Instant Apps: Instant Apps allow users to run Android applications without installing them. Exploring Instant Apps and developing modular, on-demand features can enhance user engagement and simplify app discovery.

8. Blockchain Integration: Blockchain technology offers opportunities for secure and decentralized applications. Exploring integration options such as Ethereum, Hyperledger, or other blockchain platforms can lead to innovative Android applications.

9. Accessibility: Focusing on creating inclusive and accessible applications is crucial. Future work can involve implementing features that improve accessibility, such as support for screen readers, magnification, color contrast, and alternative input methods.

1. Security and Privacy: With increased concerns about data security and privacy, developers should stay updated on best practices and consider implementing security measures like encryption, secure authentication, and data anonymization to protect user data

**REFERENCES**

* James E. White. Mobile agents. In J. Bradshaw, editor, Software Agents, chapter 18, pages 437–472. AAAI/MIT Press, Menlo Park, CA, 1997.
* Volker Roth and MehrdadJalali. Concepts and architecture of a security-centric mobile agent server. In Proc. Fifth International Symposium on Autonomous Decentralized Systems (ISADS 2001), pages 435–442, Dallas, Texas, U.S.A., March 2001. IEEE Computer Society. ISBN 0-7695-1065-5.
* Walter Binder and Volker Roth. Secure mobile agent systems using Java – where are we heading? In Proc. 17th ACM Symposium on Applied Computing, Special Track on Agents, Interactions, Mobility, and Systems (SAC/AIMS), Madrid, Spain, March 2002. ACM.
* S. G. Belmon and B. S. Yee. Mobile agents and intellectual property protection. In Rothermel and Hohl [22], pages 172–182.
* C. Arora, P. Nirankari, H. Ghosh, and S. Chaudhury. Content based image retrieval using mobile agents. In Third International Conference on Computational Intelligence and Multimedia Applications (ICCIMA ’99), pages 248–252, 1999.