Real-Time Weather Forecasting App

A PROJECT REPORT

Submitted By

Rishabh Sharma
(2000290140102)
Mayank Agrawal
(2000290140067)
Pramod Pandey
(2000290140090)

Submitted in partial Fulfillment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Ms. SHALIKA ARORA

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(MAY 2022)

DECLARATION

We hereby declare that the work presented in this report entitled “Real-Time Weather Forecasting App”, was carried out by Rishabh Sharma, Mayank Agrawal and Pramod Pandey. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute. We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

We affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Rishabh Sharma	Name: Mayank Agrawal
Roll No.: (2000290140102)	Roll. No.: (2000290140067)

Name: Pramod Pandey

Roll. No.: (2000290140090)

CERTIFICATE

Certified that **Rishabh Sharma <200029014005787>, Mayank Agrawal< 200029014005752> and Pramod Pandey<200029014005775>** have carried out the project work having "**Real-Time Weather Forecasting App**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Rishabh Sharma (2000290140102)

Mayank Agrawal (2000290140067)

Pramod Pandey (2000290140090)

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

Date:

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

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

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

ABSTRACT

Weather forecasting is the application of science and technology to predict the state of the atmosphere for a given location. Ancient weather forecasting methods usually relied on observed patterns of events, also termed pattern recognition. For example, it might be observed that if the sunset was particularly red, the following day often brought fair weather. However, not all these predictions prove reliable.

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(dataset). The role of the admin is to add previous weather data in database, so that system will calculate weather (estimated rainfall in inches) based on these data. Weather forecasting system takes parameters such as temperature, humidity, and wind and will forecast weather based on previous record therefore this prediction will prove reliable. This system can be used in Air Traffic, Marine, Agriculture, Forestry, Military, and Navy etc.

The report has been made in attempt to provide user with real time weather forecasting with 5 days in future forecasting. For making this project we have
studied various concepts related to the weather app and how different factors affect weather app. We also studied about various API Application Programming Interfaces that can be used to provide weather data in easier manner. The project aims at providing most details like humidity, air speed, and many other features.

The prediction of a weather can be used for future planning. Forecasting accuracy is the most important factor in selecting any forecasting methods. Research efforts in improving the accuracy of forecasting models are increasing since the last decade.
Weather forecasting is the attempt by meteorologists to predict the weather conditions at some future time and the weather conditions that may be expected. The climatic condition parameters are based on the temperature, wind, humidity, rainfall and size of data set.

ACKNOWLEDGEMENTS

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

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

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

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

Rishabh Sharma (2000290140102)
Mayank Agrawal (2000290140067)
Pramod Pandey (2000290140090)

TABLE OF CONTENTS

List of Chapters

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Project Description

Weather forecasting is the application of science and technology to predict the state of the atmosphere for a given location ancient weather forecasting methods usually rolled on observed patterns of events, also termed pattern recognition. People have attempted to predict the weather informally for millennia 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 condition or cloud cover, weather forecasting now relies on Computer based model that take many atmospheric factors into account. The inaccuracy of forecasting is due to the chaotic nature of the atmosphere, the massive computational power required to solve the equations that describe the atmosphere, the land, and the ocean, the error involved in measuring the initial conditions, and an incomplete understanding of atmospheric and related processes. 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 help narrow the error and provide confidence level in the forecast. For example, it might be observed that if the sunset was particularly red, the following day often brought fair weather.

However, not all of these predictions prove reliable. Here this system will predict weather based on parameters such as temperature, humidity and wind. This system is a web application with effective graphical user interface. User will enter current temperature; humidity and wind, System will take this parameter and will predict weather from previous data in database.

The role of the admin is to add previous weather data in database, so that system will calculate weather based on these data.

Weather forecasting system takes parameters such as temperature, humidity, and wind and will forecast weather based on previous record therefore this prediction will prove reliable. This system can be used in Air Traffic, Marine, Agriculture, Forestry, Military, and Navy etc. Forecasting the temperature and rain on a particular day and date is the main aim of this paper. Our system will process the users query and will mine the data from our repository to draw appropriate results Lasers will be provided with recommendations also and that is the key facility of our service. Personalized forecast is generated for each individual user based on their location.'

1.2 Project Scope

- ❖ The project mainly focuses on forecasting weather conditions using historical data.
- ❖ This can be done by extracting knowledge from this given data by using techniques such as association,
- ❖ Disaster Mitigation: Predicting storms, floods, droughts
- ❖ Helping those sectors which are most dependent on weather such as agriculture, aviation also depends on weather conditions.

1.3 Hardware / Software used in Project

Hardware Requirements

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

Processor	Pentium and above
Speed	1.1 GHz
RAM	512 MB (min)
Hard Disk	20GB

Keyboard	Standard Windows Keyboard
Mouse	Two or Three Button Mouse
Monitor	SVGA

Table.1.3.1 Hardware Requirements

Software Requirements

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

Weather Forecasting Using HTML, CSS, JavaScript

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

Table.1.3.2 Software Requirements

CHAPTER 2

FEASIBILITY STUDY

2.1 Technical Feasibility

A large part of determining resources has to do with assessing technical feasibility. The technical requirements considered in the proposed project are as this project is basically built for people so the basic requirement is to generate an environment for the person through the help of the project so that he can detect and predict the weather. The technical work which is done in project Weather forecasting application is considered technically feasible because its internal technical capability is sufficient to support the project requirements.

Here we are using an open weather map API for fetching the real time temperature, it can show the current weather, hourly forecast, daily forecast, climate forecast and historical weather. For selecting the countries and cities in them we have used Countries now API which provides geolocation, population, and general information about countries in simple JSON responses.

2.2 Operational Feasibility

Operational feasibility is dependent on human resources available for the project and involves projecting whether the system will be used if it is developed and implemented. Operational feasibility is a measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.
Operational feasibility reviews the willingness of the organization to support the proposed system. This is probably the most difficult of the feasibility to gauge. In order to determine this feasibility, it is important to understand the management commitment to the proposed project. When we search weather in this project, it will detect the current weather as well as it will predict the weather of next seven days.

2.3 Behavioral Feasibility

Behavioral Feasibility includes how strong the reaction of users will be towards the development of a new system that involves an application to be used in their daily life. So resistant to change is identified. It is an application which can be used anywhere and anytime with the help of internet connection it will also store the data in local storage so that if internet turns off, a person can see the last time data in this application.

2.4 Technical Feasibility

This project uses reactjs in which data is fetched with the help of two APIs first one is Countries now which is open source and freely available to anyone which will show the information about the countries. Second API is open weather map API which will show the data of weather and it needs a key to open in your system, a user can generate key and The API's generous free plan allows users up to 60 calls per minute.

CHAPTER 3

DESIGN

3.1 Data Flow Diagram

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

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

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

system, different details of a system are slowly introduced through the different hierarchies. The data flow diagramming technique also follows a simple set of intuitive concepts and rules.

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

Level 0

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

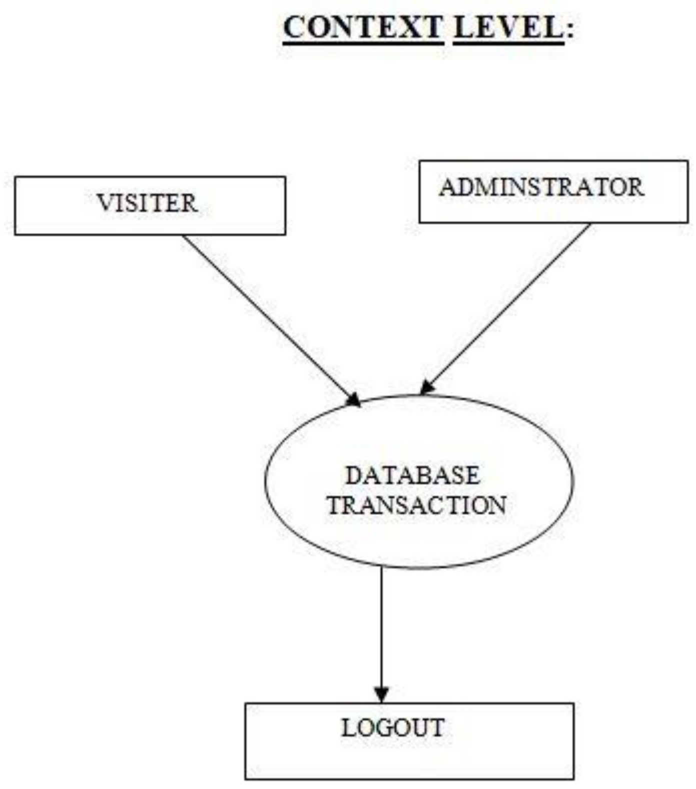


Fig.3.1.1 Level 0 DFD Weather Forecasting Application

Level 1

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

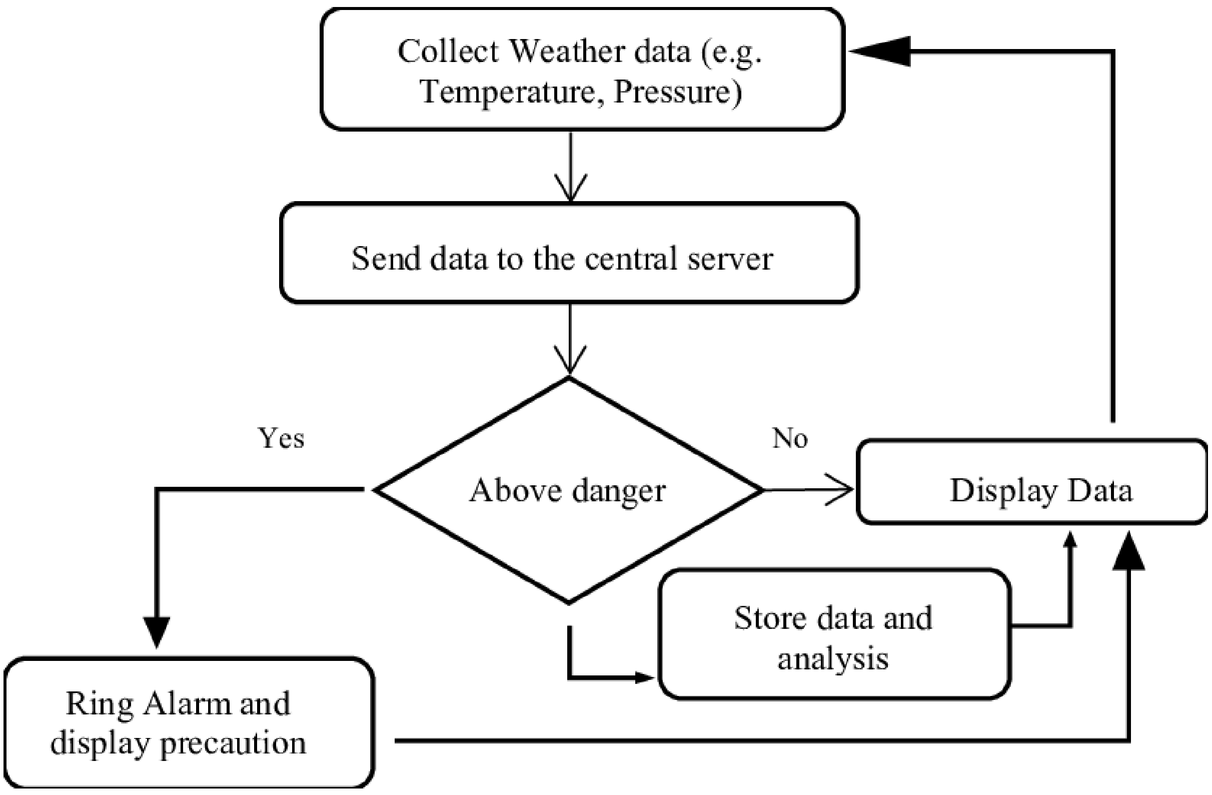

Collect Weather data (e.g. Temperature, Pressure)
Send data to the central server
Yes
No
Above danger
Display Data
Store data and analysis
Ring Alarm and display precaution

Fig.3.1.2 Level 1 DFD Weather Forecasting Application

Level 2

Complete details of every function performed by Admin. Every step is explained in detail. The four components of a data flow diagram (DFD) are:

External Entities/Terminators are outside of the system being modelled. Terminators represent where information comes from and where it goes. In designing a system, we have no idea about what these terminators do or how they do it. Processes modify the inputs in the process of generating the output Data Stores represent a place in the process where data comes to rest. A DFD does not say anything about the relative timing of the processes, so a data store might accumulate data over a year for the annual accounting process. Data Flows shows how data moves between terminators, processes, and data stores (those that cross the system boundary are known as 10 or Input Output Descriptions).

