

# **WEATHER FORCAST APP**

## **A PROJECT REPORT**

### **Mini Project (01CE0609)**

*Submitted by*

**Kashyap Vekariya(92200103103)**

**Dhairya Aundhia(92200103268)**

**Yash Dhedhi (92200103292)**

## **BACHELOR OF TECHNOLOGY**

*in*

**Computer Engineering**



**Faculty of Engineering & Technology**

**Marwadi University, Rajkot**

**April, 2025**



**Marwadi**  
University  
Marwadi Chandarana Group



**Mini Project (01CE0609)**

Department of Computer Engineering

**Faculty of Engineering & Technology**

**Marwadi University**

**A.Y. 2025-26**

**CERTIFICATE**

This is to certify that the project report submitted along with the project entitled weather app has been carried out by Kashyap Vekariya(92200103103), Dhairya Aundhia(92200103268) , Yash Dhedhi (92200103292) under my guidance in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering, 6<sup>th</sup> Semester of Marwadi University, Rajkot during the academic year 2025-26.

Prof. Parth Shah

Internal Guide

Department of Computer Engineering

Dr. Krunal Vaghela

Professor & Head

Department of Computer Engineering




**Mini Project (01CE0609)**

Department of Computer Engineering  
**Faculty of Engineering & Technology**  
**Marwadi University**  
**A.Y. 2025-26**

**CERTIFICATE**

This is to certify that the project report submitted along with the project entitled **weather app** has been carried out by **Kashyap Vekariya(92200103103)** under my guidance in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering, 6<sup>th</sup> Semester of Marwadi University, Rajkot during the academic year 2025-26.

  
Prof. Parth Shah

Assistant Professor  
Department of Computer Engineering

Dr. Krunal Vaghela

Professor & Head  
Department of Computer Engineering



**Mini Project (01CE0609)**

Department of Computer Engineering  
**Faculty of Engineering & Technology**

**Marwadi University**

**A.Y. 2025-26**

**CERTIFICATE**

This is to certify that the project report submitted along with the project entitled **weather app** has been carried out by **Dhairya Aundhia(92200103268)** under my guidance in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering, 6<sup>th</sup> Semester of Marwadi University, Rajkot during the academic year 2025-26.

A handwritten signature in blue ink, appearing to be 'P. Shah', is written over the printed name of Prof. Parth Shah.

Prof. Parth Shah

Assistant Professor

Department of Computer Engineering

Dr. Krunal Vaghela

Professor & Head

Department of Computer Engineering



**Mini Project (01CE0609)**

Department of Computer Engineering  
**Faculty of Engineering & Technology**  
**Marwadi University**  
**A.Y. 2025-26**

**CERTIFICATE**

This is to certify that the project report submitted along with the project entitled **weather app** has been carried out by **Yash Dhedhi (92200103292)** under my guidance in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering, 6<sup>th</sup> Semester of Marwadi University, Rajkot during the academic year 2025-26.

  
Prof. Parth Shah

Assistant Professor

Department of Computer Engineering

Dr. Krunal Vaghela

Professor & Head

Department of Computer Engineering



**Marwadi**  
University  
Marwadi Chandarana Group



### **Mini Project (01CE0609)**

Department of Computer Engineering  
**Faculty of Engineering & Technology**  
**Marwadi University**  
**A.Y. 2025-26**

### **DECLARATION**

We hereby declare that the **Mini Project (01CE0609)** report submitted along with the Project entitled **weather app** submitted in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering to Marwadi University, Rajkot, is a bonafide record of original project work carried out by me / us at Marwadi University under the supervision of **Prof. Parth Shah** and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due reference.

	Name of the Student	Sign of Student
1	Kashyap Vekariya	
2	Dhairya Aundhia	
3	Yash Dhedhi	

## **Acknowledgement**

I sincerely thank Prof. Parth Shah for her guidance and support as my Internal Guide. Her knowledge and encouragement have been very important for the success of this project. I truly appreciate her mentorship and dedication, which have helped me learn and grow. She has always been supportive, approachable, and ready to share helpful advice. Her efforts have made a big positive impact on my project. I would also like to thank our Head of Department, Dr. Krunal Vaghela, for creating a supportive environment and encouraging innovation and learning. His guidance and motivation have inspired me to give my best throughout this project.

## **Abstract**

*A weather forecast app or website helps users check the weather based on their location and time. This project aims to create a user-friendly platform with various useful features. Key functions include real-time weather updates, wind speed, and temperature readings. Users can search for weather by precise local information. The app also provides forecasts for the next 3-4 days, helping users plan ahead. The design will be simple and attractive, using weather icons and charts to show information clearly. By using modern web technologies and APIs, this project will offer accurate and reliable weather details, keeping users informed and ready for any weather conditions. Also user can save locations to directly access the saved locations weather.*



## Table of Contents

Acknowledgement.....	i
Abstract .....	ii
Table of Contents.....	iii
List of Figures.....	vi
List of Tables.....	vii
<b>Chapter 1 Introduction to Project and Project Management.....</b>	<b>1</b>
1.1 Project Summary.....	1
1.2 Purpose.....	1
1.3 Objective.....	1
1.4 Scope (what it can do and can't do).....	1
1.5 Technology .....	2
1.6 Project Planning.....	2
1.6.1 Project Development Approach and Justification.....	2
1.6.2 Project Effort and Time, Cost Estimation.....	2
1.6.3 Roles and Responsibilities.....	3
1.7 Project Scheduling (Gantt Chart/PERT/Network Chart).....	3
<b>Chapter 2 System Analysis.....</b>	<b>4</b>
2.1 Study of Current System.....	4
2.2 Problem and Weaknesses of Current System.....	4
2.3 Requirements of New System.....	5
2.4 System Feasibility.....	5
2.4.1 Does the system contribute to the overall objectives of the organization?.....	5
2.4.2 Can the system be implemented using the current technology and within the given cost and schedule constraints.....	5
2.4.3 Can the system be integrated with other systems which are already in place?.....	5
2.5 Activity / Process in New System / Proposed System.....	6
2.6 Features of New System / Proposed System.....	6

2.7 List Main Modules / Components / Processes / Techniques of New System /	
Proposed System.....	7
2.8 Selection of Hardware / Software / Algorithms / Methodology / Techniques /	
Approaches and Justification .....	7
<b>Chapter 3 System Design.....</b>	<b>8</b>
3.1 System Design & Methodology.....	8
3.1.1 Flow Chart.....	8
3.1.2 Sequence Diagram.....	9
3.1.3 Class Diagram.....	10
3.1.4 Use Case Diagram .....	11
3.1.5 ER Diagram.....	12
3.2 Database Design / Data Structure Design / Circuit Design / Process Design /	
Structure Design.....	13
3.2.1 User Table.....	13
3.2.2 Login Table.....	13
<b>Chapter 4 Implementation.....</b>	<b>14</b>
4.1 Process / Program / Technology / Modules Specification(s).....	14
4.2 Implementation of Platform / Environment.....	15
4.3 Finding / Results / Outputs.....	20
<b>Chapter 5 Implementation.....</b>	<b>27</b>
5.1 Testing Plan / Strategy.....	27
5.2 Test Results and Analysis.....	28
5.2.1 Test Cases (test ID, test condition, expected output, actual output, remark).....	28
5.2.2 Result Analysis / Comparison.....	28
<b>Chapter 6 Implementation.....</b>	<b>29</b>
6.1 Overall Analysis of Project Viabilities.....	29
6.2 Problem Encountered and Possible Solutions.....	29
6.3 Summary of Project work.....	29
6.4 Limitations and Future Enhancement.....	30
6.5 Project Outcomes.....	30

References .....	31
------------------	----

### List of Figures

Fig 1.1 Gantt Chart.....	3
Fig 3.1 Flow Chart .....	8
Fig 3.2 Sequence Diagram.....	9
Fig 3.3 Class Diagram.....	10
Fig 3.4 Use Case Diagram.....	11
Fig 3.5 ER Diagram.....	12
Fig 4.1 Home Page Code.....	16
Fig 4.2 Login Page Code .....	17
Fig 4.3 Register Page Code.....	18
Fig 4.4 Home Page2 Code .....	19
Fig 4.5 Saved Location Page Code.....	20
Fig 4.6 Home Page.....	21
Fig 4.7 Login Page.....	23
Fig 4.8 Register Page.....	24
Fig 4.13 Home Page 2.....	25
Fig 4.14 Saved Page.....	27

**List of Tables**

Table 3.1 User Database Table .....13

Table 3.2 Save Database Table .....13

Table 5.1 Test Cases Table ..... 29

## Table of Contents

Acknowledgement.....	i
Abstract .....	ii
List of Figures .....	iii
List of Tables .....	iv
List of Abbreviations .....	v
Table of Contents .....	vi
<b>Chapter 1 Introduction to Project and Project Management.....</b>	<b>1</b>
1.1 Project Summary.....	2
1.2 Purpose.....	3
1.3 Objective.....	4
1.4 Scope.....	5
1.5 Technology.....	6
1.6 Project Planning.....	7
1.6.1 Project Development Approach and Justification.....	8
1.6.2 Project Effort and Time, Cost Estimation.....	
1.6.3 Roles and Responsibilities.....	
1.7 Project Scheduling (Gantt Chart/PERT/Network Chart).....	
<b>Chapter 2 System Analysis.....</b>	<b>4</b>
2.1 Study of Current System.....	4
2.2 Problem and Weaknesses of Current System.....	4
2.3 Requirements of New System.....	5
2.4 System Feasibility.....	5
2.4.1 Does the system contribute to the overall objectives of the organization?.....	5
2.4.2 Can the system be implemented using the current technology and within the given cost and schedule constraints.....	5
2.4.3 Can the system be integrated with other systems which are already in place?.....	5
2.5 Activity / Process in New System / Proposed System.....	6
2.6 Features of New System / Proposed System.....	

