

# **AERO FORECAST**

*A project report submitted in partial fulfilment of the requirements for  
the award of the degree of*

## **Bachelor of Technology**

in

**Department of CSE-Artificial Intelligence and Machine Learning**

*By*

<b>D. Vishnu Vardhan</b>	<b>-</b>	<b>2411CS020138</b>
<b>D. Satish Kumar</b>	<b>-</b>	<b>2411CS020139</b>
<b>D. Uday</b>	<b>-</b>	<b>2411CS020141</b>
<b>D. Trilok krishna</b>	<b>-</b>	<b>2411CS020150</b>
<b>G. Harsha Vardhan</b>	<b>-</b>	<b>2411CS020173</b>

Under the esteemed guidance of

**Mr. S. Ashok**

Assistant Professor



**Department of Artificial Intelligence and Machine Learning School  
Of Engineering**

**MALLAREDDYUNIVERSITY**

Masiammaguda, Dulapally, Hyderabad, Telangana-500100

**2025**



# **MALLA REDDY UNIVERSITY**

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

**Department of Computer Science and Engineering**  
**(Artificial Intelligence and Machine Learning)**

**CERTIFICATE**

This is to certify that this is the project report of the Application Development entitled **AERO FORECAST** submitted by **Vishnu Vardhan (2411CS020138), Satish Kumar (2411CS020139), Uday (2411CS020141), Trilok krishna (2411CS020150), Harsha Vardhan (2411CS020173)** towards the partial fulfillment of the award of Bachelor's Degree in Project Development from the Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning), Malla Reddy University, Hyderabad. This is a record of project work done by us. The results embodied in the work have not been submitted to any other university or institute for the award of any degree or diploma.

**INTERNAL GUIDE**

Mr. S. Ashok

**HEAD OF THE  
DEPARTMENT**

Dr . R Nagaraju  
CSE(AI&ML)

**DEAN**

Dr . G Gifta Jerith  
CSE(AI&ML)

**EXTERNAL EXAMINER**

## DECLARATION

We hereby declare that the project report entitled “Aero Forecast” Using Machine Learning” has been carried out by us and this work has been submitted to the Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning), Malla Reddy University, Hyderabad in partial fulfillment of the requirements for the award of degree of Bachelor of Technology. I further declare that this project has not been submitted in full or part for the award of any other degree in any other educational institutions.

Place:Hyderabad

Date:    /    / 2025

Name	RollNumber	Signature
D. Vishnu Vardhan	2411CS020138	
D. Satish kumar	2411CS020139	
D. Uday	2411CS020141	
D. Trilok krishna	2411CS020150	
G. Harsha Vardhan	2411CS020173	

## ACKNOWLEDGEMENT

We extend my sincere gratitude to all those who have contributed to the completion of this project report. Firstly, I would like to extend my gratitude to Dr. V. S. K Reddy, Vice Chancellor, for his visionary leadership and unwavering commitment to academic excellence.

We would like to thank Dr. G Gifta Jerith ,Dean, School of Engineering - Artificial Intelligence and Machine Learning, for her encouragement and support throughout our academic pursuit.

We am also grateful to Dr. R Nagaraju, Head of the Department of Computer Science and Engineering–Artificial Intelligence and Machine Learning, for providing us with necessary resources and facilities to carry out this project.

We would also like to express my deepest appreciation to our project guide, Mr. S. Ashok, Assistant Professor, whose invaluable guidance, insightful feedback, and unwavering support have been instrumental throughout the course of this project for successful outcomes.

We are deeply indebted to all of them for their support, encouragement, and guidance, without which this project would've been impossible.

D. Vishnu Vardhan	-	2411CS020138
D. Satish kumar	-	2411CS020139
D. Uday	-	2411CS020141
D. Trilok krishna	-	2411CS020150
G. Harsha Vardhan	-	2411CS020173

## **ABSTRACT**

The Aero Forecast application is an innovative weather tool designed to deliver real-time weather updates while focusing on public safety, agricultural planning, and healthcare awareness. Unlike traditional weather apps, Aero Forecast goes beyond basic meteorological data by offering a range of features that cater to diverse user needs, particularly those of rural and agricultural communities. It not only provides current weather information but also detailed seasonal forecasts that guide farmers in planning crop cycles, helping them select optimal crops based on upcoming weather patterns.

By integrating seasonal and climate data with crop recommendations, Aero Forecast aids in maximizing yields and minimizing potential crop loss from sudden weather changes. In addition to agricultural guidance, Aero Forecast includes a unique health feature that alerts users to potential seasonal disease risks tied to specific weather conditions, such as changes in humidity, temperature, or rainfall. This function serves both urban and rural populations, helping raise awareness and encouraging preventive health measures against seasonal illnesses. By providing timely information on possible health risks, Aero Forecast strengthens public health efforts, especially in areas with limited medical access, enabling communities to take proactive precautions.