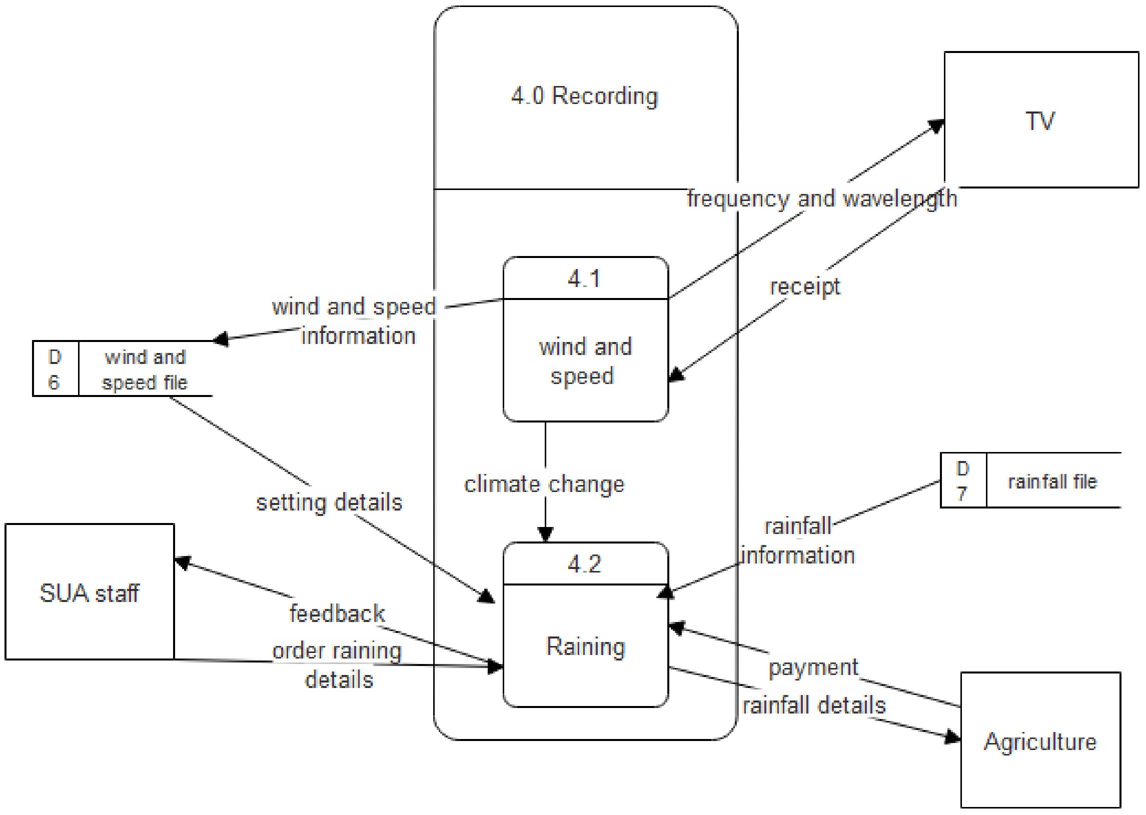


Fig.3.1.3 Level 2 DFD Weather Forecasting Application

3.2 Use Case Diagram

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

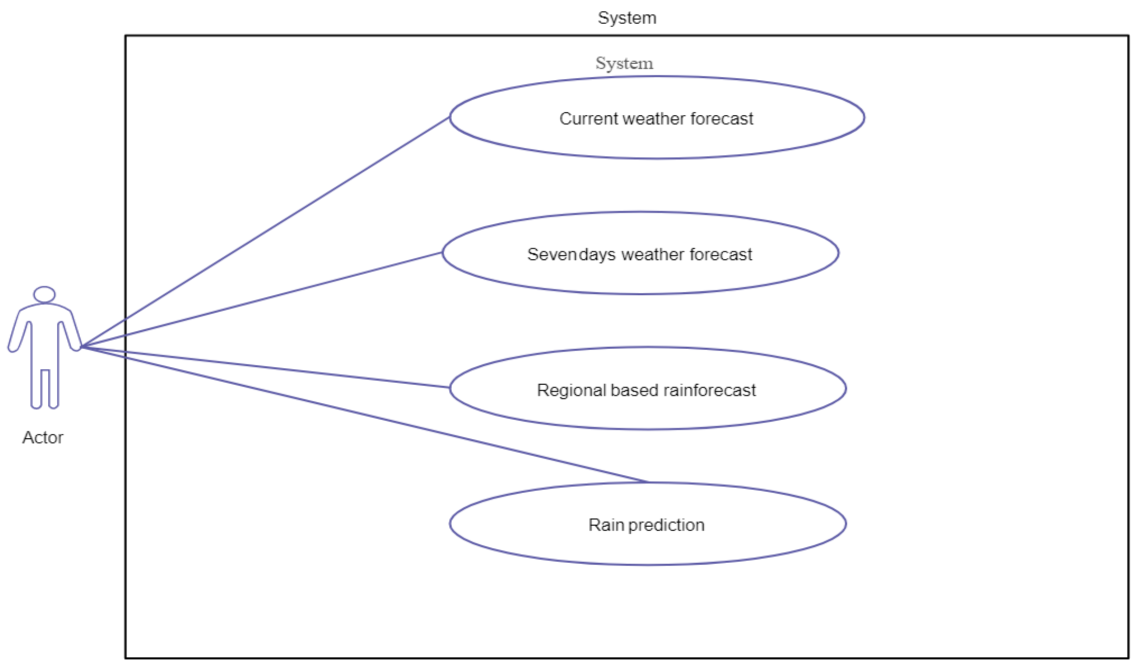


Fig.3.2.1 Use Case Diagram Weather Forecasting app

3.3 Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

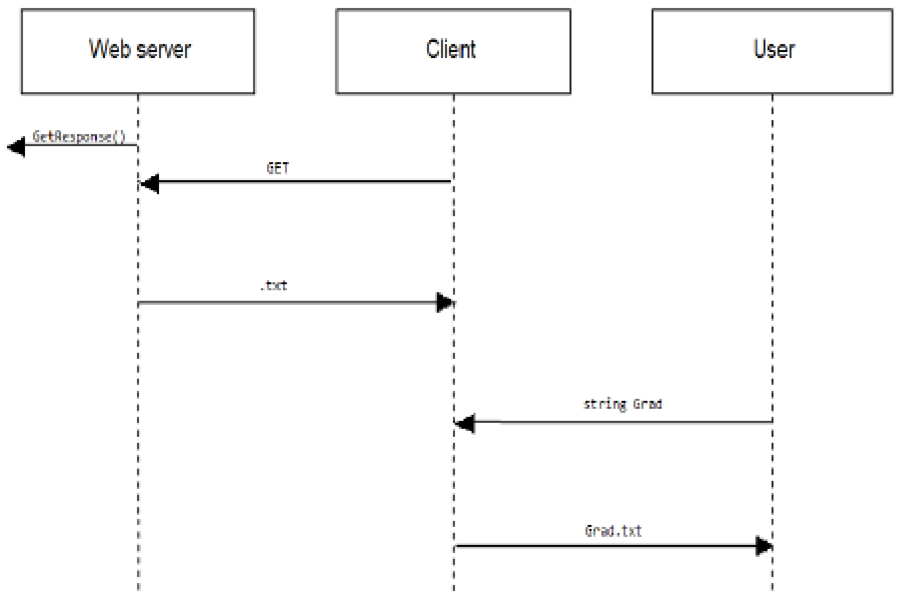


Fig.3.3.1 Sequence diagram Weather Forecasting app

3.4 Collaborative Diagram

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).

These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.

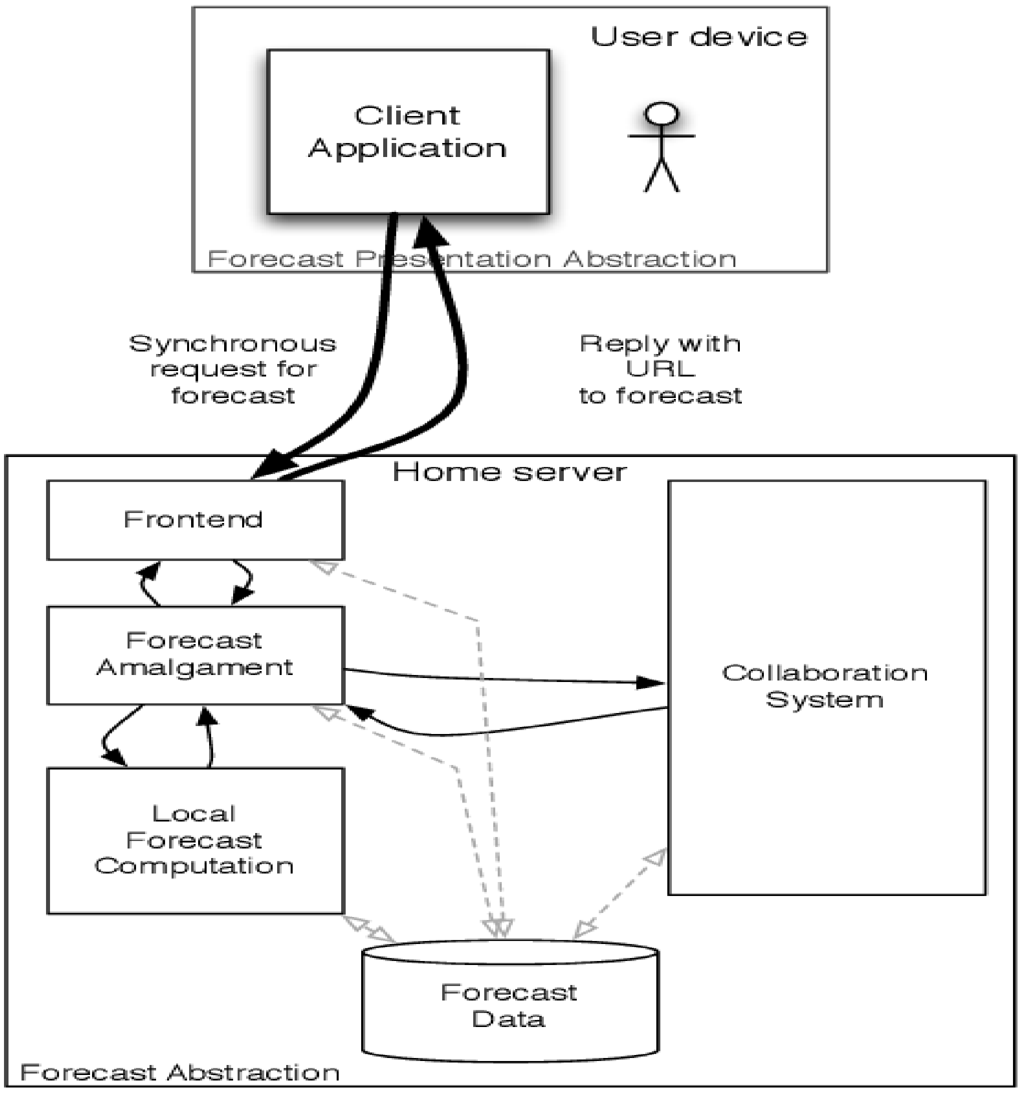


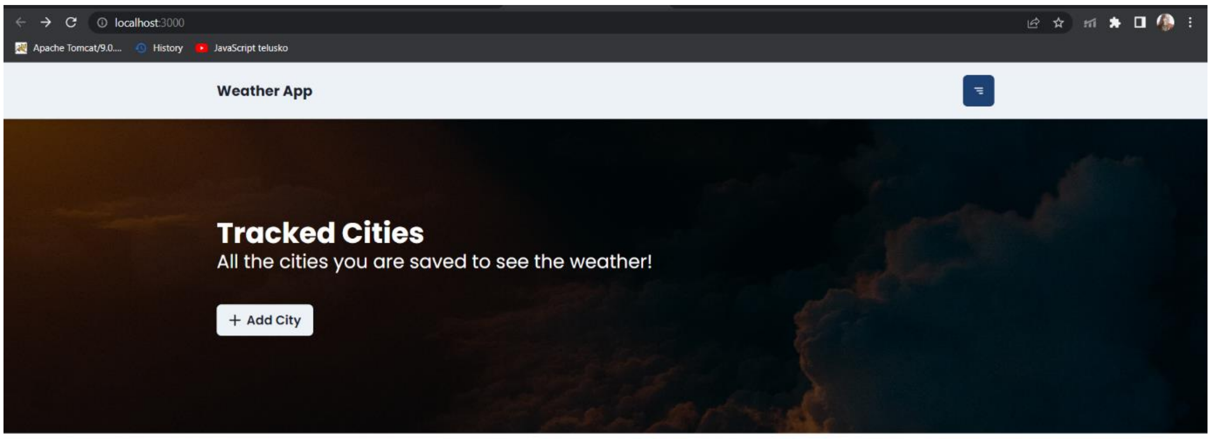
Fig.3.4.1 Collaborative diagram Weather Forecasting app

CHAPTER 4

FORM DESIGN

4.1 Input/Output Form Screenshots

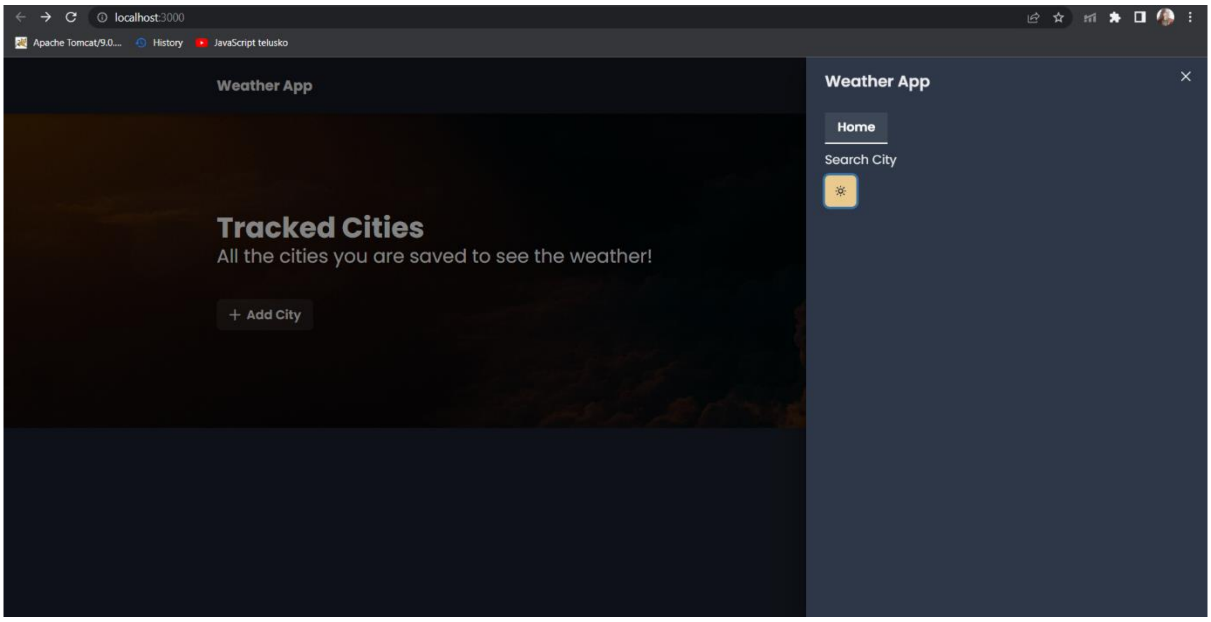
Home Page Screenshot



Tracked Cities

All the cities you are saved to see the weather!