2.7 List Main Modules / Components / Processes / Techniques of New System / Proposed System.....	7
2.8 Selection of Hardware / Software / Algorithms / Methodology / Techniques / Approaches and Justification .....	7
<b>Chapter 3 System Design.....</b>	<b>8</b>
3.1 System Design & Methodology.....	8
3.1.1 Flow Chart.....	8
3.1.2 Sequence Diagram.....	9
3.1.3 Class Diagram.....	10
3.1.4 Use Case Diagram .....	11
3.1.5 ER Diagram.....	12
3.2 Database Design / Data Structure Design / Circuit Design / Process Design / Structure Design.....	13
3.2.1 User Table.....	13
3.2.2 Login Table.....	13
<b>Chapter 4 Implementation.....</b>	<b>14</b>
4.1 Process / Program / Technology / Modules Specification(s).....	14
4.2 Implementation of Platform / Environment.....	15
4.3 Finding / Results / Outputs.....	20
<b>Chapter 5 Implementation.....</b>	<b>27</b>
5.1 Testing Plan / Strategy.....	27
5.2 Test Results and Analysis.....	28
5.2.1 Test Cases (test ID, test condition, expected output, actual output, remark).....	28
5.2.2 Result Analysis / Comparison.....	28
<b>Chapter 6 Implementation.....</b>	<b>29</b>
6.1 Overall Analysis of Project Viabilities.....	29
6.2 Problem Encountered and Possible Solutions.....	29
6.3 Summary of Project work.....	29
6.4 Limitations and Future Enhancement.....	30
6.5 Project Outcomes.....	30

**References** .....



# CHAPTER 1

## INTRODUCTION

### 1.1 Project Summary

Weather Ease simplifies short-term weather planning by offering reliable and user-friendly 3-4 days forecasts. The app provides accurate weather details, including temperature, precipitation, wind speed, and humidity, with visually appealing graphics for better understanding.

### 1.2 Purpose

The purpose of the Weather Forecast App is to provide users with accurate and reliable weather information for the next 3-4 days, enabling informed decision-making and efficient planning. By delivering timely updates and actionable insights, the app helps users prepare for daily activities, travel, and outdoor events while ensuring safety and convenience.

### 1.3 Objective

Our goal is to develop a user-friendly web app, Weather Ease, for forecasting weather reliably. We aim to simplify

- Provide Accurate Weather Information.
- Enhance User Convenience.
- Promote Safety and Preparedness.

### 1.4 Scope

The scope of the Weather Forecast App encompasses providing accurate, short-term weather predictions for the next 3-4 days, catering to users across diverse geographical regions. The app is designed to deliver real-time, location-specific updates, ensuring users can effectively plan their daily activities, travel, and outdoor events. It targets a broad audience, including commuters, families, and outdoor enthusiasts, by offering features such as detailed forecasts, customizable alerts, and multi-location tracking.

With a user-friendly interface and visually rich graphics, the app aims to simplify weather monitoring. Additionally, it offers lifestyle integration by providing tailored suggestions

based on forecast data, such as appropriate clothing or travel plans. Built on robust technology and real-time data APIs, the app is scalable for future enhancements like extended forecasting, AI-driven recommendations, and community-based weather reporting. This makes it a comprehensive solution for weather-related planning and preparedness.

## 1.5 Technology

- **Visual Studio Code:** The primary development tool for creating the web application.
- **ReactJs Programming Language:** Utilized for frontend development to ensure code efficiency and reliability.
- **React:** Employed for designing user interfaces, providing flexibility and customization options.
- **MongoDB:** Integrated for user authentication, real-time data storage, and communication within the app, enhancing reliability and scalability.

## 1.6 Project Planning

### 1.6.1 Project Development Approach and Justification

We're using a step-by-step approach to develop Weather Ease. First, we'll plan everything out carefully to make sure we're on the right track. Then, we'll build the app bit by bit, adding features one by one. This way, we can test each part as we go to make sure everything works smoothly. We chose this approach because it helps us stay organized and fix any problems early on. Plus, it allows us to get feedback from users along the way, so we can make improvements based on what they need. By taking things step by step, we're making sure Weather Ease turns out to be the best it can be for everyone who uses it.

### 1.6.2 Project Effort and Time, Cost Estimation

We predict that creating Weather Ease will take about 5 months of work. This includes everything from planning to testing the app. We're aiming to spend around 40 hours a week on the project, totalling to approximately 800 hours altogether. As for costs, we've calculated that the project will require around 30,000 rupees. This covers expenses like software, tools, and other resources needed for development.

### 1.6.3 Roles and Responsibilities

As the solo member of the team working on the development of Weather Ease, all roles and responsibilities are assumed by me and my group members. This encompasses various tasks including project planning, app design, coding, testing, and refinement. Additionally, i will manage project timelines, budgets, and coordinate with any external resources or services required for the project.

## 1.7 Project Scheduling

### Gantt Chart

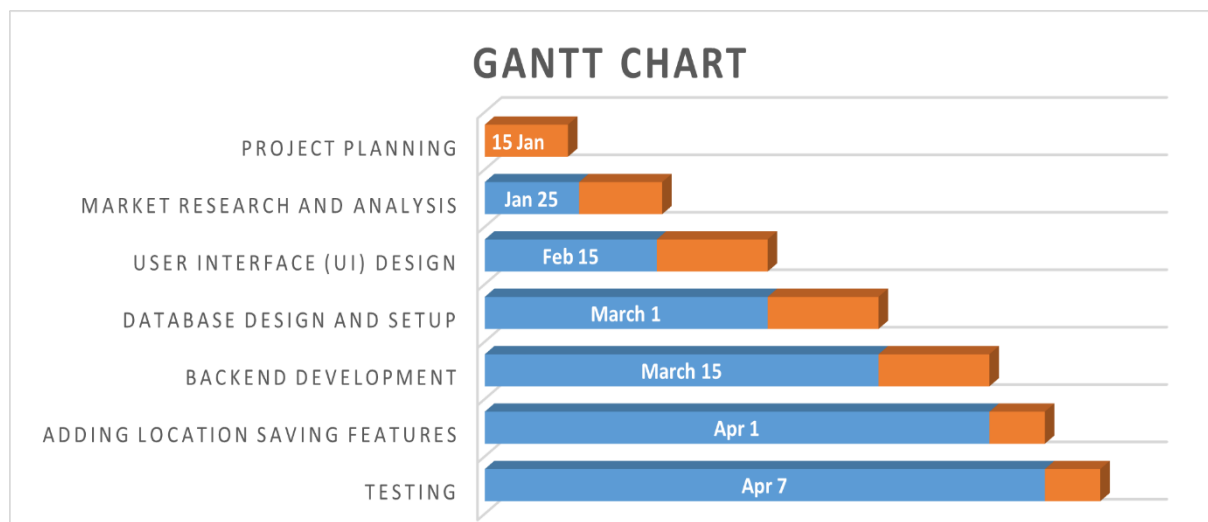


Fig 1.1 Gantt Chart

## CHAPTER 2

### SYSTEM ANALYSIS

#### 2.1 Study of Current System

This study involves analysing existing applications in the market to understand their features, functionalities, strengths, and weaknesses. Below are the findings from our examination of several existing pet adoption apps:

- **Self app developer – Weather Forecast**
  - Features: Offers a location-based search, customizable location, and a light/dark mode.
- **Save Locations**
  - Features: Enables to save multiple location for users to easy access. User can add new locations as well as delete saved locations. Can switch between locations easily.
- **Weather Forecast**
  - Features: Enables location-based weather forecast, displays weather forecast for next 3-4 days from today.

#### 2.2 Problem and Weaknesses of Current System

- **No Community Feel:** Most apps don't help users connect with each other. They're missing features that users are not able to save multiple locations.
- **Wrong Info:** Sometimes, the apps don't get location or weather details right. This can make users annoyed and unsure about trusting the app.
- **Too Many Ads:** Some apps show lots of ads, which can get in the way and make using the app annoying.
- **Hard to Use:** A few apps have tricky or badly designed menus, which can be frustrating for users trying to navigate the app.