The app also incorporates a vital emergency feature for users who may find themselves in immediate danger during natural disasters like floods or hurricanes. With a one-touch SOS button, users can send alerts to rescue teams, sharing their location and request for assistance. This feature is designed to work offline by leveraging SMS and GPS technology, ensuring emergency communication even in disaster zones with no internet connectivity.

In summary, Aero Forecast combines real-time weather insights with critical tools for agriculture, health, and emergency response. Its user-friendly design and multifunctional approach make it an essential tool for communities seeking dependable weather information and immediate support in emergencies, addressing crucial environmental, health, and safety needs across diverse user groups.

## **CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
<b>1</b>	<b>INTRODUCTION</b>	
	1.1 Project Identification	1
	1.2 Objective of project	1
	1.3 Scope of the project	2
<b>2</b>	<b>ANALYSIS</b>	
	2.1 Project Planning and Research	3
	2.2 Software requirement specification	4
	2.2.1 Software requirement	4
	2.2.2 Hardware requirement	5
	2.3 Model Selection and Architecture	6
<b>3</b>	<b>DESIGN</b>	
	3.1 Introduction	7
	3.2 DFD Diagram	7
	3.3 ER Diagram	7
	3.4 UML Diagram	7
	3.5 Data Set Descriptions	8
	3.6 Data Preprocessing Technique	8
	3.7 Methods & Algorithms	10
<b>4</b>	<b>DEPLOYMENT AND RESULTS</b>	
	4.2 Source Code	11
	4.3 Model Implementation	11

and Training	
4.4 Model Evaluation Metric	12
4.5 Model Deployment: Testing and Validation	12
4.6 Web GUI's Development	13
4.7 Results	40
<b>CONCLUSION</b>	
Project conclusion	43
Future Scope	44
References	45

# **CHAPTER-1**

## **INTRODUCTION**

### **1.1 PROJECT IDENTIFICATION / PROBLEM DEFINITION**

#### **1.1.1 Background**

With climate patterns becoming increasingly unpredictable, real-time access to accurate weather information is essential. While there are numerous weather applications available, they often lack tailored features that address the specific needs of various user groups, such as farmers, healthcare providers, and individuals in high-risk disaster areas. The AREO FORECAST application is designed to fill this gap, offering a comprehensive weather solution that goes beyond simple forecasts.

#### **1.1.2 Problem Statement**

Rural communities, particularly those in agriculture-dependent regions, face unique challenges when weather patterns shift unexpectedly. Farmers require not only timely weather updates but also guidance on seasonal trends to make informed planting and harvesting decisions. Additionally, seasonal weather patterns often correlate with specific health risks, but few applications alert users to potential disease outbreaks based on environmental conditions. Lastly, natural disasters pose a serious threat, especially in regions with unreliable internet connectivity, where access to emergency assistance can be critical. There is a need for a robust, multi-functional application that provides real-time weather data, agricultural insights, health alerts, and emergency support, even offline.

### **1.2 OBJECTIVE OF THE PROJECT**

#### **1.2.1 Primary Objective**

The main goal of AREO FORECAST is to develop a multi-functional weather application that serves as a reliable resource for public safety, agricultural planning, health awareness, and emergency response.

### 1.2.2 Specific Objectives

- To provide accurate, real-time weather reports and seasonal forecasts.
- To guide farmers on optimal crop selection based on upcoming weather patterns.
- To alert users to potential seasonal diseases influenced by specific weather conditions.
- To ensure emergency communication in disaster situations through an offline-capable SOS feature that notifies rescue teams.

## 1.3 SCOPE OF THE PROJECT

### 1.3.1 Functional Scope

AREO FORECAST is designed to be a holistic weather application with multiple features catering to diverse user needs:

- **Real-Time Weather Updates:** Up-to-the-minute weather data, including temperature, humidity, precipitation, and other key metrics.
- **Seasonal Forecasts for Agriculture:** Insightful data and suggestions to aid farmers in selecting crops based on upcoming seasonal weather trends.
- **Health Risk Alerts:** Notifications related to potential seasonal diseases based on environmental conditions, helping users take preventive measures.
- **Emergency Assistance Feature:** An offline SOS button that alerts rescue teams to the user's location during natural disasters, regardless of internet connectivity.

### 1.3.2 Geographic and User Scope

AREO FORECAST is intended for use by individuals across urban and rural areas. It is particularly valuable for:

- **Farmers and Agricultural Workers:** who rely on weather forecasts and agricultural guidance to optimize crop yields.
- **Public Health Officials and Individuals:** interested in staying informed about seasonal health risks.

- **Residents in Disaster-Prone Regions:** needing access to emergency support during natural disasters.