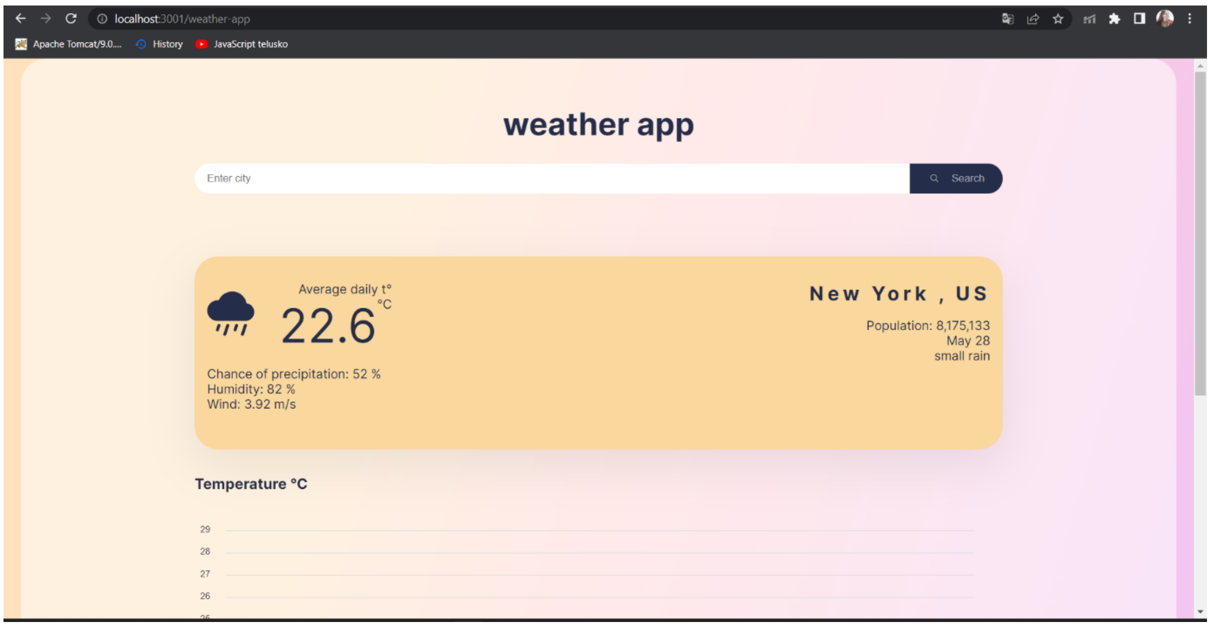
Dark theme Screenshot



Tracked Cities

All the cities you are saved to see the weather!

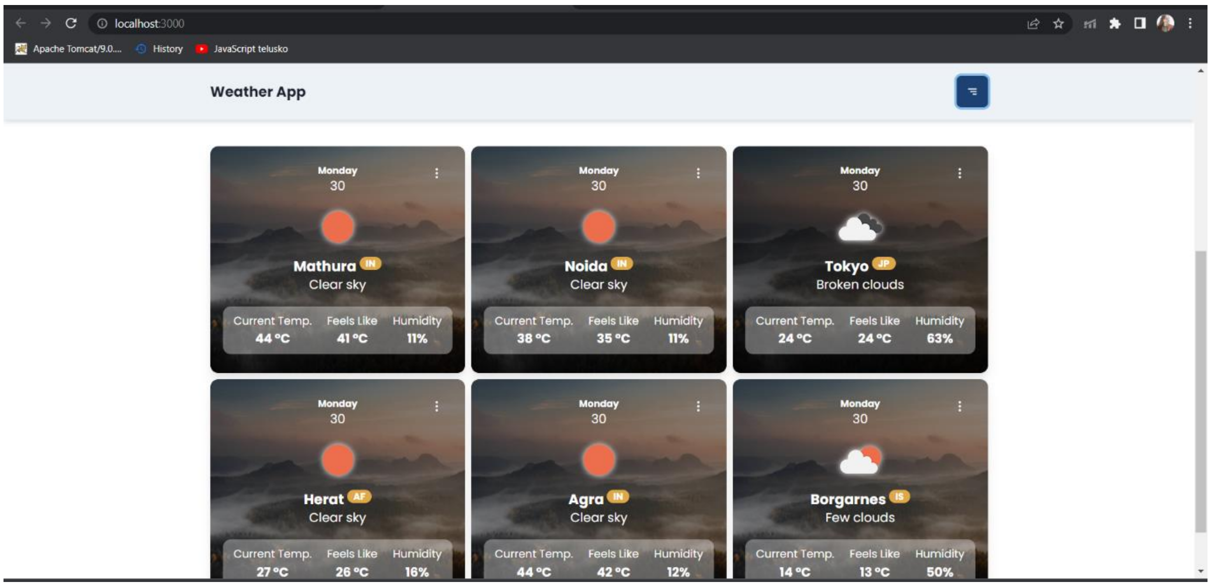
Detailed Temperature Screenshot



Average daily t°
22.6 °C
Chance of precipitation: 52 %
Humidity: 82 %
Wind: 3.92 m/s

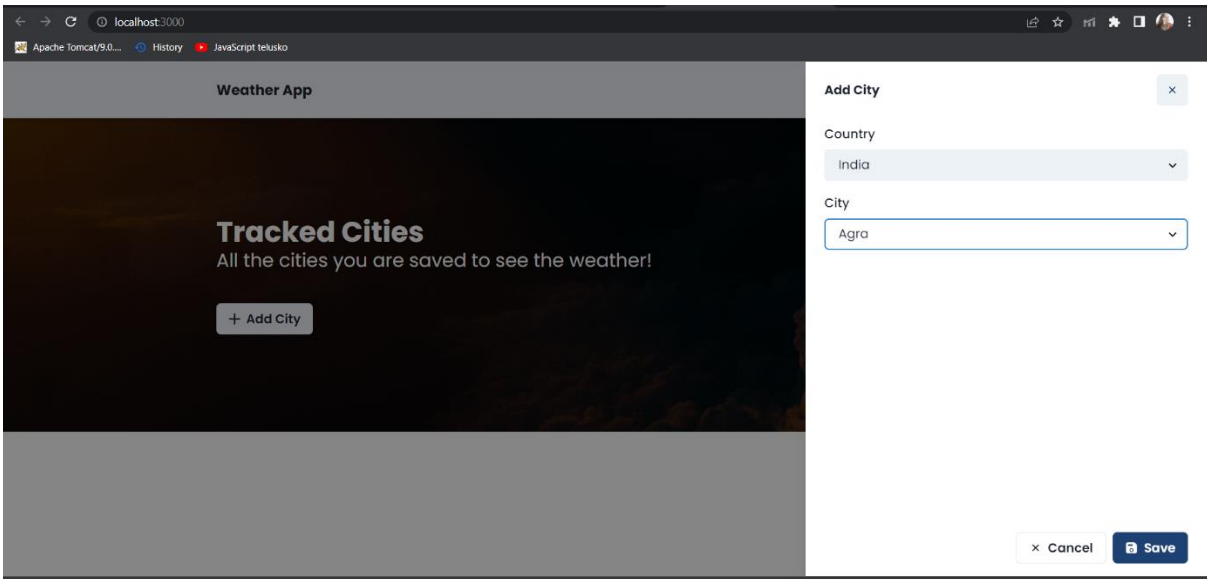
New York , US
Population: 8,175,133
May 28
small rain

Temperature Cards Screenshot



City	Day	Weather	Current Temp.	Feels Like	Humidity
Mathura IN	Monday 30	Clear sky	44 °C	41 °C	11%
Noida IN	Monday 30	Clear sky	38 °C	35 °C	11%
Tokyo JP	Monday 30	Broken clouds	24 °C	24 °C	63%
Herat AF	Monday 30	Clear sky	27 °C	26 °C	16%
Agra IN	Monday 30	Clear sky	44 °C	42 °C	12%
Borgarnes IS	Monday 30	Few clouds	14 °C	13 °C	50%