## 2.3 Requirements of New System

- Make it easy for users to talk to each other, share tips, and support each other.
- Make sure all the information about weather forecast and locations is correct so users can trust the app.
- Don't show too many ads that annoy users. Keep the focus on helping user find correct and reliable output.
- Design the app so it's simple and easy to understand for everyone, with clear buttons and menus.

## 2.4 System Feasibility

### 2.4.1 Does the system contribute to the overall objectives of the organization?

No, the system does not directly contribute to the overall objectives of the organization. In the context of my major project, the focus lies primarily on developing technical skills and understanding software development concepts rather than directly aligning with broader organizational objectives

### 2.4.2 Can the system be implemented using the current technology and within the given cost and schedule constraints

Yes, if I have a proper team, the system can be implemented within the given schedule constraints. By effectively distributing tasks among team members, we can ensure timely completion of each phase of development. The estimated cost for the project is reasonable and can be managed efficiently. With careful planning and resource allocation, we can complete the project within the allocated budget.

### 2.4.3 Can the system be integrated with other systems which are already in place?

Yes, the system can be integrated with other systems that are already in place. By using standard protocols and APIs, we can ensure compatibility and seamless communication between our system and existing systems.

## 2.5 Activity / Process in New System

- **Requirement Gathering:** This initial phase involves gathering requirements from stakeholders, such as users, weather forecast applications, and administrators, to understand their needs and preferences for the app.
- **System Design:** In this phase, the system architecture, database schema, user interface design, and overall system flow are designed based on the gathered requirements.
- **Development:** In this phase, we write the code for the app, bringing the design to life and implementing features such as weather displaying, user profiles, and saving location functionality.
- **Testing:** Once development is complete, we thoroughly test the app to identify and fix any bugs or issues, ensuring it functions properly and meets user expectations.
- **Deployment:** After successful testing, the app is deployed to app stores, making it available for users to download and use on their devices.

## 2.6 Features of New System

- **Easy Navigation:** Simple and clear menus make it easy for users to find what they need without confusion.
- **Detailed Weather Forecast:** Each location should give precise forecast for the current day as well as for next 3-4 days, so users can plan accordingly their schedule.
- **Community Interaction:** A forum allows users to connect with each other, share advice, and ask questions about any suggestions for the application to be improved.
- **Personalized Favourites:** Users can save their favourite location for quick access and comparison, making the accessing process smoother.

## 2.7 List Main Modules / Components / Processes / Techniques of New System

- **User Management Module:** Responsible for user registration, login, profile management, and authentication processes.
- **Saved Location:** Allows users to save their favourite locations for quick access and comparison.
- **Backend Database:** Manages data storage and retrieval for user profiles, saved location and other app-related information.

## 2.8 Selection of Hardware / Software / Methodology

- **Hardware :** Users' devices should meet the following specifications: modern mobile devices with a minimum of 2GB RAM, quad-core processor (1.8GHz or higher), and at least 32GB of internal storage to handle the app's 30MB size and ensure smooth performance.
- **Software :** I opt for commonly used development tools like Visual Studio Code , providing robust environments for efficient app creation.
- **Methodology:** I use Agile development, which allows us to adapt to changes in the project as needed, ensuring our progress remains flexible and iterative.

## CHAPTER 3

### SYSTEM DESIGN

#### 3.1 System Design & Methodology

##### 3.1.1 Flow Chart

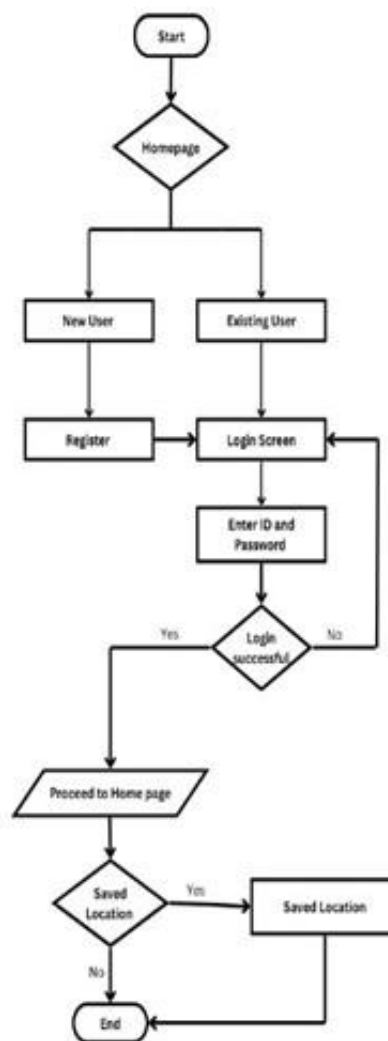


Fig 3.1 Flow Chart

This flowchart illustrates the user journey in a weather app. It covers user registration, login, and accessing the homepage to check for saved locations.



### 3.1.2 Sequence Diagram

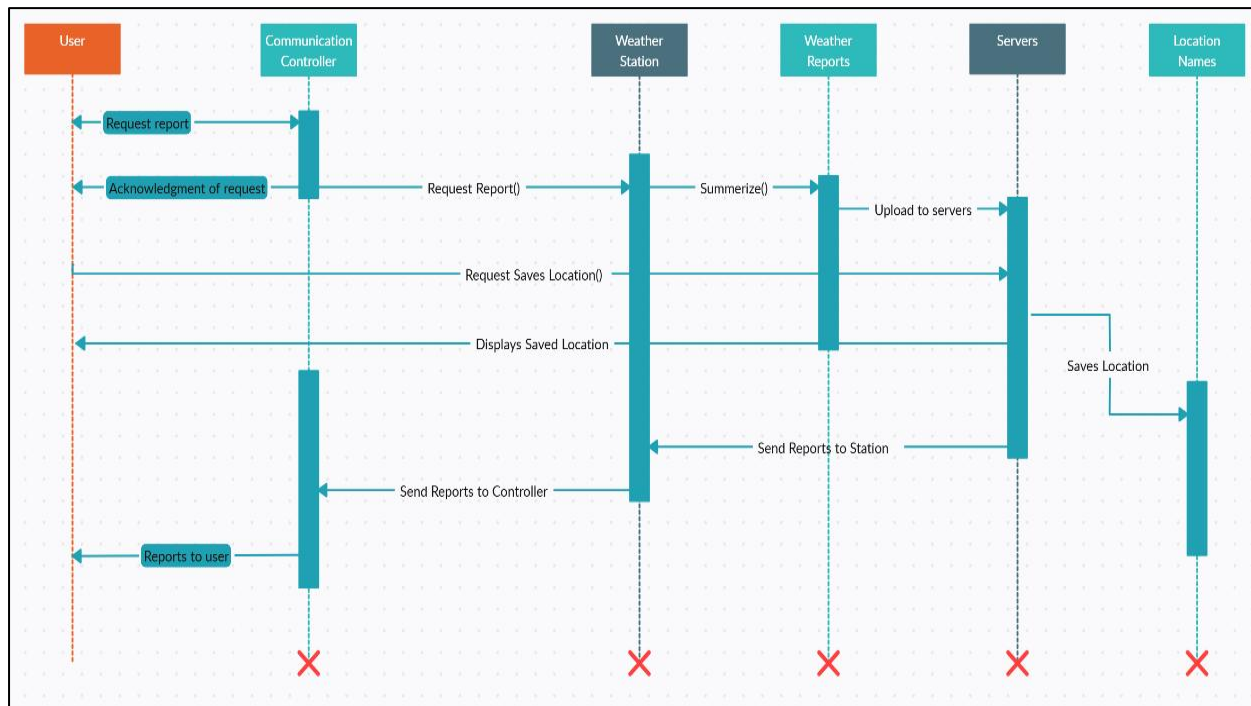


Fig 3.2 Sequence Diagram

The above diagram displays the sequence diagram of weather forecasting app which demonstrates how user interacts with server to use the functionality of weather forecast app. In which if user wants to fetch the weather than it will request to server with Request report() to the weather stations and further sends it to server to store data and sends back to the user. Also it performs task to save the location of user preferred to the server of easy accessibility for future use. And once it is saved user can direct fetch the weather of saved location immediately without working on the above process repeatedly.