### 1.3.3 Technological Scope

AREO FORECAST will integrate advanced technologies to ensure reliability and accessibility:

- **GPS and Offline SMS Technology:** to enable the emergency SOS feature even in areas with no internet connectivity.
- **User-Friendly Interface:** to ensure ease of use for a broad range of users, regardless of technological familiarity.

## CHAPTER-2

### ANALYSIS

#### 2.1. PROJECT PLANNING AND RESEARCH

##### 2.1.1 Project Scope and Objectives

The AREO FORECAST application is designed to address the needs of diverse user groups by providing real-time weather information, seasonal forecasts for agricultural guidance, health risk alerts, and an emergency assistance feature. The application's primary goals are to enhance public safety, improve agricultural productivity, and raise healthcare awareness. Research in climate data analysis, emergency communication protocols, and public health patterns forms the basis of the application's development.

##### 2.1.2 Feasibility Study

A feasibility study was conducted to assess the practicality of AREO FORECAST, focusing on technical, operational, and economic factors:

- **Technical Feasibility:** Given the availability of accurate weather APIs, offline SMS capabilities, and mobile GPS technologies, developing an application with real-time data, crop recommendations, and health alerts is technically feasible.
- **Operational Feasibility:** The project will benefit rural and urban populations, particularly farmers, healthcare providers, and individuals in disaster-prone regions, making the app highly practical and beneficial.
- **Economic Feasibility:** The potential for agricultural optimization and improved emergency response justifies development costs, as it will help save resources, lives, and potentially improve local economies in agriculture-dependent areas.

##### 2.1.3 Research and Data Collection

Data was gathered from:

- **Weather APIs** (such as OpenWeather or WeatherStack) for real-time weather updates and seasonal forecasts.
- **Agricultural Research Centers** for reliable crop selection and planning data based on weather patterns.
- **Healthcare Data** from government and health organizations to link seasonal weather trends with potential health risks.

## 2.2. SOFTWARE REQUIREMENT SPECIFICATION

### 2.2.1 Functional Requirements

- **Real-Time Weather Updates:** The app must provide real-time weather information, including temperature, precipitation, humidity, and other metrics.
- **Seasonal Forecasts for Agriculture:** The app should include detailed seasonal weather forecasts and crop recommendations based on these forecasts.
- **Health Risk Alerts:** The app should notify users about potential seasonal diseases based on weather conditions.
- **Emergency Assistance Feature:** The SOS button should function offline and send an SMS with GPS location to rescue teams in case of a natural disaster.

### 2.2.2 Non-Functional Requirements

- **Usability:** The app must be user-friendly with a clear interface, easy to navigate for all age groups.
- **Reliability:** It must provide accurate data from trusted sources and maintain a stable offline SOS functionality.
- **Scalability:** The app should be able to handle increased user load, especially during emergency situations.
- **Performance:** The app should load weather data quickly and efficiently, and the SOS feature must be responsive.

## 2.3. SOFTWARE REQUIREMENTS

### 2.3.1 Software Components

- **Frontend:** For the user interface, a mobile-friendly UI framework such as Flutter or React Native will ensure compatibility across Android and iOS devices.
- **Backend:** A cloud-based service (e.g., AWS, Google Cloud) to handle data processing and storage for weather data, health alerts, and user locations during emergencies.
- **APIs:** Weather APIs for real-time and forecast data, healthcare databases for disease alerts, and SMS/GPS APIs for offline SOS communication.
- **Data Storage:** A scalable, cloud-based database (e.g., Firebase, MySQL) to store user data, emergency logs, and historical weather data.

### 2.3.2 Security and Privacy Requirements

- **Data Encryption:** User data, including health alerts and SOS messages, must be encrypted to protect privacy.
- **User Consent:** User consent is required for location tracking and health alert notifications.
- **Access Control:** Access to sensitive data and emergency messages should be restricted to authorized rescue teams and administrators.

## 2.4. HARDWARE REQUIREMENTS

### 2.4.1 Mobile Device Requirements

AREO FORECAST is designed for mobile use, so users must have an Android or iOS smartphone.

The application should support most modern devices with:

- **GPS Functionality:** Necessary for accurate location tracking in SOS situations.
- **SMS Capabilities:** Required for offline emergency notifications in the event of network outages.

- **Internet Connection:** For general app use (weather updates, health alerts), although the SOS function should work offline.

### 2.4.2 Server Requirements

The backend requires robust cloud infrastructure to support real-time data processing and storage:

- **CPU and RAM Requirements:** The server must have adequate CPU and memory to handle large volumes of user requests, especially during peak times or emergencies.
- **Scalability:** The server should be scalable to accommodate spikes in usage.
- **Reliability:** The cloud provider should have backup options and high availability to ensure continuous service.