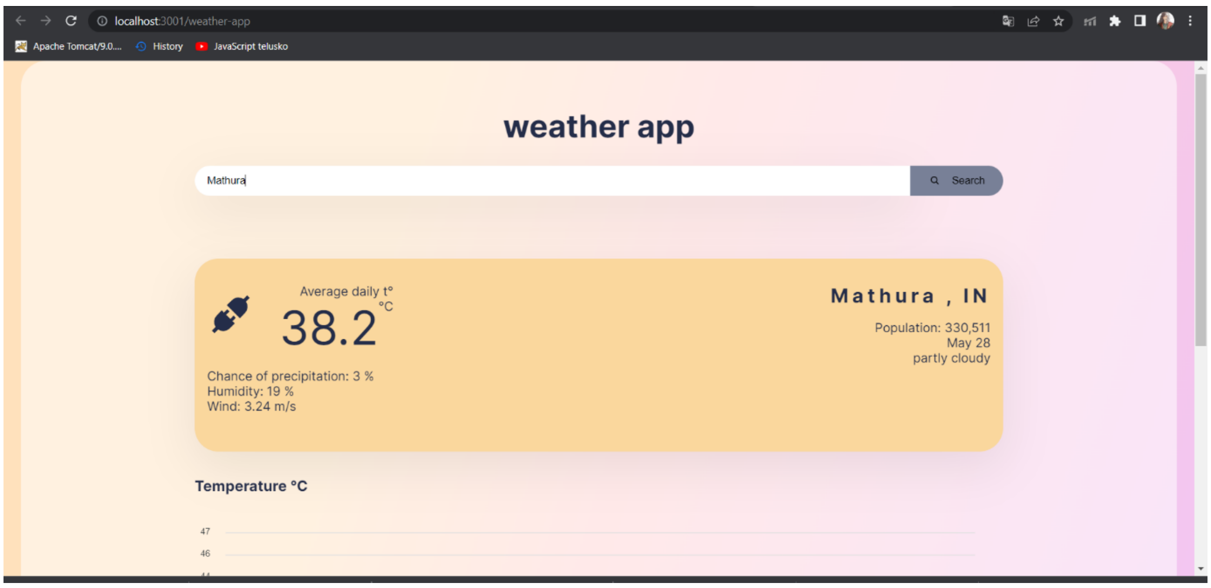
Input Screenshot 1



Country: India

City: Agra

Input Screenshot 2



Mathura

Average daily t° °C

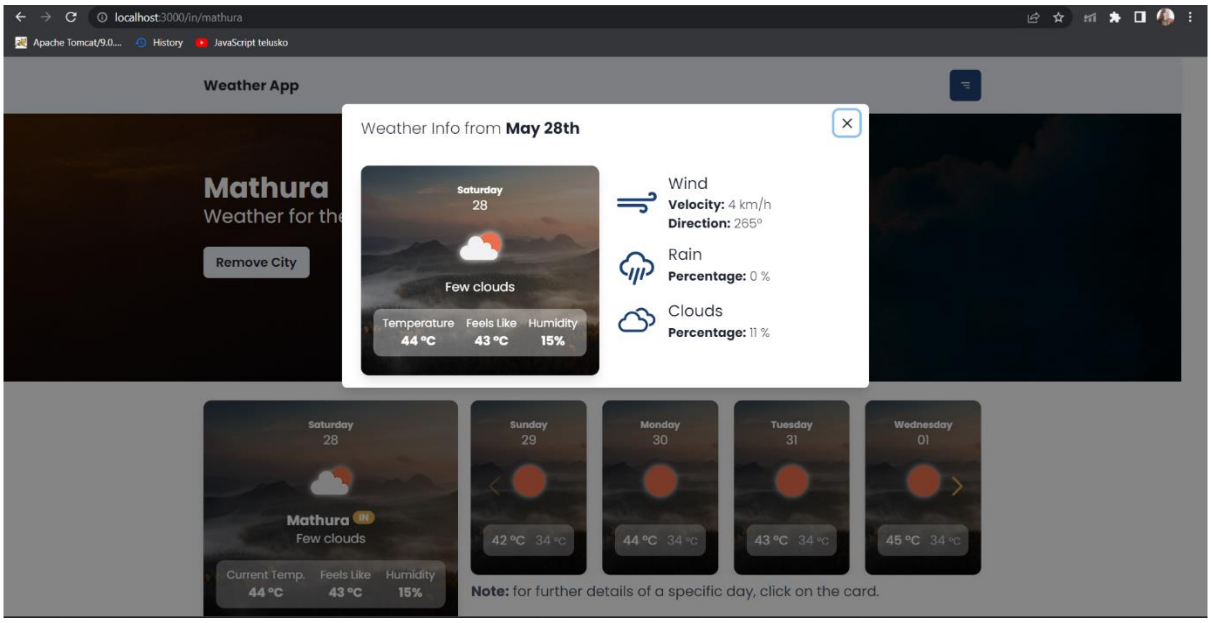
38.2

Chance of precipitation: 3 %
Humidity: 19 %
Wind: 3.24 m/s

Mathura , IN

Population: 330,511
May 28
partly cloudy

Output Screenshot 1



Weather Info from **May 28th**

Saturday 28
Few clouds
Temperature 44 °C · Feels Like 43 °C · Humidity 15%

Wind
Velocity: 4 km/h
Direction: 265°

Rain
Percentage: 0 %

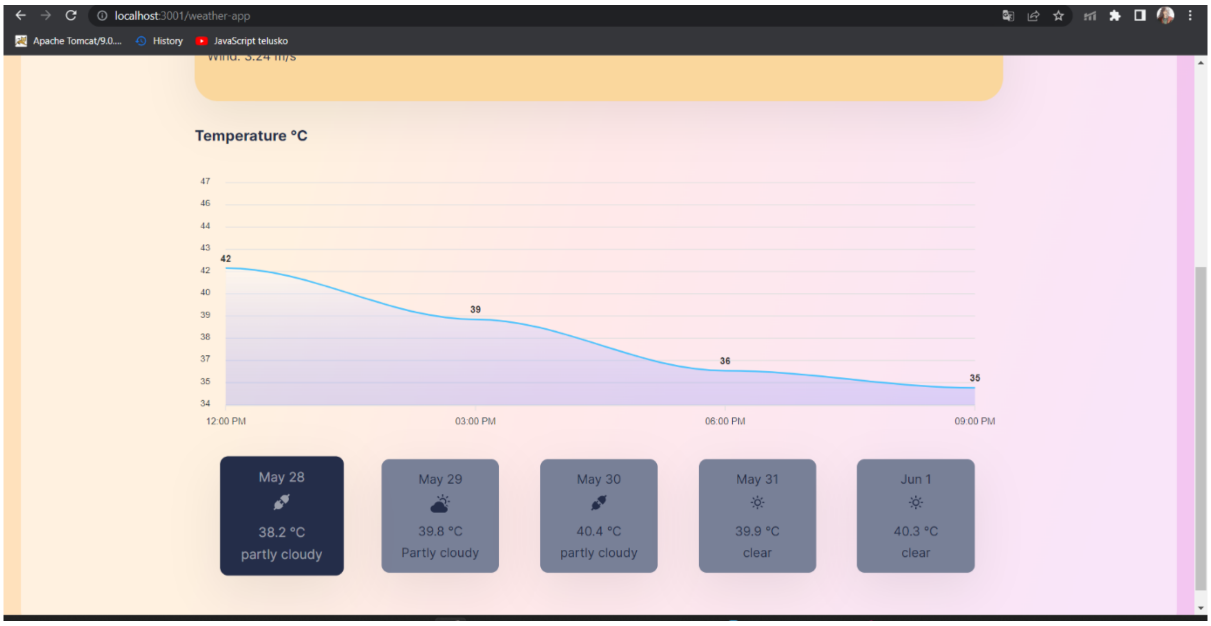
Clouds
Percentage: 11 %

Saturday 28 · Mathura IN · Few clouds · Current Temp. 44 °C · Feels Like 43 °C · Humidity 15%

Sunday 29	Monday 30	Tuesday 31	Wednesday 01
42 °C 34 °C	44 °C 34 °C	43 °C 34 °C	45 °C 34 °C

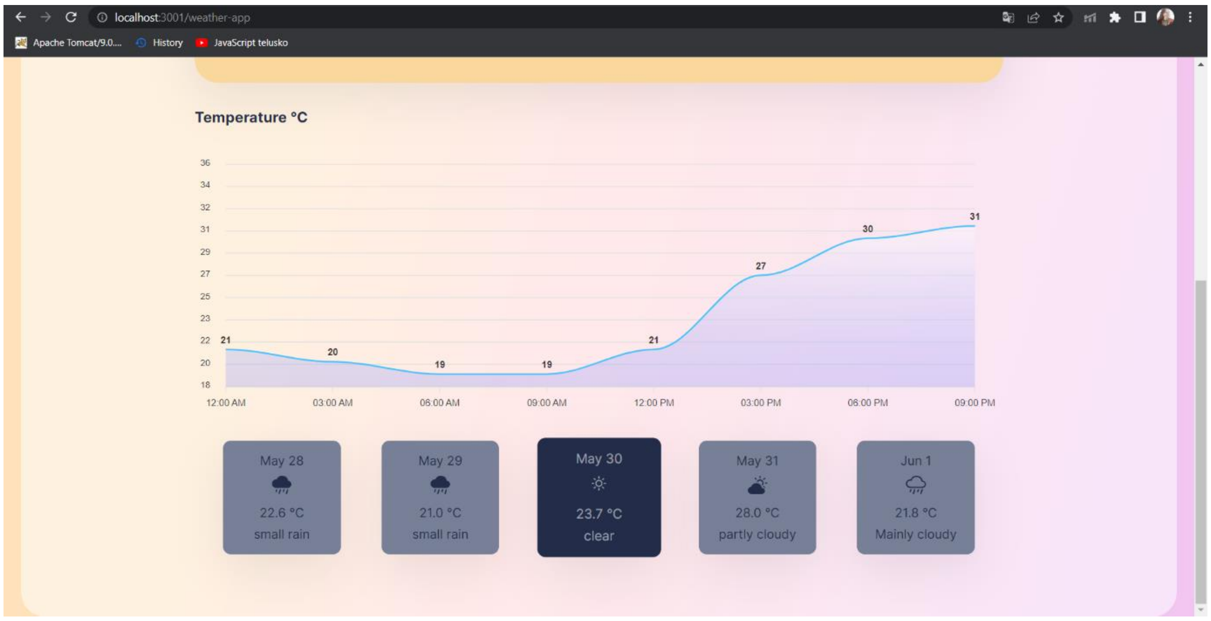
Note: for further details of a specific day, click on the card.

Output Screenshot 2



May 28	May 29	May 30	May 31	Jun 1
38.2 °C	39.8 °C	40.4 °C	39.9 °C	40.3 °C
partly cloudy	Partly cloudy	partly cloudy	clear	clear

Graphical Representation of other days Screenshot



May 28	May 29	May 30	May 31	Jun 1
22.6 °C	21.0 °C	23.7 °C	28.0 °C	21.8 °C
small rain	small rain	clear	partly cloudy	Mainly cloudy

CHAPTER 5

MODULE WISE CODE

Index.js

```
import React from 'react';

import ReactDOM from 'react-dom';
import App from './App';


ReactDOM.render(<App />, document.getElementById('root'));
```

App.js

```
import React from "react";

import Dashboard from "./components/Dashboard";


function App() {
if (
process.env.REACT_APP_API_KEY_APPID === undefined ||
process.env.REACT_APP_API_KEY_UNSPLASH === undefined
) {
return (
<h4>
Please add .env and add follow REACT_APP_API_KEY_APPID &
REACT_APP_API_KEY_UNSPLASH.
</h4>
);
}
return (
<div className="App">
<Dashboard />
</div>
);
}

export default App;
```

Hooks

(useCoordinations.js)

```
import { useState } from "react";

export const useCoordinations = () => {
const [lat, setLat] = useState("");
const [long, setLong] = useState("");
const [loadingLocation, setLoadingLocation] = useState(false);

const findCoordinates = () => {
if (navigator.geolocation) {
setLoadingLocation(true);
navigator.geolocation.getCurrentPosition(
(pos) => {
console.log(pos);
setLat(pos.coords.latitude);
setLong(pos.coords.longitude);
},
(positionError) => {
setLat("27.390205");
setLong("77.154007");
console.log(positionError);
}
);
setLoadingLocation(false);
} else {
console.log("It's not supported by this browser.");
}
};

const updateLocation = (lat, long) => {
setLat(lat);
setLong(long);
//console.log('Updated Location')
};

return [{ lat, long }, loadingLocation, findCoordinates, updateLocation];
};
```

useImageFetch.js

```
import { useState, useEffect } from 'react';
import { SEARCH_DEFAULT } from '../api';

export const useImageFetch = () => {
const [image, setImage] = useState('');
const fecthImage = async endpoint => {
try {
const result = await (await fetch(endpoint)).json();
setImage(result.results[Math.floor(Math.random() * 10)].urls.regular);
//console.log(result.results);
} catch (error) {
console.log(error);
}
}
useEffect(() => {
```

```
fecthImage(`${SEARCH_DEFAULT}`);
}, []);

return [image, fecthImage];

}
```

useNightMode.js

```
import { useState, useEffect } from 'react';

export const useNightMode = () => {
//detect browser is using the dark mode
let checkTheme = window.matchMedia && window.matchMedia('(prefers-color-scheme:
dark)').matches;

const [nightMode, setNightMode] = useState(checkTheme);

const nightModeChanged = () => {
setNightMode(!nightMode);
localStorage.setItem('theme', !nightMode);
}

useEffect(() => {
if(localStorage.getItem('theme') === 'false'){
setNightMode(false);
}

}, []);
return [nightMode, nightModeChanged];

}
```

UseTempUnit.js

```
import { useState, useEffect } from 'react';

export const useTempUnit = () => {
const [unitMode, setUnitMode] = useState(false);

const unitModeChanged = () => {
setUnitMode(!unitMode);
localStorage.setItem('unit', !unitMode);
}

useEffect(() => {
if (JSON.parse(localStorage.getItem('unit'))) {
setUnitMode(!unitMode);
} else {
localStorage.setItem('unit', unitMode);
}
}, []);
return [unitMode, unitModeChanged];

}
```

useWeatherFetch.js

```
import { useState, useEffect } from 'react';
import { SEARCH_BY_LOCATION, DEFAULT_URL, API_URL_APPID, API_APPID,
GET_NEXT_DAYS_HOURS, DEF_N_D_H } from '../api';
```

```
export const useWeatherFetch = (searchCity, lat = '', long = '') => {
const [weather, setWeather] = useState('');
const [error, setError] = useState(false);
const [loading, setLoading] = useState(false);

const fetchWeather = async endpoint => {
try {
const result = await (await fetch(endpoint)).json();
return result;

} catch (error) {
console.log(error);
}
}

//searching location name
const searchByLocation = (searchCity) => {
if (searchCity) {
setLoading(true);
const search = searchCity.charAt(0).toUpperCase() + searchCity.slice(1);
fetchWeather(`${API_URL_APPID}/?q=${search}&APPID=${API_APPID}`).then((res)
=> {
//console.log('Getting api search');
if (res.cod !== '404') {
setWeather({ ...res, city: res.name, country: res.sys.country });
setError(false);
//fetch next days & hourly via lat and lon
const lat = res.coord.lat;
const long = res.coord.lon;
fetchWeather(`${GET_NEXT_DAYS_HOURS}&lat=${lat}&lon=${long}`).then((res) => {
//console.log('Getting api next days');
//console.log("Next days", res);
setWeather(prev => ({ ...prev, daily: res.daily, hourly: res.hourly, current: res.current }));
setLoading(false);
});
setLoading(false);
} else {
setError(true);
setLoading(false);
//console.log("error 404");
}

});
}
}

//get weather with lat and long
const getWeatherLocation = (lat, long) => {
setLoading(true);
if (lat && long) {
fetchWeather(`${SEARCH_BY_LOCATION}&lat=${lat}&lon=${long}`).then((res) => {
//console.log('Getting api');
setWeather({ ...res, city: res.name, country: res.sys.country });
});
fetchWeather(`${GET_NEXT_DAYS_HOURS}&lat=${lat}&lon=${long}`).then((res) => {
setWeather(prev => ({ ...prev, daily: res.daily, hourly: res.hourly, current: res.current }));
setLoading(false);
});
}

}


//*****/
useEffect(() => {
setLoading(true);
//default fetch...
fetchWeather(`${DEFAULT_URL}`).then((res) => {
```

```
setWeather({ ...res, city: res.name, country: res.sys.country });
});
//fetch next days
fetchWeather(`${DEF_N_D_H}`).then((res) => {
setWeather(prev => ({ ...prev, daily: res.daily, hourly: res.hourly, current: res.current }));
setLoading(false);
});


}, []);
return [weather, loading, error, fetchWeather, searchByLocation, getWeatherLocation];
}
```

Helper

```
export const convertF = (x) => {
return convertC(x)*1.8 + 32;
}

export const convertC = (x) =>{
return x - 273.15;
}
```

Color

```
export const themeLight = {
bgSidebar:'#FFFFFF',
textColor:'#000',
textLightColor:'#C5C5C5',
bgContent:'#F6F6F8',
bgBlue:'#4050D1',
bgOrange:'#FFC05D',
bgYellow:'#FFDB4B',
bgTemp: '#fff',
bgTempActive:"#000",
bgTempText: "#000",
bgTempTextActive:"#fff",
bgIcon: "#fff"
}

export const themeDark = {
bgSidebar:'#212121',
textColor:'#FFFFFF',
textLightColor:'#BEBEBE',
bgContent:'#292929',
bgBlue:'#092C6C',
bgOrange:'#FFC05D',
bgYellow:'#F5D47C',
bgTemp: '#000',
bgTempActive:"#fff",
bgTempText: "#fff",
bgTempTextActive:"#000",
bgIcon: "#fff"
}


import {themeLight, themeDark} from './colors';


export {
themeLight,
themeDark
};
```