### 3.1.3 Class Diagram

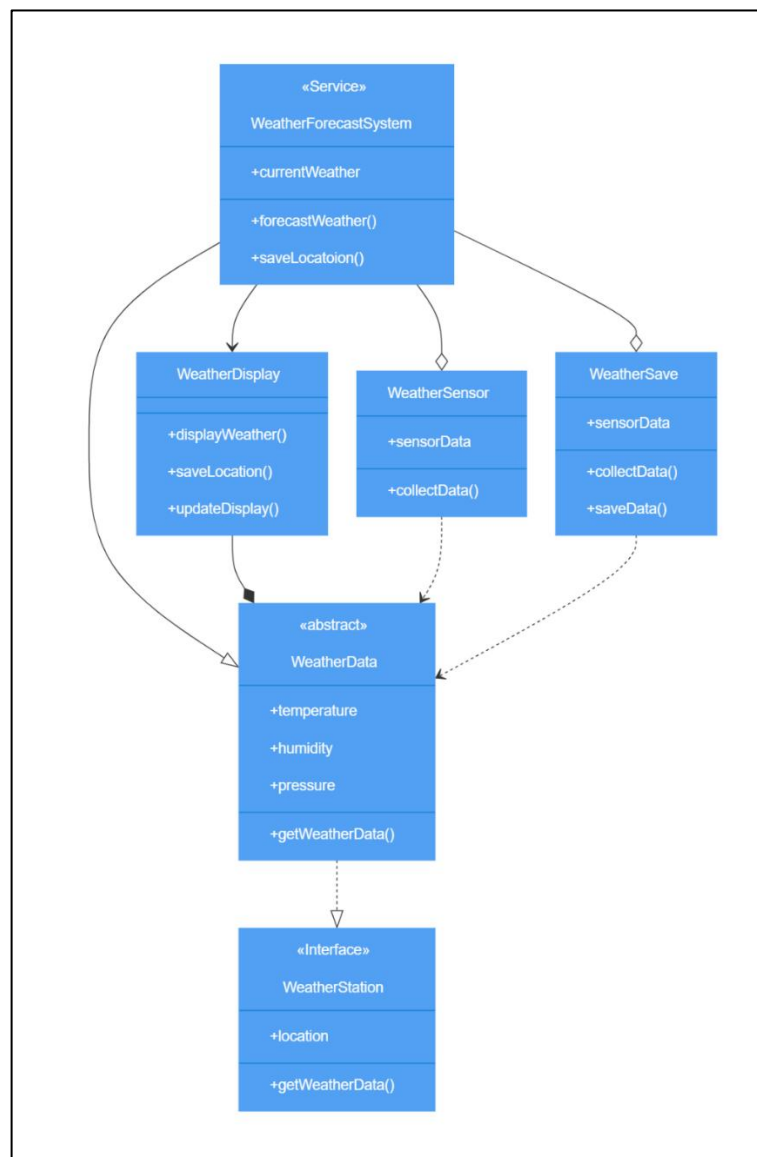


Fig 3.3 Class Diagram

This class diagram shows the architecture of a weather forecasting system. It includes classes like **WeatherDisplay**, **WeatherSensor**, and **WeatherSave**, all interacting with shared **WeatherData**. The system handles displaying, collecting, and saving weather data.

### 3.1.4 Use Case Diagram

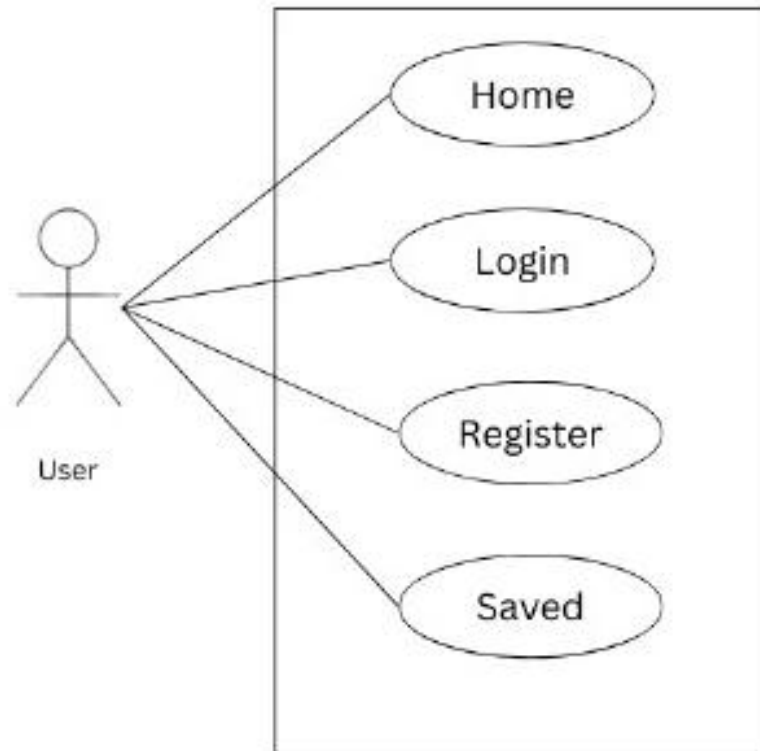
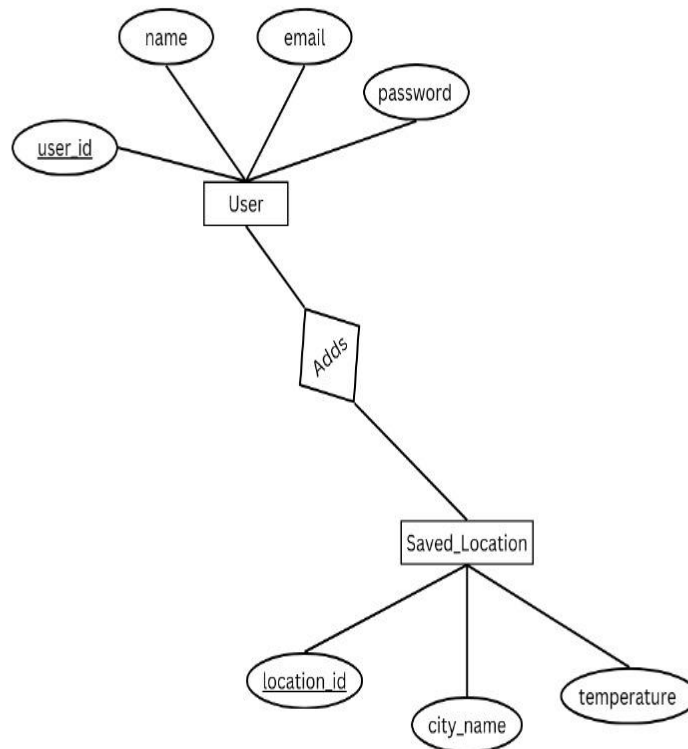


Fig 3.4 Use Case Diagram

- **Login:** Allows users to log in to the app.
- **Home:** Displays available weather forecast.
- **Saved:** Allows users to save their needed locations.
- **Register:** Allows users to register to the application if he/she is new to application.

### 3.1.5 ER Diagram

Fig 3.5 ER Diagram



- **Entities:**

- **User:** Represents app users. Attributes include password, Email, name, and User\_id (primary key).
- **Saved Location:** Stores user location. Attributes include location\_id (primary key), city\_name, temperature.

- **Relationships:**

- **Adds:** Connects User to Saved Location, indicating that users can add locations

## 3.2 Database Design / Data Structure Design / Circuit Design

### 3.2.1 User Table

Table 3.1 User Database Table

Field name	Data type	Constraint	Description
<b>user_id</b>	<b>int</b>	<b>primary key</b>	<b>Unique id of user</b>
user_name	varchar(50)	not null	Name of user
password	varchar(20)	not null	Password of user
email	varchar(50)	not null	Email of user

### 3.2.2 Saved Table

Table 3.2 Location Database Table

Field name	Data type	Constraint	Description
<b>location_id</b>	<b>int</b>	<b>primary key</b>	<b>Unique id of user</b>
city_name	varchar(50)	not null	Name of city

## CHAPTER 4

### IMPLEMENTATION

#### 4.1 Process / Program / Technology / Modules Specification(s)

- **Login Page:** This page serves as the entry point for users, requiring them to input their credentials to access the app's features securely.
- **Register Page:** Users can create a new account by providing necessary details like username, email, and password, facilitating the registration process.
- **Profile Page:** Here, users can view and manage their personal information, including current location, other details, and logout option.
- **Home Page:** The central hub of the app, displaying featured weather forecast, latest updates, and quick access to essential functionalities.
- **Saved Page:** Users can save their preferred locations on this page, facilitating easy access to their favorites for future reference.