## 2.5 MODEL SELECTION AND ARCHITECTURE

### 2.5.1 Model Selection

AREO FORECAST follows a **Client-Server Model**:

- **Client Side (Mobile App):** Handles user interaction, displaying real-time weather data, seasonal forecasts, and health alerts. Also allows users to send SOS alerts during emergencies.
- **Server Side (Cloud Backend):** Processes requests, stores and retrieves weather and health data, and manages the SOS message handling system.

### 2.5.2 System Architecture

The proposed architecture for AREO FORECAST is a **Three-Tier Architecture**:

- **Presentation Layer:** The mobile application interface, developed in Flutter or React Native, provides users with weather updates, health alerts, and an SOS button.
- **Application Layer (Middleware):** The backend processes data from weather and health APIs, generates crop recommendations, and manages emergency messages.

- **Data Layer:** A cloud database stores user data, health alert history, weather forecasts, and SOS logs, ensuring data integrity and quick retrieval.

### 2.5.3 Emergency Module Architecture

The emergency system is built on a **Resilient Architecture** to ensure availability during disasters:

- **Offline SMS Trigger:** The SOS button sends a pre-defined SMS containing the user's GPS location directly to local rescue services.
- **Location Tracking:** Integrates with GPS to provide real-time user location, updating as the user moves, ensuring accurate and timely rescue operations.

### 2.5.4 API Integration Architecture

**AREO FORECAST** relies on various third-party APIs for comprehensive data:

- **Weather API Integration:** Fetches real-time weather data and seasonal forecasts to populate the app.
- **Health Data API:** Retrieves relevant health alerts based on seasonal conditions.
- **SMS and GPS APIs:** Provides offline SOS functionality through SMS and GPS, critical for the app's emergency feature.

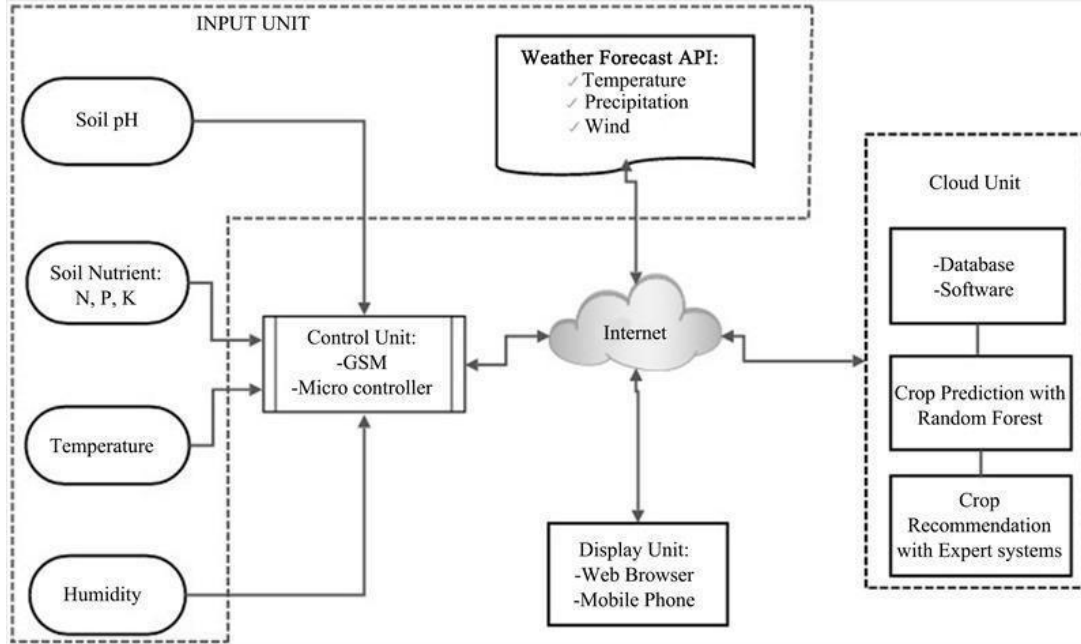
## CHAPTER-3

### DESIGN

#### 3.1 INTRODUCTION

The AREO FORECAST application is a comprehensive, user-centric weather and disaster response platform aimed at delivering accurate weather forecasts, agricultural recommendations, health alerts, and emergency assistance. AREO FORECAST utilizes various data sources and AI-based systems to serve a diverse user base, particularly farmers and residents in disaster-prone regions. This design document outlines the core components, data handling, and algorithms involved in developing AREO FORECAST to ensure it meets the needs of its target audience.