API

```
//*** OPEN WEATHER & UNSPLASH KEYS****//

const API_APPID = process.env.REACT_APP_API_KEY_APPID;
const API_UNPSLASH = process.env.REACT_APP_API_KEY_UNSPLASH;
const API_URL_APPID = "https://api.openweathermap.org/data/2.5/weather";
const DEFAULT_URL =
`${API_URL_APPID}/?APPID=${API_APPID}&lat=41.390205&lon=2.154007`;
const SEARCH_BY_LOCATION = `${API_URL_APPID}?appid=${API_APPID}`;
//&lat={lat}&lon={lon}

const GET_NEXT_DAYS_HOURS =
`https://api.openweathermap.org/data/2.5/onecall?exclude=minutely&appid=${API_APPID}
`; //&lat={lat}&lon={lon}
const DEF_N_D_H =
`https://api.openweathermap.org/data/2.5/onecall?exclude=minutely&appid=${API_APPID}
&lat=41.390205&lon=2.154007`;

//*** UNSPLASH ****//
const URL_UNSPLASH = "https://api.unsplash.com/search/photos";
const SEARCH_BY_WORD =
`${URL_UNSPLASH}?client_id=${API_UNPSLASH}&page=1&query=`;
const SEARCH_DEFAULT =
`${URL_UNSPLASH}?client_id=${API_UNPSLASH}&page=1&query=Spain`;

export {
API_URL_APPID,
API_APPID,
DEFAULT_URL,
SEARCH_BY_LOCATION,
SEARCH_BY_WORD,
SEARCH_DEFAULT,
GET_NEXT_DAYS_HOURS,
DEF_N_D_H,
};
```

Components

Layout:-

Dashboard

```
import React, { useState, useEffect } from 'react';

import { ThemeProvider } from 'styled-components';
//include layout
import Header from './layouts/Header';
import Highlights from './layouts/Highlights';
import Sidebar from './layouts/Sidebar';
import Today from './layouts/Today';
import Week from './layouts/Week';
import Container from './layouts/Container';
import Spinner from './elements/Spinner';
import SpinnerContainer from './elements/SpinnerContainer';
import { StyledGlobal, StyledDashboard } from '../styles';
import { themeLight, themeDark } from '../constants';
import { useImageFetch } from '../hooks/useImageFetch';
import { useCoordinations } from '../hooks/useCoordinations';
import {useWeatherFetch} from '../hooks/useWeatherFetch';
import { useNightMode } from '../hooks/useNightMode';
import { SEARCH_BY_WORD } from '../api';
import { useTempUnit } from '../hooks/UseTempUnit';

const Dashboard = () => {
const [nightMode, nightModeChanged] = useNightMode();
const [unitMode, unitModeChanged] = useTempUnit();
const [image, fetchImage] = useImageFetch();
const [{ lat, long }, loadingLocation, findCoordinates] = useCoordinations();
const [
weather,
loading,
error,
fetchWeather,
searchByLocation,
getWeatherLocation,
] = useWeatherFetch();
const [showDays, setShowDays] = useState(false);
const fetchCoordinates = () => {
findCoordinates();
getWeatherLocation(lat, long);
};
const nightModeCallback = () => {
nightModeChanged();
};
const showDaysCallback = (enabled) => {
setShowDays(enabled);
};

const doSearchLocation = (searchTerm) => {
searchByLocation(searchTerm);
fetchImage(`${SEARCH_BY_WORD}${weather.city}`);
};
//onSlideChange={() => console.log('slide change')}
//onSwiper={(swiper) => console.log(swiper)}
const unitTempCallback = (enabled) => {
unitModeChanged(enabled);
};

//console.log("location", lat, long);
//console.log('Weather', weather);

useEffect(() => {
//default fetching..
getWeatherLocation(lat, long);
fetchImage(`${SEARCH_BY_WORD}${weather.city}`);
}, [lat, long]);
```

```
if (!weather)
return (
<ThemeProvider theme={nightMode ? themeDark : themeLight}>
<Spinner />
<StyledGlobal />
</ThemeProvider>
);

return (
<ThemeProvider theme={nightMode ? themeDark : themeLight}>
<StyledDashboard>
<Sidebar
findCoordinates={fetchCoordinates}
data={weather}
searchCallback={doSearchLocation}
error={error}
image={image}
titleLocation={weather}
unitTemp={unitMode}
/>
<Container>
<Header
unitMode={unitMode}
unitTempCallback={unitTempCallback}
nightModeCallback={nightModeCallback}
nightMode={nightMode}
showDaysCallback={showDaysCallback}
showActive={showDays}
/>
{loading || loadingLocation ? (
<SpinnerContainer />
) : (
<>
{!showDays ? (
<Week data={weather.daily} tempUnit={unitMode} />
) : (
<Today tempUnit={unitMode} data={weather.hourly} />
)}
<Highlights data={weather.current} />
</>
)}
</Container>
<StyledGlobal />
</StyledDashboard>
</ThemeProvider>
);
}
export default Dashboard;
```

Elements:

LocationBox

```
import React from 'react'


const LocationBox = ({ image, titleLocation }) => (
<div className="location">
<div className="location_inner">
<img src={image} width="100%" alt = {titleLocation}/>
<span>{titleLocation}</span>
</div>
</div>


);
export default LocationBox;
```

Search Bar

```
import React, { useState } from 'react'
import { MdGpsFixed } from 'react-icons/md';
import { BiSearch } from 'react-icons/bi';

const SearchBar = ({ findCoordinates, searchCallback, error }) => {
const [searchTerm, setSearchTerm] = useState('');
const doChangeInput = (e) => {
const { value } = e.target;
setSearchTerm(value);
}
const doSearch = () => {
searchCallback(searchTerm);
setSearchTerm("");
}
return (
<div className="search_bar">
<div className="search_content">
<div className="search-color"><BiSearch /></div>
<div className="search__box">
<input type="text"
placeholder="Search for places ..."
onChange={doChangeInput}
onKeyPress={e => e.key === 'Enter' && doSearch()}
value={searchTerm} />
</div>
<div className="gray-border" onClick={findCoordinates}><MdGpsFixed /></div>
</div>
{error && (<div className="error">Please type correct!</div>)}
</div>);

};
export default SearchBar;
```

Spinner

```
import React from 'react';
import { StyledSpinner } from '../../styles';
const Spinner = () => (
<StyledSpinner>
<svg width="200" height="150" viewBox="0 0 800 600" fill="none"
xmlns="http://www.w3.org/2000/svg" data-reactroot="">
<path d="M667.214 299.996C667.214 346.785 655.45 390.834 634.705 429.331C634.63
429.451 634.57 429.586 634.495 429.706C588.335 515.095 497.991 573.11 394.101
573.11C243.273 573.11 120.988 450.84 120.988 299.996C120.988 284.685 122.246 269.673
124.67 255.034C124.76 254.45 124.865 253.851 124.97 253.268C146.029 131.133 248.213
36.7618 374.045 27.6322C380.66 27.1382 387.351 26.8984 394.101 26.8984C476.528
26.8984 550.423 63.4045 600.489 121.134C642.069 169.06 667.214 231.594 667.214
299.996Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path fillRule="evenodd" clipRule="evenodd" d="M394.105 11C386.961 11 379.882 11.2537
372.883 11.7764L372.881 11.7765C239.727 21.4373 131.596 121.301 109.311
250.543C106.539 266.619 105.098 283.139 105.098 299.992C105.098 459.614 234.499 589
394.105 589C504.199 589 599.928 527.437 648.712 436.853L649.593 437.327L648.712
436.853C670.664 396.117 683.113 349.506 683.113 299.992C683.113 227.609 656.505
161.436 612.505 110.72L613.221 110.099L612.505 110.72C559.523 49.6293 481.329 11
394.105 11ZM372.735 9.78185C379.785 9.25536 386.914 9 394.105 9C481.933 9 560.67
47.8994 614.016 109.41L613.26 110.065L614.016 109.41C658.319 160.476 685.113 227.11
685.113 299.992C685.113 349.845 672.578 396.781 650.473 437.802C601.354 529.007
504.963 591 394.105 591C233.395 591 103.098 460.719 103.098 299.992C103.098 283.025
104.549 266.391 107.34 250.203L108.324 250.373L107.34 250.203C129.779 120.064
238.656 19.5103 372.735 9.78185Z" fill="#B5CDFB"></path>
<path className="fill" d="M634.495 429.706C588.335 515.096 497.991 573.111 394.101
573.111C243.273 573.111 120.988 450.841 120.988 299.997C120.988 284.685 122.246
269.673 124.67 255.035C125.434 255.184 126.167 255.394 126.93 255.633C126.93 255.633
188.627 215.565 255.906 274.388C323.185 333.195 308.816 497.569 394.715
468.637C480.614 439.69 496.839 424.647 530.411 426.952C563.968 429.272 594.442
436.217 600.04 417.687C604.8 401.971 627.805 422.746 634.495 429.706Z"
fill="#000000"></path>
```

```
<path className="fill" d="M581.966 299.891C602.187 320.651 594.644 355.286 567.627 365.778C554.156 371.002 541.045 375.283 531.93 376.42C504.419 379.863 476.894 390.191 452.826 335.723C428.743 281.272 395.5 324.842 367.99 298.5C340.465 272.171 417.278 219.366 427.77 117.332C438.263 15.2986 394.348 96.692 377.853 94.4015C367.197 92.9201 370.31 54.603 373.857 27.6614C373.962 27.6468 374.052 27.6315 374.157 27.6315C380.637 27.1374 387.358 26.8984 394.108 26.8984C472.763 26.8984 547.451 60.9343 599.089 120.31C581.322 128.767 563.257 134.784 552.57 138.885C526.197 148.989 526.197 207.901 522.919 225.099C521.063 234.828 553.483 270.645 581.966 299.891Z" fill="#000000"></path>
<path d="M200.037 92.899C200.281 91.2473 200.448 89.5713 200.448 87.8528C200.448 68.8544 185.047 53.4531 166.049 53.4531H153.278C134.28 53.4531 118.879 68.8544 118.879 87.8528H98.3994C79.4012 87.8528 64 103.254 64 122.252C64 141.251 79.4012 156.652 98.3994 156.652H182.169C201.168 156.652 216.569 141.251 216.569 122.252C216.569 109.807 209.937 98.9383 200.037 92.899Z" fill="#B5CDFB"></path>
<path d="M171.135 69.3307C171.821 69.3307 172.377 68.7746 172.377 68.0887C172.377 67.4027 171.821 66.8467 171.135 66.8467C170.449 66.8467 169.893 67.4027 169.893 68.0887C169.893 68.7746 170.449 69.3307 171.135 69.3307Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M181.406 76.4518C182.092 76.4518 182.648 75.8957 182.648 75.2098C182.648 74.5238 182.092 73.9678 181.406 73.9678C180.72 73.9678 180.164 74.5238 180.164 75.2098C180.164 75.8957 180.72 76.4518 181.406 76.4518Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M171.754 80.4644C172.44 80.4644 172.996 79.9084 172.996 79.2225C172.996 78.5365 172.44 77.9805 171.754 77.9805C171.068 77.9805 170.512 78.5365 170.512 79.2225C170.512 79.9084 171.068 80.4644 171.754 80.4644Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M203.762 136.591C204.447 136.591 205.004 136.035 205.004 135.349C205.004 134.663 204.447 134.107 203.762 134.107C203.076 134.107 202.52 134.663 202.52 135.349C202.52 136.035 203.076 136.591 203.762 136.591Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M190.406 114.713C191.092 114.713 191.648 114.157 191.648 113.471C191.648 112.786 191.092 112.229 190.406 112.229C189.72 112.229 189.164 112.786 189.164 113.471C189.164 114.157 189.72 114.713 190.406 114.713Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
```

<path d="M183.4 99.9029C184.086 99.9029 184.642 99.3469 184.642 98.6609C184.642 97.975 184.086 97.4189 183.4 97.4189C182.714 97.4189 182.158 97.975 182.158 98.6609C182.158 99.3469 182.714 99.9029 183.4 99.9029Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M190.406 130.329C191.092 130.329 191.648 129.773 191.648 129.087C191.648 128.401 191.092 127.845 190.406 127.845C189.72 127.845 189.164 128.401 189.164 129.087C189.164 129.773 189.72 130.329 190.406 130.329Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M197.098 106.978C198.14 106.978 198.985 106.133 198.985 105.091C198.985 104.049 198.14 103.204 197.098 103.204C196.056 103.204 195.211 104.049 195.211 105.091C195.211 106.133 196.056 106.978 197.098 106.978Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M203.76 121.53C204.802 121.53 205.647 120.685 205.647 119.643C205.647 118.601 204.802 117.756 203.76 117.756C202.718 117.756 201.873 118.601 201.873 119.643C201.873 120.685 202.718 121.53 203.76 121.53Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M177.68 147.939C178.722 147.939 179.567 147.094 179.567 146.052C179.567 145.01 178.722 144.165 177.68 144.165C176.638 144.165 175.793 145.01 175.793 146.052C175.793 147.094 176.638 147.939 177.68 147.939Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M187.276 86.723C188.318 86.723 189.162 85.8782 189.162 84.8361C189.162 83.794 188.318 82.9492 187.276 82.9492C186.233 82.9492 185.389 83.794 185.389 84.8361C185.389 85.8782 186.233 86.723 187.276 86.723Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M194.326 146.056C195.368 146.056 196.213 145.211 196.213 144.169C196.213 143.127 195.368 142.282 194.326 142.282C193.284 142.282 192.439 143.127 192.439 144.169C192.439 145.211 193.284 146.056 194.326 146.056Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M703.625 376.271H626.957C609.57 376.271 595.475 362.176 595.475 344.788V344.788C595.475 327.401 609.57 313.306 626.957 313.306H703.625C721.012 313.306 735.107 327.401 735.107 344.788V344.788C735.107 362.176 721.012 376.271 703.625 376.271Z" fill="#B5CDFB"></path>
<path d="M677.02 344.788H665.332C647.945 344.788 633.85 330.692 633.85 313.305V313.305C633.85 295.918 647.945 281.822 665.332 281.822H677.02C694.408