## 4.2 Implementation of Platform / Environment

- Home page

```
export default function MainPanel() {
  const [active, setActive] = useState(0);
  const { address, setAddress } = useContext(WeatherContext); // Using WeatherContext to get address
  const [email, setEmail] = useState('');
  const [searchedLocation, setSearchLocation] = useState(null);

  // Function to handle saving location
  const handleSaveLocation = () => {
    if (email) {
      axios
        .post('http://localhost:3001/save-location', { email, location: address })
        .then((response) => {
          alert(response.data); // Display success or error message
        })
        .catch((err) => console.error('Error saving location:', err));
    } else {
      alert('Please log in to save locations.');
```

Fig 4.1 Home Page Code

- **Login Page**

```
const Login = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [errors, setErrors] = useState({});
  const [showSuccess, setShowSuccess] = useState(false);
  const [showWrongPassword, setShowWrongPassword] = useState(false);
  const [showNoRecords, setShowNoRecords] = useState(false);
  const navigate = useNavigate();

  const validate = () => {
    const errors = {};
    if (!email) {
      errors.email = "Email is required";
    } else if (!/^[a-zA-Z0-9+@-]+\.[a-zA-Z0-9+@-]+$/i.test(email)) {
      errors.email = "Email address is invalid";
    }
    if (!password) {
      errors.password = "Password is required";
    } else if (password.length < 6) {
      errors.password = "Password must be at least 6 characters long";
    }
    return errors;
  };

  const handleSubmit = (event) => {
    event.preventDefault();
    const validationErrors = validate();
    setErrors(validationErrors);

    if (Object.keys(validationErrors).length === 0) {
      axios.post('http://localhost:3001/login', { email, password })
        .then(result => {
          if (result.data === "Success") {
            localStorage.setItem('userEmail', email);
            setShowSuccess(true);
            setTimeout(() => {
              setShowSuccess(false);
              navigate('/home');
            }, 1000);
          } else if (result.data === "Wrong password") {
            setShowWrongPassword(true);
            setTimeout(() => setShowWrongPassword(false), 2000);
          } else {
            setShowNoRecords(true);
            setTimeout(() => setShowNoRecords(false), 2000);
          }
        });
    }
  };
};
```

Fig 4.2 Login Page Code



- **Register Page**

```
const Register = () => {
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [errors, setErrors] = useState({});
  const [showRegistered, setShowRegistered] = useState(false);
  const [showSuccess, setShowSuccess] = useState(false);
  const navigate = useNavigate();

  const validateField = (field, value) => {
    let error = "";
    if (field === "name") {
      if (!value.trim()) {
        error = "Name is required";
      } else if (value.length < 2) {
        error = "Name must be at least 2 characters";
      }
    } else if (field === "email") {
      if (!value.trim()) {
        error = "Email is required";
      } else if (!/\S+@\S+\.\S+/.test(value)) {
        error = "Invalid email format";
      }
    } else if (field === "password") {
      if (!value.trim()) {
        error = "Password is required";
      } else if (value.length < 6) {
        error = "Password must be at least 6 characters";
      } else if (!/(?=.*[A-Z])(?=.*\d)/.test(value)) {
        error = "Password must include an uppercase letter and a number";
      }
    } if (field === "confirmPassword") {
      if (!value.trim()) {
        error = "Confirm Password is required";
      } else if (value !== password) {
        error = "Passwords do not match";
      }
    }
    return error;
  };

  const handleFieldChange = (field, value) => {
    if (field === "name") setName(value);
    if (field === "email") setEmail(value);
    if (field === "password") setPassword(value);
  };
}
```

Fig 4.3 Register Page Code

- **Home Page (After Login)**

```

export default function MainPanel2() {
  const [active, setActive] = useState(0)
  const { time, date } = useDate()
  const { address, forecast } = useContext(WeatherContext)
  useEffect(() => {
    const transition = () => {
      setTimeout(function () {
        const replacers = document.querySelectorAll('[data-replace]')
        for (let i = 0; i < replacers.length; i++) {
          const replaceClasses = JSON.parse(
            replacers[i].dataset.replace.replace(/'/g, '')
          )
          Object.keys(replaceClasses).forEach(function (key) {
            replacers[i].classList.remove(key)
            replacers[i].classList.add(replaceClasses[key])
          })
        }
      }, 100)
    }
    transition()
  }, [])

  return (
    <div className="relative z-0 grid justify-center min-h-screen gap-3 p-8" style={{ backgroundColor: '#282c34' }}>
      {location ? (
        <div className='container max-w-5xl'>
          <div>
            <div>
              className='relative z-20 grid items-start w-full grid-cols-1 grid-rows-2 md:grid-cols-3 md:grid-rows-1 sm:item
              transition-all translate-y-12 ease-out'
              data-replace='{ "translate-y-12": "translate-y-0", "opacity-0": "opacity-100" }'
            </div>
            <div>
              <h2 className='text-xl font-bold dark:text-gray-100'>
                Local time:
              </h2>
              <h2 className='font-bold text-7xl dark:text-gray-100'>
                {time || ''}
              </h2>
              <h2 className='col-span-2 text-2xl dark:text-gray-100'>
                {date || ''}{''}
              </h2>
            </div>
          </div>
          <div className='w-full col-span-1 gap-3 text-left md:col-span-2'>
            <SearchBar />
            <div className='flex items-center justify-end w-full mv-2'>

```

Fig 4.4 Home Page2 Code

- **Saved Location Page**

```
const SavedPage = () => {
  const [savedLocations, setSavedLocations] = useState([]);
  const [email, setEmail] = useState('');
  const navigate = useNavigate(); // Initialize navigate

  // Function to delete a location
  const deleteLocation = (locationToDelete) => {
    axios.post('http://localhost:3001/delete-location', { email, location: locationToDelete })
      .then(response => {
        if (response.data === "Location deleted successfully") {
          alert("Location deleted!");
          setSavedLocations(savedLocations.filter(location => location !== locationToDelete));
        } else {
          alert("Error deleting location");
        }
      })
      .catch(err => console.error('Error deleting location:', err));
  };

  // Function to handle clicking a location to search it in the main panel
  const handleLocationClick = (location) => {
    localStorage.setItem('searchedLocation', location); // Save the location to localStorage
    navigate('/home'); // Redirect to the main panel
  };

  useEffect(() => {
    const userEmail = localStorage.getItem('userEmail');
    if (userEmail) {
      setEmail(userEmail);
      axios.post('http://localhost:3001/get-saved-locations', { email: userEmail })
        .then(response => {
          setSavedLocations(response.data);
        })
        .catch(err => console.error('Error fetching saved locations:', err));
    } else {
      alert("Please log in to view saved locations.");
    }
  }, [email]);

  return (
    <div className="w-full min-h-screen bg-gray-800 text-white flex items-center justify-center">
      <div className="container mx-auto px-4 py-8">
        <h2 className="text-3xl font-bold text-center mb-6">Saved Locations</h2>
        {savedLocations.length ? (
          <ul className="list-none w-full max-w-2xl mx-auto">
            {savedLocations.map((location, index) => (
              <li>
```