#### 3.2 DFD/ER/UML Diagram



### 3.3 DATA SET DESCRIPTIONS

#### 3.3.1 Weather Data

- **Source:** Weather APIs (e.g., OpenWeather, WeatherStack) provide real-time weather data, seasonal forecasts, temperature, humidity, precipitation, and wind speed information.
- **Purpose:** Real-time weather updates help users plan daily activities, while seasonal forecasts enable farmers to select crops suitable for upcoming weather conditions.

#### 3.3.2 Agricultural Data

- **Source:** Agricultural databases and research centers provide data on optimal crops for specific weather patterns, soil conditions, and regional factors.
- **Purpose:** Seasonal forecasts are combined with agricultural data to recommend crop choices that align with projected weather conditions, maximizing yield and minimizing losses.

#### 3.3.3 Health and Disease Data

- **Source:** Health organizations' databases and research centers offer insights into diseases linked to certain weather conditions, such as flu in colder months or vectorborne diseases in rainy seasons.
- **Purpose:** The application alerts users about potential health risks based on the current and forecasted weather, promoting preventive healthcare practices.

#### 3.3.4 Emergency Response Data

- **Source:** Internal database and local government agencies' APIs for rescue operations.
- **Purpose:** Data is used to activate the offline SOS feature, which notifies local rescue teams of a user's location and emergency status during a disaster, even without internet access.

## 3.4 DATA PREPROCESSING TECHNIQUES

### 3.4.1 Data Cleaning

- **Objective:** Ensure data from various sources (weather APIs, agricultural and health databases) is accurate, up-to-date, and free from errors.
- **Methods:** Remove duplicates, handle missing values, and standardize units (e.g., temperature in Celsius or Fahrenheit) to maintain data consistency.

### 3.4.2 Data Transformation

- **Objective:** Convert raw data into a format suitable for analysis and predictions.
- **Methods:** Normalize weather data (temperature, humidity, etc.) and encode categorical variables, such as disease risk levels or crop types, for use in machine learning algorithms.

### 3.4.3 Data Integration

- **Objective:** Combine data from multiple sources (weather, agricultural, health) to provide comprehensive information for each application feature.
- **Methods:** Merge datasets based on location and time to synchronize weather conditions, crop recommendations, and health alerts for a specific area.

### 3.4.4 Data Filtering

- **Objective:** Filter data to match specific user preferences and location requirements.
- **Methods:** Utilize filters based on user location, season, and crop preference to display relevant data, enhancing user experience and app efficiency.

## 3.5. METHODS & ALGORITHMS

### 3.5.1 Real-Time Weather and Forecasting Module

- **Algorithm:** NWP Algorithm(Numerical Weather Prediction)

- **Function:** Analyzes Atmospheric data through mathematical models to forecast weather conditions. It computes changes in weather variables like temperature, pressure, and wind over time to predict future weather patterns.

### 3.5.2 Health Risk Prediction Module

- **Algorithm:** Naive Bayes Classification
- **Function:** This module leverages Naive Bayes to analyze weather data and provide disease predictions based on historical health patterns associated with specific weather conditions.

### 3.5.3 Crop Recommendation System

- **Algorithm:** K-Nearest Neighbors (KNN)
- **Function:** The KNN algorithm matches current and forecasted weather data with crop suitability profiles, recommending the best crops for the upcoming season based on weather patterns and soil conditions.

### 3.5.4 Emergency Assistance Module

- **Algorithm:** Location-Based Routing & Offline SMS Protocol
- **Function:** This module includes offline SOS functionality, which uses GPS coordinates and SMS protocols to send an emergency alert to local authorities and rescue teams. Location-based routing optimizes rescue team dispatch by calculating the fastest route to the user.

### 3.5.5 Budget Optimization (Optional Extension)

- **Algorithm:** Linear Programming
- **Function:** If a budget optimization system is required for user-specific agricultural planning, linear programming can be used to suggest optimal crop investment strategies based on expected yield and costs.

### 3.5.6 Cultural Personalization Module (Future Enhancement)

- **Algorithm:** Rule-Based Filtering or Custom AI Model
- **Function:** If expanding for cultural personalization, this module would adapt health recommendations, crop suggestions, and other alerts based on local cultural practices and user preferences.

## **CHAPTER-4**

### **DEPLOYMENT AND RESULTS**

#### **4.1 INTRODUCTION**

This section covers the deployment and results of AREO FORECAST, an AI-driven weather and emergency response application designed to provide real-time weather updates, agricultural guidance, health risk alerts, and emergency assistance features for users, particularly in rural and disaster-prone areas. The deployment phase involves source code management, model training, testing, validation, and user interface development. Evaluation metrics are included to assess the effectiveness of each feature in meeting user needs.

#### **4.2. SOURCE CODE**

##### **4.2.1 Code Repository and Version Control**

- **Platform:** GitHub or GitLab for collaborative code management and version control.
- **Description:** All source code for the weather forecasting, crop recommendation, health alert, and emergency response modules is organized in a structured repository. Each module is managed with branching and versioning for effective updates.