281.822 708.503 295.918 708.503 313.305V313.305C708.503 330.692 694.408 344.788 677.02 344.788Z" fill="#B5CDFB"></path>
<path d="M682.765 292.521C683.451 292.521 684.007 291.965 684.007 291.279C684.007 290.593 683.451 290.037 682.765 290.037C682.079 290.037 681.523 290.593 681.523 291.279C681.523 291.965 682.079 292.521 682.765 292.521Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M693.037 299.632C693.723 299.632 694.279 299.076 694.279 298.39C694.279 297.704 693.723 297.148 693.037 297.148C692.351 297.148 691.795 297.704 691.795 298.39C691.795 299.076 692.351 299.632 693.037 299.632Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M683.385 303.645C684.07 303.645 684.627 303.089 684.627 302.403C684.627 301.717 684.07 301.161 683.385 301.161C682.699 301.161 682.143 301.717 682.143 302.403C682.143 303.089 682.699 303.645 683.385 303.645Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M724.99 359.778C725.676 359.778 726.232 359.222 726.232 358.536C726.232 357.85 725.676 357.294 724.99 357.294C724.304 357.294 723.748 357.85 723.748 358.536C723.748 359.222 724.304 359.778 724.99 359.778Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M711.631 337.899C712.317 337.899 712.873 337.343 712.873 336.657C712.873 335.971 712.317 335.415 711.631 335.415C710.945 335.415 710.389 335.971 710.389 336.657C710.389 337.343 710.945 337.899 711.631 337.899Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M704.625 323.095C705.311 323.095 705.867 322.539 705.867 321.853C705.867 321.167 705.311 320.611 704.625 320.611C703.939 320.611 703.383 321.167 703.383 321.853C703.383 322.539 703.939 323.095 704.625 323.095Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M711.631 353.523C712.317 353.523 712.873 352.967 712.873 352.281C712.873 351.595 712.317 351.039 711.631 351.039C710.945 351.039 710.389 351.595 710.389 352.281C710.389 352.967 710.945 353.523 711.631 353.523Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M718.324 330.158C719.366 330.158 720.211 329.314 720.211 328.272C720.211 327.23 719.366 326.385 718.324 326.385C717.282 326.385 716.438 327.23 716.438 328.272C716.438 329.314 717.282 330.158 718.324 330.158Z" fill="none" stroke="#221b38" strokeWidth="1"></path>

```
<path d="M724.988 344.71C726.031 344.71 726.875 343.865 726.875 342.823C726.875 341.781 726.031 340.937 724.988 340.937C723.946 340.937 723.102 341.781 723.102 342.823C723.102 343.865 723.946 344.71 724.988 344.71Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M698.906 371.118C699.949 371.118 700.793 370.274 700.793 369.232C700.793 368.19 699.949 367.345 698.906 367.345C697.864 367.345 697.02 368.19 697.02 369.232C697.02 370.274 697.864 371.118 698.906 371.118Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M698.906 309.91C699.949 309.91 700.793 309.066 700.793 308.024C700.793 306.981 699.949 306.137 698.906 306.137C697.864 306.137 697.02 306.981 697.02 308.024C697.02 309.066 697.864 309.91 698.906 309.91Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
<path d="M715.553 369.237C716.595 369.237 717.44 368.392 717.44 367.35C717.44 366.308 716.595 365.463 715.553 365.463C714.511 365.463 713.666 366.308 713.666 367.35C713.666 368.392 714.511 369.237 715.553 369.237Z" fill="none" stroke="#221b38" strokeWidth="1"></path>
</svg>
<span>Loading...</span>
</StyledSpinner>
)
export default Spinner;
```

Spinner Container

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

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

Weather Icon

```
import React from 'react';

const WeatherIcon = ({icon}) =>(
<div className = "icon_weather">
<img src = {require('../../images/v2/' + icon + '.png')} width = "150" alt = "weather"/>
</div>
);

export default WeatherIcon;
```

Weather Info

```
import React from 'react';

import { convertC, convertF } from '../../helpers';

const WeatherInfo = ({ data,unit }) => (
<>
<div className="temperature">
{unit ? convertF(data.main.temp).toFixed(0) : convertC(data.main.temp).toFixed(0)}<span
className="degree">°</span> <span className="unit">{unit ? "F": "C"}</span>
</div>
<div className="date">
<span className="day">{new Date().toLocaleString('en-US', { weekday: 'long' })}</span>,
<span className="hours">{new Date().toLocaleString('en-US', { hour: 'numeric', minute:
'numeric', hour12: true })}</span>
</div>
<hr />
<div className="info-weather">
<div>
<svg width="24" height="24" viewBox="0 0 24 24" fill="none"
xmlns="http://www.w3.org/2000/svg" data-reactroot="">
```

```
<path strokeLinejoin="round" strokeLinecap="round" strokeMiterlimit="10" strokeWidth="1" stroke="#293449" fill="none" d="M9 14.5954V3.99927C9 2.89967 8.1 2 7 2C5.9 2 5 2.89967 5 3.99927V14.5954C3.5 15.3951 2.7 17.1944 3.1 18.9938C3.4 20.3933 4.6 21.5928 6 21.8927C8.6 22.4925 11 20.4932 11 17.9941C11 16.4947 10.2 15.1952 9 14.5954Z"></path>
<path strokeLinejoin="round" strokeLinecap="round" strokeMiterlimit="10" strokeWidth="1" stroke="#293449" fill="none" d="M7 18.4943C7.27614 18.4943 7.5 18.2705 7.5 17.9943C7.5 17.7182 7.27614 17.4943 7 17.4943C6.72386 17.4943 6.5 17.7182 6.5 17.9943C6.5 18.2705 6.72386 18.4943 7 18.4943Z"></path>
<path fill="none" d="M13 7C14.6452 7 16 8.35484 16 10C16 11.6452 14.6452 13 13 13" undefined="1"></path>
<path strokeLinejoin="round" strokeLinecap="round" strokeMiterlimit="10" strokeWidth="1" stroke="#293449" d="M13 7C14.6452 7 16 8.35484 16 10C16 11.6452 14.6452 13 13 13"></path>
<path strokeLinejoin="round" strokeLinecap="round" strokeMiterlimit="10" strokeWidth="1" stroke="#293449" d="M13 3.6V2"></path>
<path strokeLinejoin="round" strokeLinecap="round" strokeMiterlimit="10" strokeWidth="1" stroke="#293449" d="M18.7 15.6954L17.6 14.5954"></path>
<path strokeLinejoin="round" strokeLinecap="round" strokeMiterlimit="10" strokeWidth="1" stroke="#293449" d="M17.6 5.39999L18.7 4.29999"></path>
<path strokeLinejoin="round" strokeLinecap="round" strokeMiterlimit="10" strokeWidth="1" stroke="#293449" d="M19.4 9.99707H21"></path>
</svg>
<span>Feels like {unit ? convertF(data.main.feels_like).toFixed(0) : convertC(data.main.feels_like).toFixed(0)} ° {unit ? "F": "C"}</span>

</div>
{data.clouds && (<div>
<svg width="24" height="24" viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg" data-reactroot="">
<path strokeLinejoin="round" strokeLinecap="round" strokeMiterlimit="10" strokeWidth="1" stroke="#293449" fill="none" d="M18.1 9.7C17.3 7.6 15.2 6 12.7 6C9.5 6 6.9 8.5 6.9 11.7C6.9 11.8 6.9 11.9 6.9 12C6.4 11.7 5.9 11.6 5.3 11.6C3.5 11.5 2 13 2 14.8C2 16.6 3.5 18 5.3 18H17.9C20.2 18 22 16 22 13.8C22 11.6 20.3 9.9 18.1 9.7Z"></path>
</svg>
<span>Cloudly - {data.clouds.all}%</span>
</div>)}
</div>
</>
);
export default WeatherInfo;
```

Layout :

Container

```
import React from 'react';

import { StyledContainer } from './../../styles';

const Container = ({ children }) => (
<StyledContainer>
{children}
</StyledContainer>
);

export default Container;
```

Header

```
import React from 'react';

import { StyledHeader } from './../../styles';
import { IoIosSunny } from 'react-icons/io';
import { RiMoonClearFill } from 'react-icons/ri';

const Header = ({ nightMode, nightModeCallback, showDaysCallback, showActive,
unitTempCallback,unitMode }) => {
```

```
const enableToday = (enabled) => {
showDaysCallback(enabled);
}
const changedTemp = (enabled) =>{
unitTempCallback(enabled);
}
return (
<StyledHeader>
<div className="today-week">
<span className={`today ${showActive ? "active" : ""}`} onClick={() =>
enableToday(true)}>Today</span>
<span className={`week ${showActive ? "" : "active"}`} onClick={() =>
enableToday(false)}>Week</span>
</div>
<div className="temperature">
<div className={`celsius ${unitMode ? "" : "active"}`} onClick={() =>
changedTemp(false)}>
<span className="degree">°</span> C
</div>
<div className={`fahrenheit ${unitMode ? "active" : ""}`} onClick={() =>
changedTemp(true)}>
<span className="degree">°</span> F
</div>
</div>
<div className="toggle-theme">
<input type="checkbox" className="checkbox" id="chk" checked={nightMode}
onChange={nightModeCallback} />
<label className="label" htmlFor="chk">
<div className="sun"><IoIosSunny size={15} /></div>
<div className="moon"><RiMoonClearFill size={12} /></div>
<div className="ball"></div>
</label>
</div>
</StyledHeader>
);
}

export default Header;
```

Highlight

```
import React from 'react';

import { StyledHighlights } from './../../styles';
import { FaMapMarkerAlt } from 'react-icons/fa';
import { FiArrowDown, FiArrowUp } from 'react-icons/fi';
import { WiHumidity, WiBarometer } from 'react-icons/wi';
import { MdVisibility } from 'react-icons/md';
import { GiSunRadiations } from 'react-icons/gi';
const Highlights = ({ data }) => (
<StyledHighlights>
{data && <div className="highlights_info">
<h1>Today's Highlights</h1>
<div className="highlights_inner">
<div className="box_info">
<span className="type-info">UV Index</span>
<div className="flex-box">
<span className="result-uv"><GiSunRadiations /></span>
<span className="info-text"><span>{Math.round(data.uvi)}</span></span>
</div>
</div>
<div className="box_info">
<span className="type-info">Wind Status</span>
<div className="info-text"><span>{data.wind_speed}</span> km/s</div>
<div className="ssw">
<div className="icon-ssw"><FaMapMarkerAlt /></div>
SSW
</div>
</div>
<div className="box_info">
<span className="type-info">Sunrise & Sunset</span>
<div className="sunrise icon"><span><FiArrowUp /></span>{(new Date(data.sunrise *
1000)).toLocaleString('en-US', { hour: 'numeric', minute: 'numeric', hour12: true })}</div>
<div className="sunset icon"><span><FiArrowDown /></span>{(new Date(data.sunset *
1000)).toLocaleString('en-US', { hour: 'numeric', minute: 'numeric', hour12: true })}</div>
</div>
<div className="box_info">
<span className="type-info">Humidity</span>
<div className="flex-box">
<span className="icon-h"><WiHumidity /></span>
<span className="info-text"><span>{data.humidity}</span>% </span>
</div></div>
<div className="box_info">
<span className="type-info">Pressure</span>
<div className="flex-box">
<span className="icon-p"><WiBarometer /></span>
<span className="info-text"><span>{data.pressure}</span>hPa </span></div>
</div>
<div className="box_info">
<span className="type-info">Visibility</span>
<div className="flex-box">
<span className="icon-v"><MdVisibility /></span>
<span className="info-text"><span>{(data.visibility / 1000).toFixed(1)}</span>km </span>
</div></div>
</div>
</div>}
</StyledHighlights>
);

export default Highlights;
```

Sidebar

```
import React from "react";

import SearchBar from "../elements/searchBar";
import WeatherIcon from "../elements/WeatherIcon";
import LocationBox from "../elements/LocationBox";
import WeatherInfo from "../elements/WeatherInfo";
import { StyledSidebar } from "./../../styles";
import { countries } from "country-data";

const Sidebar = ({
findCoordinates,
searchCallback,
image,
titleLocation,
error,
data,
unitTemp,
}) => {
return (
<StyledSidebar>
<SearchBar
findCoordinates={findCoordinates}
searchCallback={searchCallback}
error={error}
/>
{data.weather[0] && <WeatherIcon icon={data.weather[0].icon} />}
<WeatherInfo data={data} unit={unitTemp} />
{titleLocation.country && (
<LocationBox
image={image}
titleLocation={
titleLocation.city + ", " + countries[titleLocation.country].name
}
/>
)}
</StyledSidebar>
);
};

export default Sidebar;
```