Fig 4.5 Save Location Page Code

4.3 Finding / Results / Outputs

- Home Page

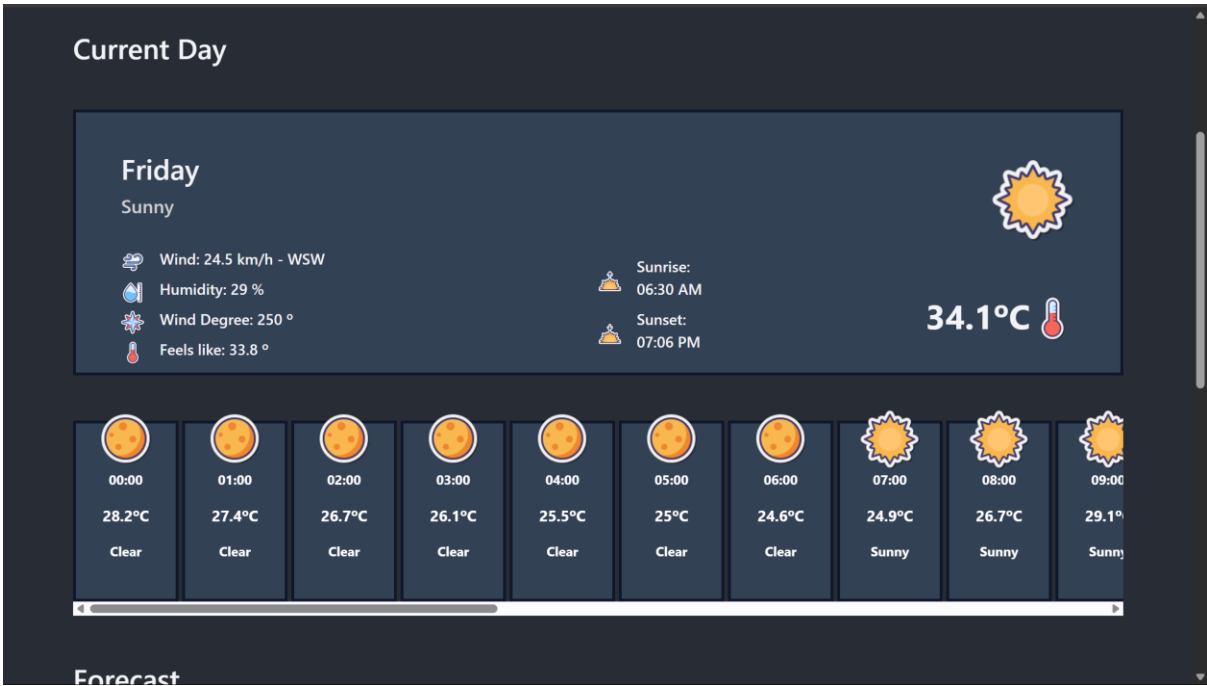
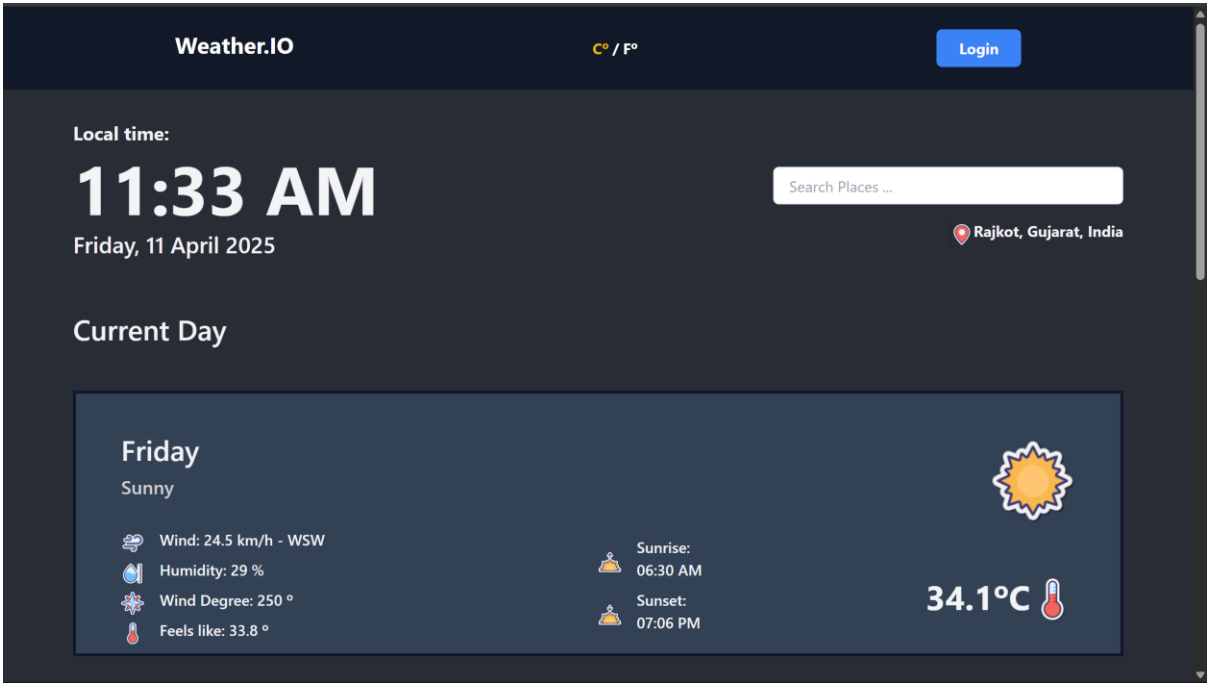
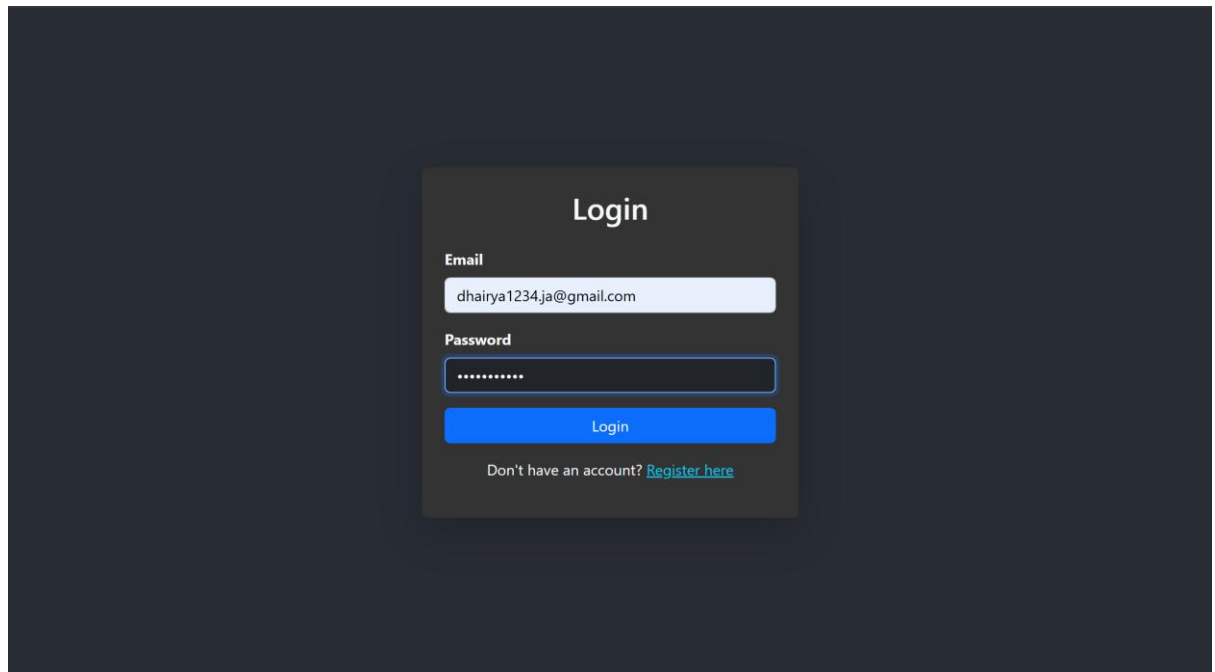




Fig 4.6 Home Page

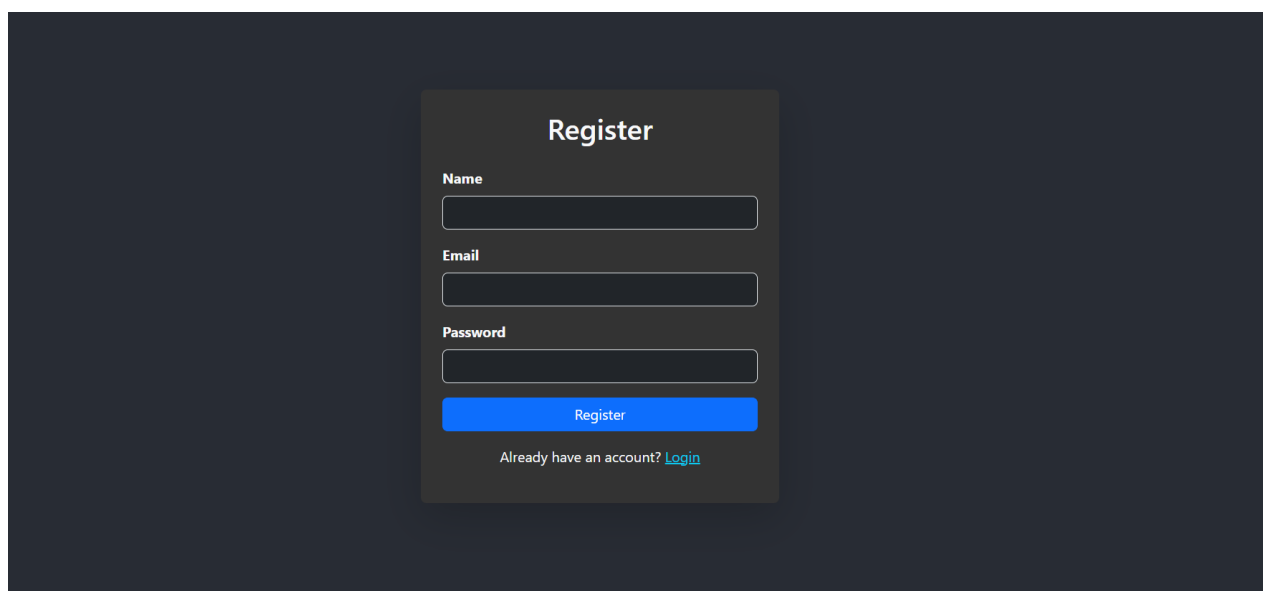
- **Login Page**



A screenshot of a login page with a dark blue background. In the center is a dark gray rectangular box containing the login form. The box is titled "Login" in white text. Below the title, there are two input fields: "Email" with the text "dhairya1234.ja@gmail.com" and "Password" with masked characters ".....". Below these fields is a blue button labeled "Login". At the bottom of the box, there is a link that says "Don't have an account? [Register here](#)".