##### **4.2.2 Code Documentation**

- **Codebase Structure:** The codebase is modular, separating the AI engine, databases, and external services into clear components.
- **Documentation:** Each module includes in-code comments and separate documentation files, describing functions, API calls, data handling, and dependencies.

#### **4.3. MODEL IMPLEMENTATION AND TRAINING**

##### **4.3.1 Data Loading and Preprocessing**

- **Description:** Weather data, health statistics, and agricultural records are gathered and cleaned for uniform formatting and consistency.

- **Tools:** Libraries such as Pandas, NumPy, and scikit-learn are used to preprocess data before feeding it into models.

#### 4.3.2 Model Training

- **Weather Prediction Model:** A time-series model trained to forecast weather changes for upcoming seasons.
- **Crop Recommendation Model:** Machine learning model (e.g., decision trees) trained with seasonal data to recommend crop varieties.
- **Health Alert Model:** Naive Bayes or logistic regression model trained to predict disease outbreaks based on weather patterns.

#### 4.3.3 Training Environment

- **Platform:** AWS SageMaker or Google Colab for large-scale training with access to GPUs and TPUs if necessary.
- **Data Storage:** Trained models and preprocessed data stored in cloud storage, with regular backups.

### 4.4 MODEL EVALUATION METRICS

#### 4.4.1 Weather Prediction Model

**Metrics:** Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to measure forecast accuracy.

#### 4.4.2 Crop Recommendation Model

**Metrics:** Accuracy and F1 Score, indicating the reliability of crop recommendations per forecasted seasonal conditions.

#### 4.4.3 Health Alert Model

**Metrics:** Precision, Recall, and Accuracy, to assess correct prediction of health risks based on weather trends.

#### 4.4.4 Emergency Response Feature

**Metrics:** Success rate of emergency alerts in offline mode, GPS accuracy, and response time from the emergency module.

## 4.5 MODEL DEPLOYMENT: TESTING AND VALIDATION

### 4.5.1 Functional Testing

**Description:** Each module, including weather updates, crop and health recommendations, and emergency features, is tested for standalone functionality.

### 4.5.2 Integration Testing

**Description:** Validates data flow and interdependence between modules, ensuring, for example, that the forecast module feeds correctly into crop and health alert modules.

### 4.5.3 User Acceptance Testing (UAT)

**Test Group:** Farmers, health workers, and users in disaster-prone areas are involved to validate the app's usability and functionality.

**Feedback Collection:** User feedback gathered to identify potential issues and improvements.

### 4.5.4 Deployment Environment

- **Platform:** Cloud-based deployment on AWS or Azure for scalability, with backend managed through Django or Flask.
- **Backend Infrastructure:** REST API setup to handle data requests, storage, and responses across the modules.

## 4.6. WEB GUI DEVELOPMENT

### 4.6.1 User Interface Design

- **Design Goals:** Simplify access to weather, health, crop, and emergency modules with an intuitive and visually clear interface.
- **Tools:** Developed using frontend frameworks like React or Angular, with responsive design for compatibility across devices.

### 4.6.2 User Input Layer

- **Functionality:** Users input location and preferences, such as crop type or health alert preferences, enabling the system to personalize recommendations.

- **Input Components:** Text fields for location, dropdowns for crop and health alerts, and an emergency button.

#### 4.6.3 Output Layer

- **Description:** Displays personalized weather forecasts, crop suggestions, health alerts, and the emergency status for easy access.
- **Components:** Dashboard view for real-time updates, notification panel for alerts, and visual indicator for emergency functionality.

#### 4.6.4 Offline Capability

- **Purpose:** Ensure critical features, such as the emergency button, work without an internet connection.
- **Implementation:** Uses GPS and SMS for offline operation, allowing users to send location details and emergency alerts without connectivity.

## UI PROGRAM CODE :

```
import React from 'react';
import { Link, useLocation, useNavigate } from 'react-router-dom';
import { Stethoscope, Sprout, PhoneCall, Sun, Moon, LogOut } from 'lucide-react';
import { useTheme } from '../context/ThemeContext';

export default function Navigation() {
  const location = useLocation();
  const navigate = useNavigate();
  const { theme, toggleTheme } = useTheme();

  const isActive = (path: string) => location.pathname === path;

  const handleSignOut = () => {
    navigate('/');
  };

  return (
    <nav className="bg-white/90 dark:bg-gray-800/90 shadow-md backdrop-blur-sm mb-8 transition-colors duration-200">
      <div className="container mx-auto px-4">
        <div className="flex items-center justify-between py-4">
          <div className="flex items-center space-x-8">
            <Link
              to="/medical"
              className={`flex items-center space-x-2 px-4 py-2 rounded-lg transition-colors`}
              isActive('/medical')
              ? 'bg-blue-700 text-white'
            >