Today

```
import React from 'react';
import { StyledToday } from '../../styles';
import { convertC, convertF } from '../../helpers';
// Import Swiper React components
import { Swiper, SwiperSlide } from 'swiper/react';

// Import Swiper styles
import 'swiper/swiper.scss';

const Today = ({ data,tempUnit }) => (
<StyledToday>
<Swiper
spaceBetween={20}
slidesPerView={2}
breakpoints={{
// when window width is >= 640px
640: {
width: 640,
slidesPerView: 4,
},
// when window width is >= 768px
768: {
width: 768,
slidesPerView: 5,
},
// when window width is >= 991px
991: {
width: 991,
slidesPerView: 6,
},
// when window width is >= 1024px
1024: {
width: 1024,
slidesPerView: 6,
},
}}
>
{data && data.map((item, i) => (
<SwiperSlide key={i.toString()}>
<div className="box_info" >
<div>{(new Date(item.dt * 1000)).toLocaleString('en-US', { hour: 'numeric', minute:
'numeric', hour12: true })}</div>
<img src={require('../../images/v2/' + item.weather[0].icon + '.png')} alt =
{item.weather[0].description}/>
<div className="temp_info">
<span>{tempUnit ? convertF(item.temp).toFixed(0) : convertC(item.temp).toFixed(0)}°
{tempUnit ? "F": "C"}</span>
</div>
</div>
</SwiperSlide>))}
</Swiper>
</StyledToday>);

export default Today;
```

Week

```
import React from 'react';
import { StyledWeek } from '../../styles';
import { convertC, convertF } from '../../helpers';
// Import Swiper React components
import { Swiper, SwiperSlide } from 'swiper/react';
// Import Swiper styles
import 'swiper/swiper.scss';

const Week = ({ data,tempUnit }) => (
<StyledWeek>
<Swiper
spaceBetween={20}
slidesPerView={2}
breakpoints={{
// when window width is >= 640px
640: {
width: 640,
slidesPerView: 4,
},
// when window width is >= 768px
768: {
width: 768,
slidesPerView: 5,
},
// when window width is >= 991px
991: {
width: 991,
slidesPerView: 6,
},
// when window width is >= 1024px
1024: {
width: 1024,
slidesPerView: 6,
},
}}
//onSlideChange={() => console.log('slide change')}
//onSwiper={(swiper) => console.log(swiper)}
>
{data && data.map((day, i) => <SwiperSlide key={i.toString()}><div
className="box_info">
<div>{new Date(day.dt * 1000).toLocaleString('en-US', { weekday: 'short' })}</div>
<img src={require('../../images/v2/' + day.weather[0].icon + '.png')} alt =
{day.weather[0].description}/>
<div className="temp_info">
<span>{tempUnit ? convertF(day.temp.max).toFixed(0) :
convertC(day.temp.max).toFixed(0)}° </span>
<span> - </span>
<span className="light-text">{tempUnit ? convertF(day.temp.min).toFixed(0) :
convertC(day.temp.min).toFixed(0)}° {tempUnit ? "F": "C"}</span>
</div>
</div></SwiperSlide>)}
</Swiper></StyledWeek>);
export default Week;
```

.ENV

```
REACT_APP_API_KEY_APPID = 98d840973172acd7810fded3c47cbbe2
REACT_APP_API_KEY_UNSPLASH =
JDB7sGtPE6lcHWxW03K2vLIxdcpz7_8EH7UbuKPR564
```

STYLE

INDEXJS

```
import StyledGlobal from './StyledGlobal';
import StyledHeader from './StyledHeader';
import StyledContainer from './StyledContainer';
import StyledHighlights from './StyledHighlights';
import StyledSidebar from './StyledSidebar';
import StyledToday from './StyledToday';
import StyledWeek from './StyledWeek';
import StyledDashboard from './StyledDashboard';
import StyledSpinner from './StyledSpinner';
import StyledSpinnerContainer from './StyledSpinnerContainer';

export {
StyledGlobal,
StyledHeader,
StyledContainer,
StyledHighlights,
StyledSidebar,
StyledWeek,
StyledToday,
StyledDashboard,
StyledSpinner,
StyledSpinnerContainer
};
```

Style CONTAINER

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

const StyledContainer = styled.div`
@media (min-width: 991px){
float:left;
width:calc(100% - 300px);
}
@media (min-width: 1700px){
width:calc(100% - 400px);
}
background:${props => props.theme.bgContent};
```

```
padding: 30px;
`;
export default StyledContainer;
```

Styled Dashboard

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

const StyledDashboard = styled.div`
`;
export default StyledDashboard;
```

STYLEGLOBAL

```
import { createGlobalStyle } from 'styled-components';

const GlobalStyle = createGlobalStyle`
*{
box-sizing:border-box;
}
body{
font-size:20px;
font-family: "AvertaStd-Thin";
background:${props => props.theme.bgContent};
margin:0;
padding:0;
line-height:1;
}`;
export default GlobalStyle;
```

Style Header

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

```
const StyledHeader = styled.div`
display: flex;
flex-direction: row;
justify-content: space-between;
.today-week{
font-family: "Averta-Regular";
span{
cursor:pointer;
margin-right:20px;
color:${props => props.theme.textLightColor};
}
.active{
color:${props => props.theme.textColor};
border-bottom: 1px solid;
padding-bottom: 5px;
}
}
.temperature{
display:flex;
justify-content: space-between;
max-width:90px;
width:100%;
div{
font-family: "Averta-Regular";
background:${props => props.theme.bgTemp};
border-radius:30px;
width:39px;
height:39px;
color:${props => props.theme.bgTempText};
justify-content: center;
align-items: center;
display: flex;
cursor:pointer;
}
.active{
background:${props => props.theme.bgTempActive};
color:${props => props.theme.bgTempTextActive};
}
}
.checkbox {
opacity: 0;
position: absolute;
}
.label {
background-color: #111;
border-radius: 50px;
cursor: pointer;
display: flex;
align-items: center;
justify-content: space-between;
padding: 5px;
position: relative;
height: 26px;
width: 50px;
margin-right:15px;
transform: scale(1.5);
}
.label .ball {
background-color: #fff;
border-radius: 50%;
position: absolute;
top: 2px;
left: 2px;
height: 22px;
width: 22px;
transform: translateX(0px);
transition: transform 0.2s linear;
```

```
}
.checkbox:checked + .label .ball {
transform: translateX(24px);
}
.sun{
padding-top:2px;
}
.moon,.sun{
color:${props => props.theme.bgIcon};
}
.toggle-theme{
@media(max-width:768px){
position: absolute;
top: 30px;
right: 20px;
}
}
`;
export default StyledHeader;
```

<u>Style Highlights</u>import styled from 'styled-components';

```
const StyledHighlights = styled.div`

h1{
font-size:21px;
font-weight:100;
margin-top:30px;
font-family: "Averta-Regular";
color:${props => props.theme.textColor};

}
.highlights_inner{
```

```
color:${props => props.theme.textColor};
display:flex;
justify-content: space-between;
margin-top: 20px;
text-align: center;
flex-wrap: wrap;
flex-direction:row;
@media(min-width:1700px){
height: calc(100vh - (400px));
margin-top:60px;
}
.box_info{
position:relative;
background:${props => props.theme.bgSidebar};
border-radius:20px;
padding: 20px;
@media (min-width: 768px){
margin-left:1%;
}

text-align:left;
font-size:17px;
display: flex;
justify-content: center;
flex-direction: column;
min-height: 158px;
margin-bottom: 20px;
flex-basis: 30.333333%;
-webkit-box-shadow: -5px 9px 10px -7px rgba(0,0,0,0.10);
-moz-box-shadow: -5px 9px 10px -7px rgba(0,0,0,0.10);
box-shadow: -5px 9px 10px -7px rgba(0,0,0,0.10);
@media (max-width: 991px){
flex-basis: 48%;
}
```

```
@media(min-width:1700px){
max-height: 274px;
}
.type-info{
color:${props => props.theme.textLightColor};
font-size:15px;
position: absolute;
top: 20px;
left: 20px;
font-weight:bold;
@media(min-width:1700px){
font-size:20px;
}
}
.info-text{
line-height:57px;
}
.info-text span{
font-size:40px;
font-weight:bold;
@media(max-width:620px){
font-size:30px;
}
}
.emoticons{
font-size: 16px;
}
.icon-ssw{
display: inline-block;
margin-right: 10px;
background-color: #4d4df3;
width: 25px;
height: 25px;
color: #fff;
```

```
padding:5px;
border-radius: 50px;
font-size: 15px;
}
.icon{
margin:10px 0;
}
.icon span{
display:inline-block;
margin-right: 10px;
background-color: #f9e241;
border:2px solid #ffd254;
width: 30px;
height: 30px;
padding:5px;
color:#fff;
border-radius: 50px;
font-size: 15px;

}
.sunrise{
margin-top:30px;
}
.icon-h{
display: inline-block;
margin-right: 10px;
background-color: #27a51f;
border: 2px solid #1d650b;
width: 30px;
height: 30px;
padding: 2px;
color: #fff;
border-radius: 50px;
font-size: 21px;
```

```
}
.icon-v{
display: inline-block;
margin-right: 10px;
background-color: #8a1fa5;
border: 2px solid #4c0ba2;
width: 30px;
height: 30px;
padding: 4px;
color: #fff;
border-radius: 50px;
font-size: 18px;
@media(max-width:480px){
position: absolute;
top: 15px;
right: 8px;
}
}
.result-uv{
display: inline-block;
margin-right: 10px;
background-color: #a5881f;
border: 2px solid #c5a713;
width: 30px;
height: 30px;
padding: 4px;
color: #fff;
border-radius: 50px;
font-size: 18px;

}
.icon-p{
margin-right: 10px;
background-color: #a51f1f;
```

```
border: 2px solid #a20b0b;
display: inline-block;
width: 30px;
height: 30px;
padding: 2px;
color: #fff;
border-radius: 50px;
font-size: 21px;
@media(max-width:480px){
position: absolute;
top: 15px;
right: 8px;
}
}
}
.flex-box{
margin-top:25px;
}

}
`;
export default StyledHighlights;
```

Style Sidebar

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


const StyledSidebar = styled.div`
@media (min-width: 991px){
max-width:300px;
width:100%;
float:left;
display: flex;
```

```
flex-direction: column;
justify-content: space-between;
height: 100vh;
}
@media (min-width: 1700px){
max-width:400px;
}
background:${props => props.theme.bgSidebar};
padding:30px;
/*justify-content: center;
align-items: center;
display: flex;*/

.search_content{
display:flex;
display: flex;
justify-content: center;
align-items: center;
@media(max-width:768px){
margin-top:65px;
}

input{
padding-left:10px;
border:0;
font-size:18px;
@media(max-width:768px){
font-size: 22px;
}
background:transparent;
color:${props => props.theme.textColor};
font-family: "AvertaStd-Thin";
margin-right:10px;
:focus,:active{
```

```
outline:none;
}
::placeholder{
color:${props => props.theme.textColor};
opacity:1;
font-size: 18px;
@media(max-width:768px){
font-size: 22px;
}
}

}
.gray-border{
color:${props => props.theme.textColor};
border-radius:50px;
height:30px;
width:30px;
padding:5px;
background:${props => props.theme.bgContent};
cursor:pointer;
@media(max-width:480px){
height: 45px;
width: 45px;
padding: 9px;
font-size: 25px;
flex:1;
}
}
.search-color{
color:${props => props.theme.textColor};
@media(max-width:480px){
font-size:28px;
flex:1;
}
```

```
}
}
.search__box{
@media(max-width:480px){
flex:14;
}
}
.icon_weather{
margin:30px 0;
@media(max-width:480px){
margin:10px 0;
}
}
.temperature{
padding-bottom:30px;
font-size:75px;
font-weight: 300;
color:${props => props.theme.textColor};
span{
position: relative;
top: -17px;
}
.degree{
font-size: 40px;
padding-left:5px;
font-weight: 700;
}
.unit{
top: -22px;
font-size: 35px;
margin-left:-15px;
font-weight: 900;
}
}
```

```
.error{
font-size:16px;
color:red;
}

.day{
font-size:18px;
color:${props => props.theme.textColor};
font-weight:900;
}
.hours{
font-size:20px;
color:${props => props.theme.textLightColor};
padding-left:5px;
font-weight:bold;
}
hr{
border-width: 0px;
border-color: transparent;
background:${props => props.theme.bgContent};
height:1.5px;
width:100%;
margin:30px 0;
}
.info-weather{
font-size:14px;
@media(max-width:480px){
font-size:20px;
}
color:${props => props.theme.textColor};
line-height:30px;
font-weight:900;
div{
display: flex;
```

```
justify-content: flex-start;
align-items: center;
span{
margin-left:10px;
}
path{
stroke:${props => props.theme.textColor};
}
}
}
.location{
margin:15% 0;
@media(max-width:480px){
margin:9% 0;
}
.location_inner{
position:relative;
border-radius:20px;
overflow:hidden;
height:100px;
width:100%;
:before{
content:'';
background: rgba(0,0,0,0.35);
position:absolute;
left:0;
top:0;
right:0;
bottom:0;
height:100%;
width:100%;
}
img{
object-fit: cover;
object-position: center -20px;
}
span{
position:absolute;
left:50%;
top:50%;
transform: translate(-50%,-50%);
-webkit-transform: translate(-50%,-50%);
-moz-transform: translate(-50%,-50%);
color:#fff;
font-size:17px;
@media(max-width:480px){
font-size:20px;
}
font-family:"Averta-Regular";
width:100%;
text-align:center;
}
}
}
`;
export default StyledSidebar;
```

Styled Spinner

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


const StyledSpinner = styled.div`
text-align:center;
display:flex;
justify-content:center;
align-items:center;
flex-direction: column;
```

```
height: 100vh;
span{
font-size:17px;
padding-top:20px;
color:${props => props.theme.textColor};
font-family: "Averta-Regular";
font-style:italic;
}
.fill{
fill:${props => props.theme.textColor};
}
`;

export default StyledSpinner;
```