Fig 4.7 Login Page

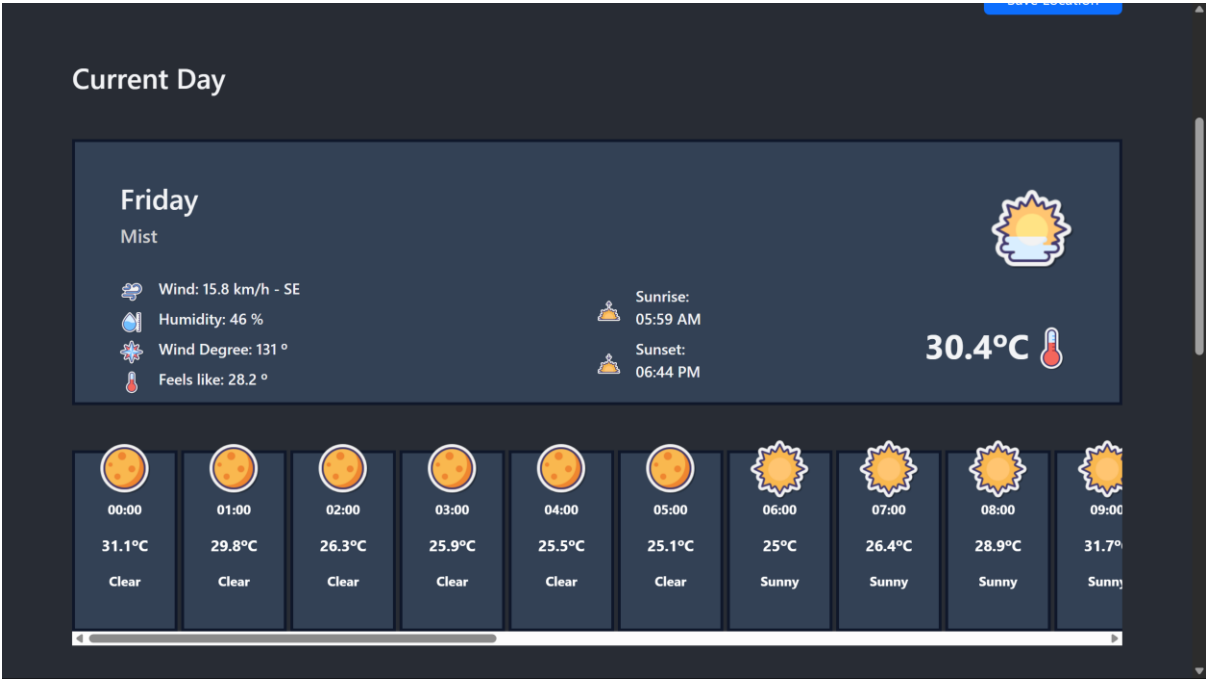
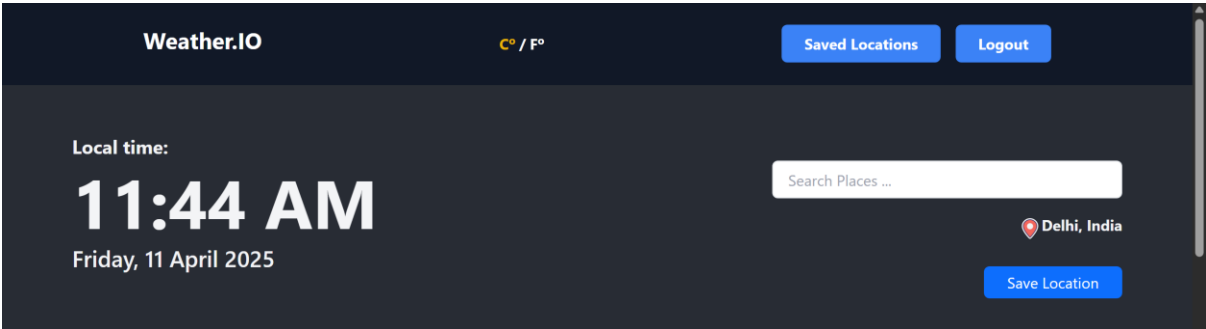
- **Register Page**



A screenshot of a register page with a dark blue background. In the center is a dark gray rectangular box containing the register form. The box is titled "Register" in white text. Below the title, there are three input fields: "Name", "Email", and "Password". Below these fields is a blue button labeled "Register". At the bottom of the box, there is a link that says "Already have an account? [Login](#)".

Fig 4.8 Register Page

- Home Page (After Login)