```

```

        : 'text-gray-600 dark:text-gray-300 hover:bg-blue-50 dark: hover:bg-gray-700'
      }}
    >
    <Stethoscope size={20} />
    <span>Medical</span>
  </Link>

  <Link
    to="/agri-care"
    className={`flex items-center space-x-2 px-4 py-2 rounded-lg transition-colors
$ {
      isActive('/agri-care')
        ? 'bg-blue-700 text-white'
        : 'text-gray-600 dark:text-gray-300 hover:bg-blue-50 dark: hover:bg-gray-700'
      }}
    >
    <Sprout size={20} />
    <span>Agri Care</span>
  </Link>

  <Link
    to="/sos"
    className={`flex items-center space-x-2 px-4 py-2 rounded-lg transition-colors
$ {
      isActive('/sos')
        ? 'bg-blue-700 text-white'
        : 'text-gray-600 dark:text-gray-300 hover:bg-blue-50 dark: hover:bg-gray-700'
      }}
    >
    <PhoneCall size={20} />

```

```

        <span>SOS</span>
      </Link>
    </div>

    <div className="flex items-center space-x-4">
      <button
        onClick={toggleTheme}
        className="p-2 rounded-lg text-gray-600 dark:text-gray-300 hover:bg-gray-100
dark:hover:bg-gray-700 transition-colors"
        aria-label="Toggle theme"
      >
        {theme === 'light' ? <Moon size={20} /> : <Sun size={20} />}
      </button>

      <button
        onClick={handleSignOut}
        className="flex items-center space-x-2 px-4 py-2 rounded-lg text-gray-600
dark:text-gray-300 hover:bg-gray-100 dark:hover:bg-gray-700 transition-colors"
      >
        <LogOut size={20} />
        <span>Sign Out</span>
      </button>
    </div>
  </div>
</div>
</nav>
);
}
import React, { useState } from 'react';
import Navigation from '../components/Navigation';

```

```

const seasonalCrops = {
  summer: [
    { name: 'Watermelon', conditions: ['High temperature', 'Well-drained soil'] },
    { name: 'Cucumber', conditions: ['Regular watering', 'Full sunlight'] },
    { name: 'Tomatoes', conditions: ['Warm soil', 'Support structures'] },
    { name: 'Muskmelon', conditions: ['Sandy loam soil', 'Regular irrigation'] },
    { name: 'Okra', conditions: ['Warm weather', 'Well-drained soil'] }
  ],
  winter: [
    { name: 'Wheat', conditions: ['Cool temperature', 'Moderate irrigation'] },
    { name: 'Peas', conditions: ['Cool weather', 'Moist soil'] },
    { name: 'Carrots', conditions: ['Well-drained soil', 'Cool climate'] },
    { name: 'Spinach', conditions: ['Cool temperatures', 'Rich soil'] },
    { name: 'Cauliflower', conditions: ['Cool weather', 'Regular watering'] }
  ],
  monsoon: [
    { name: 'Rice', conditions: ['Heavy rainfall', 'Waterlogged conditions'] },
    { name: 'Corn', conditions: ['Warm temperature', 'High humidity'] },
    { name: 'Soybeans', conditions: ['Moderate rainfall', 'Well-drained soil'] },
    { name: 'Green Gram', conditions: ['Moderate rainfall', 'Light soil'] },
    { name: 'Black Gram', conditions: ['Humid climate', 'Well-drained soil'] }
  ]
};

```

```

const allCrops = [
  ...seasonalCrops.summer.map(crop => ({ ...crop, season: 'summer' })),
  ...seasonalCrops.winter.map(crop => ({ ...crop, season: 'winter' })),
  ...seasonalCrops.monsoon.map(crop => ({ ...crop, season: 'monsoon' }))
]

```

```
];
```

```
export default function AgriCare() {  
  const [selectedSeason, setSelectedSeason] = useState("");  
  const [searchCrop, setSearchCrop] = useState("");  
  const [searchResult, setSearchResult] = useState<{  
    found: boolean;  
    crop?: typeof allCrops[0];  
    message: string;  
  } | null>(null);
```

```
  const handleCropSearch = () => {  
    if (!selectedSeason || !searchCrop) {  
      setSearchResult(null);  
      return;  
    }
```

```
    const crop = allCrops.find(  
      c => c.name.toLowerCase() === searchCrop.toLowerCase()  
    );
```

```
    if (!crop) {  
      setSearchResult({  
        found: false,  
        message: 'Crop not found in our database.'  
      });  
      return;  
    }
```

```

const isSuitableForSeason = crop.season === selectedSeason;

setSearchResult({
  found: true,
  crop,
  message: isSuitableForSeason
    ? `${crop.name} is suitable for farming in ${selectedSeason} season!`
    : `${crop.name} is not recommended for ${selectedSeason} season. It's better suited
for ${crop.season} season.`
});
};