Styled Spinner Container

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


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

export default StyledSpinnerContainer;
```

Styled Today

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


const StyledToday = styled.div`
color:${props => props.theme.textColor};
margin-top: 20px;
.box_info{
text-align: center;
display:flex;
justify-content:center;
background:${props => props.theme.bgSidebar};
color:${props => props.theme.textColor};
border-radius:20px;
padding: 15px;
width: 100%;
font-size:17px;
display: flex;
justify-content: center;
flex-direction: column;
-webkit-box-shadow: -5px 9px 10px -7px rgba(0,0,0,0.10);
-moz-box-shadow: -5px 9px 10px -7px rgba(0,0,0,0.10);
box-shadow: -5px 9px 10px -7px rgba(0,0,0,0.10);
.temp_info{
font-size:18px;
font-weight:bold;
}
img{
max-width: 80px;
@media(max-width:480px){
max-width:120px;
}
width:100%;
margin: 10px auto;
}
.light-text{
color:${props => props.theme.textLightColor};
}
}
.swiper-wrapper{
margin:10px 0px;
}
`;
export default StyledToday;
```

styled week

import styled from 'styled-components';

```
const StyledWeek = styled.div`
color:${props => props.theme.textColor};
margin-top: 20px;
.box_info{
text-align: center;
display:flex;
justify-content:center;
background:${props => props.theme.bgSidebar};
color:${props => props.theme.textColor};
border-radius:20px;
padding: 15px;
width: 100%;
font-size:17px;
min-height:140px;
display: flex;
justify-content: center;
flex-directiaon: column;
-webkit-box-shadow: -5px 9px 10px -7px rgba(0,0,0,0.10);
-moz-box-shadow: -5px 9px 10px -7px rgba(0,0,0,0.10);
box-shadow: -5px 9px 10px -7px rgba(0,0,0,0.10);

.temp_info{
font-size:18px;
font-weight:bold;
}
img{
max-width: 80px;
@media(max-width:480px){
max-width:120px;
}
width:100%;
margin: 10px auto;
}
.light-text{
color:${props => props.theme.textLightColor};
}
}
.swiper-wrapper{
margin:10px 0px;
}
`;
export default StyledWeek;
```

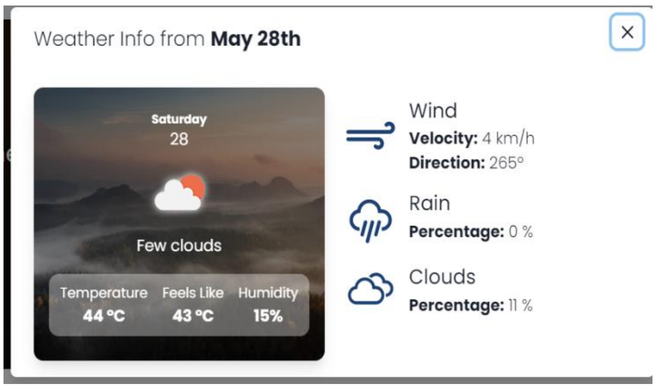
CHAPTER 6

TESTING

6.1 Test Case -1

Functional Test Cases

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



Weather Info from **May 28th**

Saturday 28

Few clouds

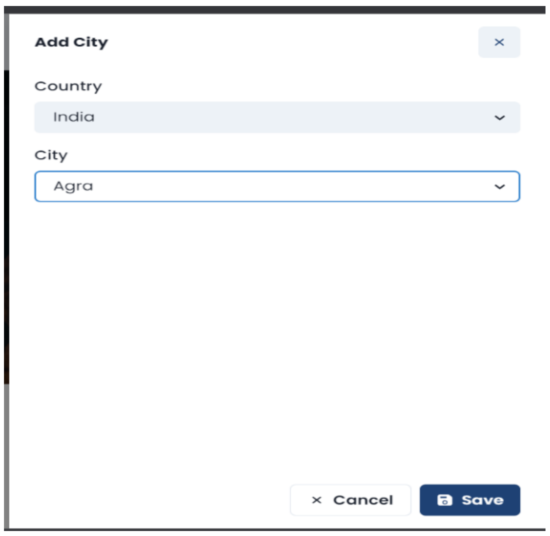
Temperature 44 °C, Feels Like 43 °C, Humidity 15%

Wind
Velocity: 4 km/h
Direction: 265°

Rain
Percentage: 0 %

Clouds
Percentage: 11 %

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

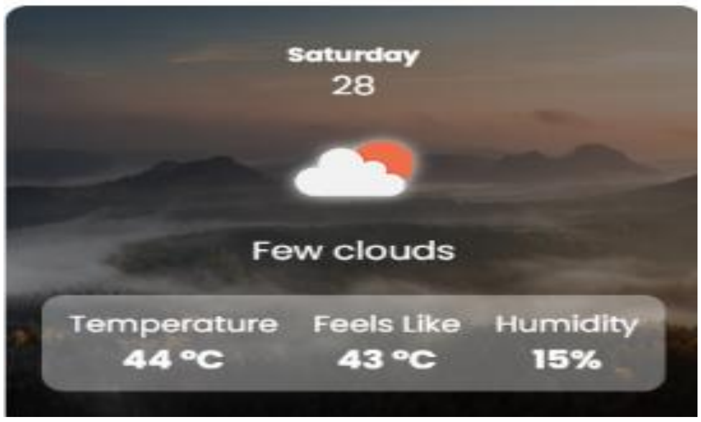


Add City

Country: India

City: Agra

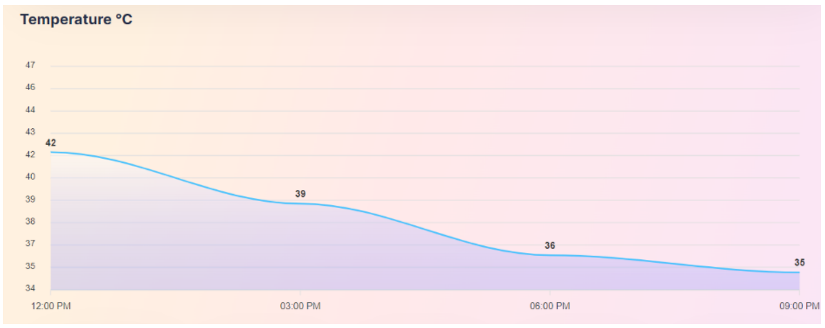
- ❖ Weather forecast should also display the minimum and maximum temperature of the day.



6.2 Test Case -2

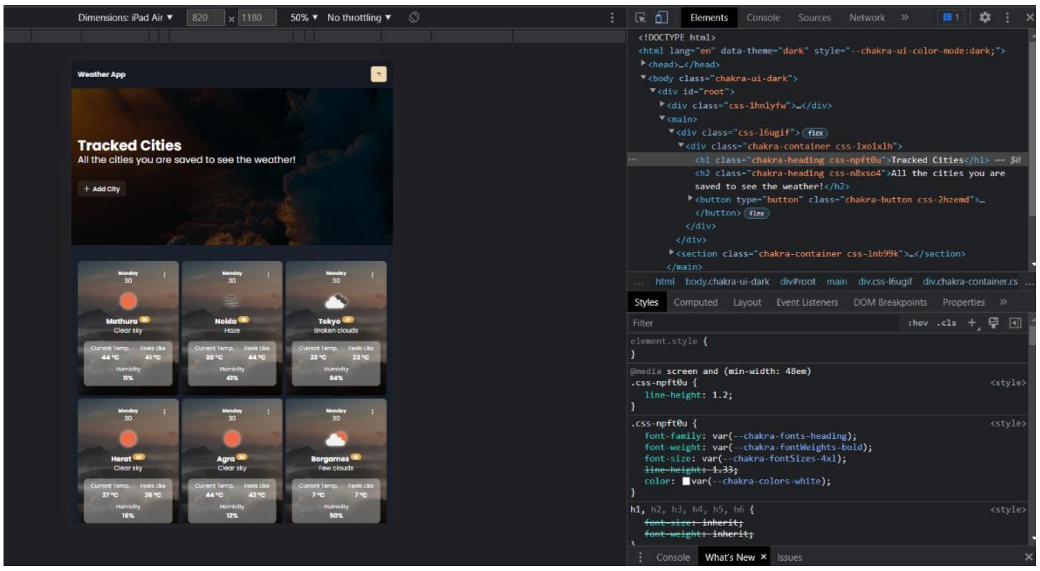
Non-Functional Test Cases

- ❖ Temperature graph is correct and smooth (Usability testing)



- ❖ Widget layout should automatically resize based on user selection / screen resolution (Portability testing).

For Tablets

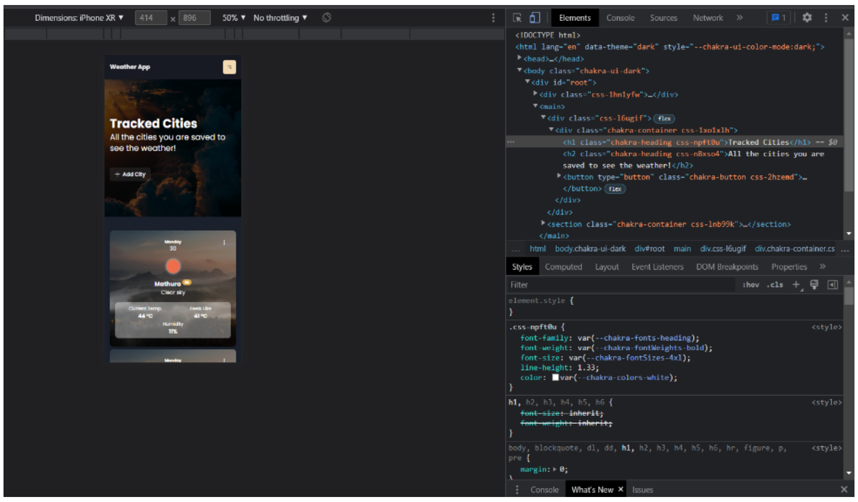


Dimensions: iPad Air ▾ 820 x 1180 50% ▾ No throttling ▾

```
<!DOCTYPE html>
<html lang="en" data-theme="dark" style="--chakra-ui-color-mode:dark;">
▶<head>…</head>
▼<body class="chakra-ui-dark">
  ▼<div id="root">
    ▶<div class="css-1hnlyfw">…</div>
    ▼<main>
      ▼<div class="css-l6ugif"> flex
        ▼<div class="chakra-container css-1xo1xlh">
            <h1 class="chakra-heading css-npft0u">Tracked Cities</h1> == $0
            <h2 class="chakra-heading css-n8xso4">All the cities you are
            saved to see the weather!</h2>
          ▶<button type="button" class="chakra-button css-2hzemd">…
            </button> flex
          </div>
        </div>
      ▶<section class="chakra-container css-1nb99k">…</section>
      </main>
```

```
element.style {
}
@media screen and (min-width: 48em)
.css-npft0u {                                                    <style>
   line-height: 1.2;
}
.css-npft0u {                                                    <style>
   font-family: var(--chakra-fonts-heading);
   font-weight: var(--chakra-fontWeights-bold);
   font-size: var(--chakra-fontSizes-4xl);
   ~~line-height: 1.33;~~
   color: var(--chakra-colors-white);
}
h1, h2, h3, h4, h5, h6 {                                         <style>
   ~~font-size: inherit;~~
   ~~font-weight: inherit;~~
```

For Mobiles



Dimensions: iPhone XR ▾ 414 x 896 50% ▾ No throttling ▾

```
<!DOCTYPE html>
<html lang="en" data-theme="dark" style="--chakra-ui-color-mode:dark;">
▶<head>…</head>
▼<body class="chakra-ui-dark">
  ▼<div id="root">
    ▶<div class="css-1hnlyfw">…</div>
    ▼<main>
      ▼<div class="css-l6ugif"> flex
        ▼<div class="chakra-container css-1xo1xlh">
            <h1 class="chakra-heading css-npft0u">Tracked Cities</h1> == $0
            <h2 class="chakra-heading css-n8xso4">All the cities you are
            saved to see the weather!</h2>
          ▶<button type="button" class="chakra-button css-2hzemd">…
            </button> flex
          </div>
        </div>
      ▶<section class="chakra-container css-1nb99k">…</section>
      </main>
```

```
element.style {
}
.css-npft0u {                                                    <style>
   font-family: var(--chakra-fonts-heading);
   font-weight: var(--chakra-fontWeights-bold);
   font-size: var(--chakra-fontSizes-4xl);
   line-height: 1.33;
   color: var(--chakra-colors-white);
}
h1, h2, h3, h4, h5, h6 {                                         <style>
   ~~font-size: inherit;~~
   ~~font-weight: inherit;~~
}
body, blockquote, dl, dd, h1, h2, h3, h4, h5, h6, hr, figure, p, <style>
pre {
   margin: ▶ 0;
```

CHAPTER 7

BIBLIOGRAPHY

Weather API:-

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

Countries API: - https://restcountries.com/v3.1/all

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