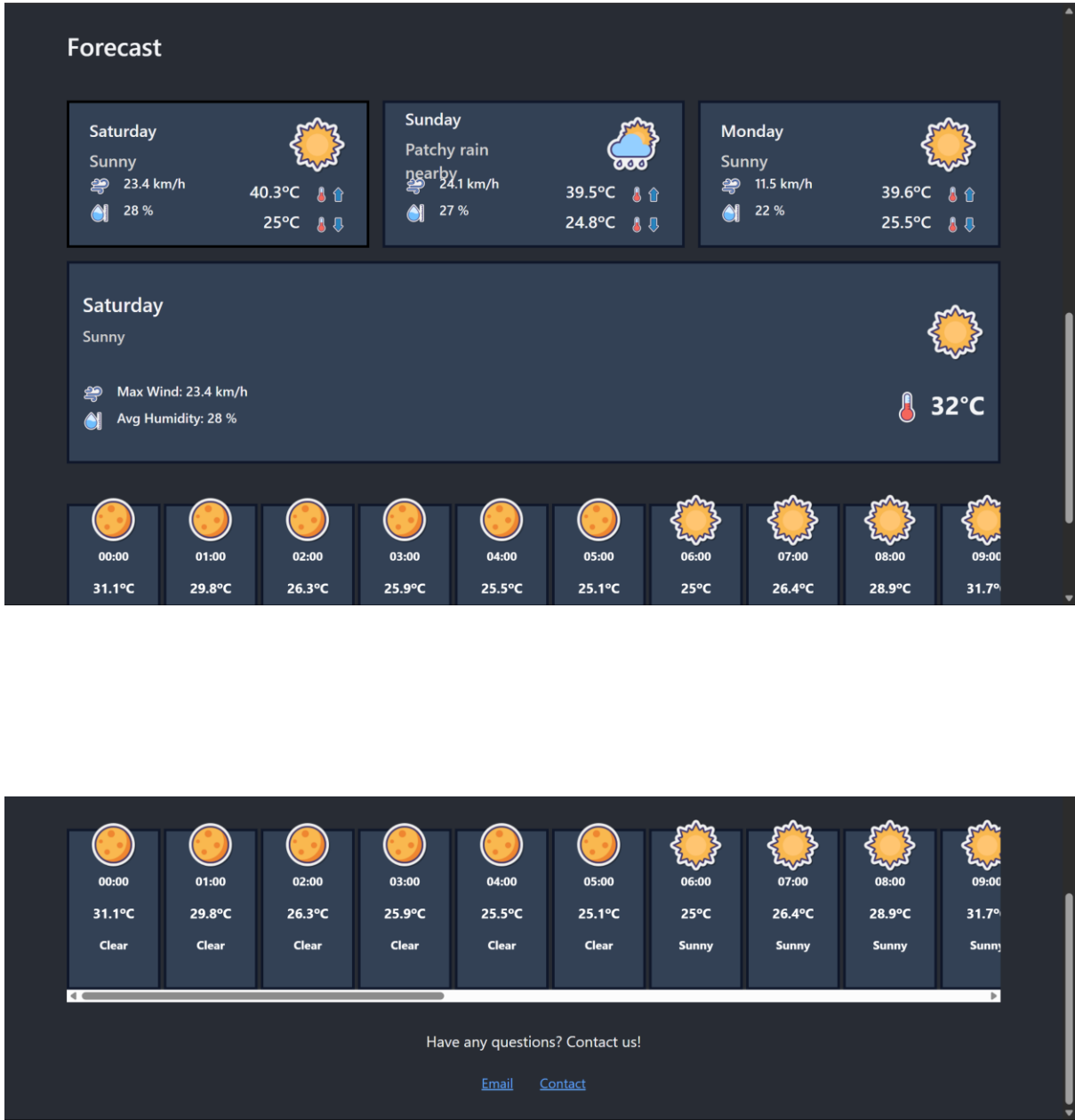


Fig 4.9 Home Page 2



- **Saved Page**

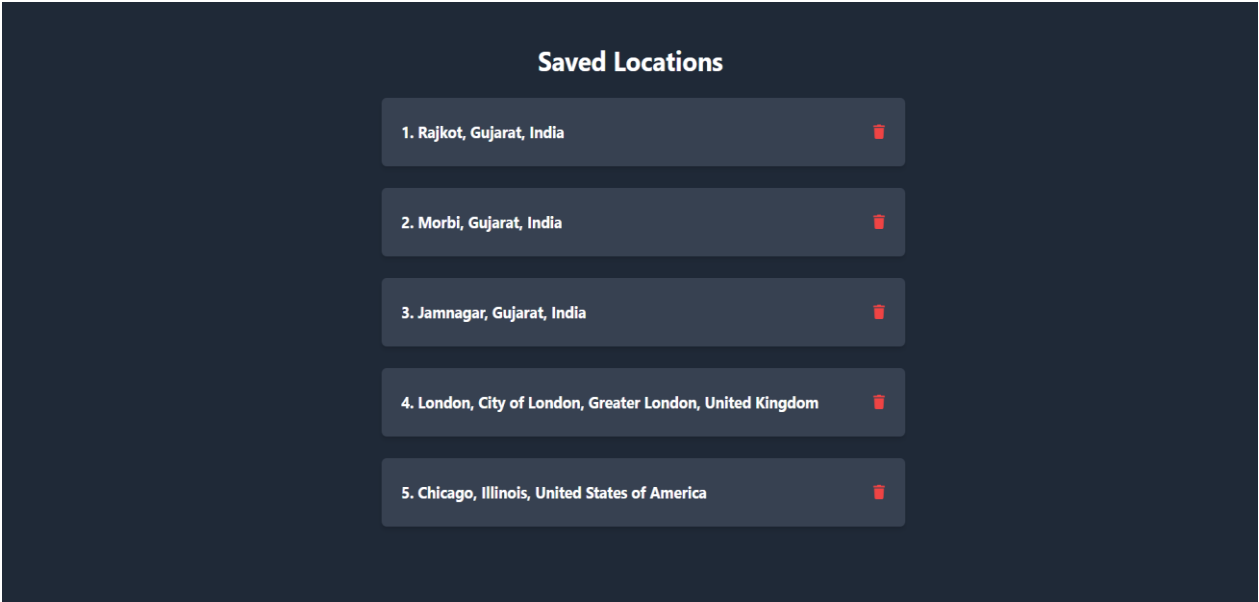
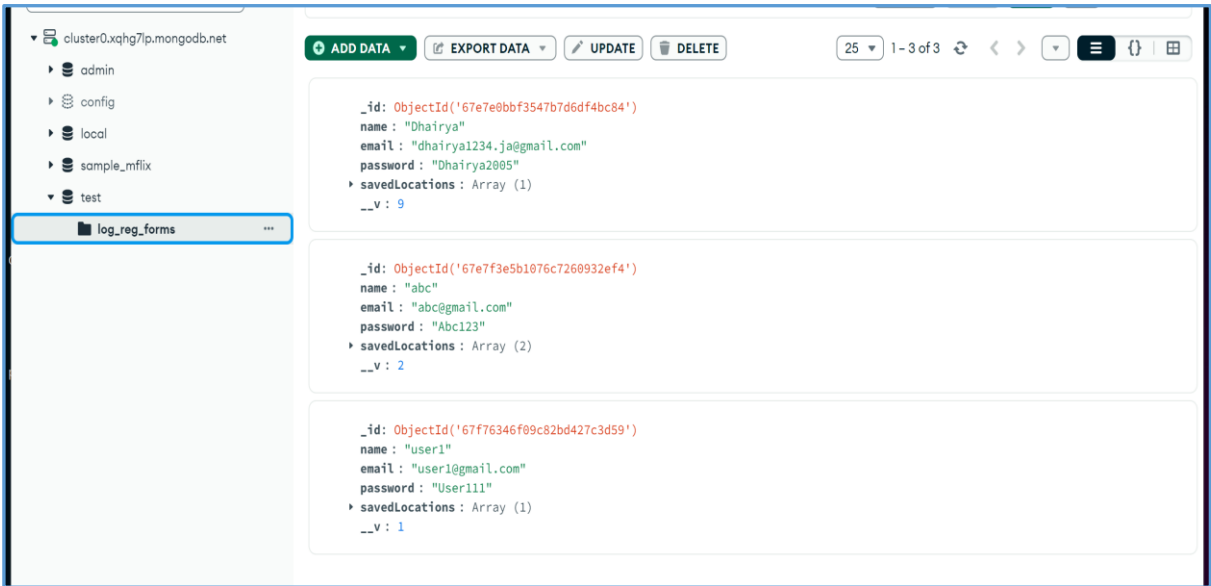


Fig 4.10 Saved Location Page



## CHAPTER 5

### TESTING

#### 5.1 Testing Plan / Strategy

- **Unit Testing:** We'll check each feature of the app individually. For example, we'll test if the login button works correctly by clicking on it and making sure it takes users to the login screen.
- **Integration Testing:** We'll test how different parts of the app work together. For instance, we'll check if users can add a pet to their favorites and then view it on their profile page.
- **User Testing:** Real users will try out the app. They'll be asked to perform tasks like finding a pet to adopt or posting on the community forum. Their feedback will help us improve the app.
- **Compatibility Testing:** We'll test the app on various devices, such as different models of smartphones and tablets, to ensure it works well on all of them.
- **Performance Testing:** We'll test how fast the app loads and how smoothly it runs, especially when there are many users using it at the same time.

## 5.2 Test Results and Analysis

### 5.2.1 Test Cases

Table 5.1 Test Cases Table

Test ID	Test Condition	Expected Output	Actual Output	Remark
TC-1	User enters valid username and password	User successfully logs into the app	User successfully login	Working
TC-2	User enters invalid username and password	App displays an error message	App display error	Working
TC-3	User selects a location from the list	Weather details are displayed	Weather details are displayed	Working
TC-4	User adds a location to saved	Location is added to the saved list	Location is added to saved list	Working
TC-5	User deletes location from saved location.	Location is successfully deleted.	Location is successfully deleted.	Working

### 5.2.2 Result Analysis / Comparison

The result analysis and comparison of the conducted test cases reveal several key findings. In cases where users entered valid and invalid login credentials (TC-1 and TC-2), the application performed as expected, successfully allowing access for valid credentials and displaying appropriate error messages for invalid ones.

Also works completely when users attempted to select a location from the list (TC-3), as the expected pet details were displayed, indicating successful functionalities. Similarly, adding a location to saved list (TC-4) did function as intended, with the location being added to the saved list.

On the other hand, actions such as posting on the deleting location (TC-5) yielded the expected outcomes, indicating successful functionalities. These results highlight areas of success and potential areas for improvement in the application's functionality and user experience.

On the other hand, actions such as posting on the deleting location (TC-5) yielded the expected outcomes, indicating successful functionalities. These results highlight areas of success and potential areas for improvement in the application's functionality and user experience.

## CHAPTER 6

### CONCLUSION & OUTCOMES

#### 6.1 Overall Analysis of Project Viabilities

The overall analysis of the project's viability indicates its potential for success. By considering factors like feasibility, resources, and market demand, we conclude that the project has a good chance of achieving its objectives.

The app's concept aligns well with current trends in pet adoption and community engagement, suggesting a favorable reception from users. Additionally, the availability of necessary resources and technologies for development further supports its viability. However, challenges such as competition and user adoption may pose obstacles that require careful consideration and strategic planning.

#### 6.2 Problem Encountered and Possible Solutions

**Problem Encountered:** One of the significant challenges encountered during the project was related to establishing a stable connection with the Firebase database and efficiently fetching data from it.

**Solutions:** To address the problem of MongoDB database connection and data fetching, several solutions could be considered. Firstly, ensuring that the app's MongoDB configuration is correctly set up and that the necessary functions are included in the project which helps stabilizing the connection.

#### 6.3 Summary of Project work

In our project, we created Weather Ease, a simple mobile app to make weather forecasting easier. We started by looking at other weather forecasting apps and then planned how GeoWeather would work. We made sure it was easy to use and looked good on different devices. We faced some problems, like getting data from the database slowly, but we fixed them by using better methods. Finally, we made GeoWeather with features like finding location easily, managing saved location, and getting feedback from user. Overall, our goal was to make weather forecast app a happy experience for everyone.

### 6.4 Limitations and Future Enhancement

- **Limitations:** One limitation of GeoWeather is the absence of location filters, like for international and national making it harder for users to find specific local locations. Additionally, the app offers only a limited functionalities of app to those users who has not logged in, which may not cover all the options users are looking for including saving location for more cities.
- **Future Enhancements:** To address these limitations, future updates could focus on adding more location filters, allowing users to easily search for specific local locations. Additionally, expanding the range of saving location pinning priority location catering to a wider audience of weather forecast users.

### 6.5 Project Outcomes

The GeoWeather project resulted in creating a simple-to-use web app for easier weather forecasting. GeoWeather allows users to find locations and their weather easily, save multiple location to the profile, and overcome common issues in existing weather forecasting apps. By continually improving, GeoWeather aims to make weather forecasting more enjoyable and responsible while building a supportive community for users.

## References

- [1] WeatherAPI. (n.d.). Official WeatherAPI Documentation. Retrieved from <https://www.weatherapi.com/docs/> - Used to understand weather API endpoints, request parameters, and response formats.
- [2] Stack Overflow. (n.d.). Programming Q&A Community. Retrieved from <https://stackoverflow.com/> - Referenced for resolving errors and troubleshooting issues during development.
- [3] RapidAPI. (n.d.). API Hub for Testing and Integration. Retrieved from <https://rapidapi.com/> - Utilized for API testing and managing weather data integrations.
- [4] <https://nominatim.org/release-docs/latest/api/Search/>  
Used for search bar