```

```

return (
  <div className="min-h-[calc(100vh-80px)]">
    <Navigation />

    <main className="container mx-auto px-4 py-8">
      <div className="card dark:bg-gray-800">
        <h2 className="text-2xl font-bold text-gray-800 dark:text-white mb-6">Seasonal
Crop Guide</h2>

        <div className="grid grid-cols-1 md:grid-cols-2 gap-6 mb-8">
          <div>
            <select
              className="input-field dark:bg-gray-700 dark:text-white dark:border-gray-600"
              value={selectedSeason}
              onChange={(e) => setSelectedSeason(e.target.value)}
            >
              <option value="">Select a season</option>
              <option value="summer">Summer</option>
              <option value="winter">Winter</option>

```

```

      <option value="monsoon">Monsoon</option>
    </select>
  </div>

```

```

<div className="flex space-x-4">
  <input
    type="text"
    placeholder="Search for a specific crop..."
    className="input-field flex-1 dark:bg-gray-700 dark:text-white dark:border-
gray-600"
    value={searchCrop}
    onChange={(e) => setSearchCrop(e.target.value)}
  />
  <button
    onClick={handleCropSearch}
    className="btn-primary dark:bg-blue-600 dark:hover:bg-blue-700"
    disabled={!selectedSeason || !searchCrop}
  >
    Check Suitability
  </button>
</div>
</div>

```

```

{searchResult && (
  <div className={`p-4 rounded-lg mb-8 ${
    searchResult.found && searchResult.crop?.season === selectedSeason
      ? 'bg-green-50 dark:bg-green-900/20 text-green-700 dark:text-green-400'
      : 'bg-yellow-50 dark:bg-yellow-900/20 text-yellow-700 dark:text-yellow-400'
    }}`>
    <p className="text-lg">{searchResult.message}</p>

```

```

    {searchResult.found && searchResult.crop && (
      <div className="mt-4">
        <h4 className="font-medium mb-2">Growing Conditions:</h4>
        <ul className="list-disc list-inside">
          {searchResult.crop.conditions.map((condition, idx) => (
            <li key={idx}>{condition}</li>
          ))}
        </ul>
      </div>
    )}
  </div>
)}

{selectedSeason && (
  <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
    {seasonalCrops[selectedSeason as keyof typeof seasonalCrops].map((crop, index)
=> (
      <div key={index} className="p-4 bg-white dark:bg-gray-700 rounded-lg
shadow">
        <h3 className="text-xl font-semibold text-blue-700 dark:text-blue-400 mb-
3">{crop.name}</h3>

        <div>
          <h4 className="font-medium text-gray-700 dark:text-gray-300 mb-
2">Growing Conditions:</h4>
          <ul className="list-disc list-inside text-gray-600 dark:text-gray-400">
            {crop.conditions.map((condition, idx) => (
              <li key={idx}>{condition}</li>
            ))}
          </ul>
        </div>

```

```

        </div>

    )}
</div>

})
</div>
</main>
</div>

);
}
import React, { useState, useEffect } from 'react';
import { Search, MapPin, ThermometerSun, Wind, Droplets } from 'lucide-react';
import { MapContainer, TileLayer, Marker, Popup } from 'react-leaflet';
import { Icon } from 'leaflet';
import Navigation from '../components/Navigation';
import 'leaflet/dist/leaflet.css';

// Fix for default marker icon
const customIcon = new Icon({
  iconUrl: 'https://unpkg.com/leaflet@1.9.4/dist/images/marker-icon.png',
  iconRetinaUrl: 'https://unpkg.com/leaflet@1.9.4/dist/images/marker-icon-2x.png',
  shadowUrl: 'https://unpkg.com/leaflet@1.9.4/dist/images/marker-shadow.png',
  iconSize: [25, 41],
  iconAnchor: [12, 41],
  popupAnchor: [1, -34],
  shadowSize: [41, 41]
});

const indianCities = [
  { name: 'Hyderabad', area: 'Gachibowli', state: 'Telangana', temp: 32, weather: 'Mostly Sunny', humidity: 65, windSpeed: 12, lat: 17.4435, lng: 78.3772 },

```

{ name: 'Hyderabad', area: 'Hi-Tech City', state: 'Telangana', temp: 31, weather: 'Clear Sky', humidity: 62, windSpeed: 10, lat: 17.4456, lng: 78.3801 },

{ name: 'Hyderabad', area: 'Banjara Hills', state: 'Telangana', temp: 32, weather: 'Sunny', humidity: 60, windSpeed: 11, lat: 17.4156, lng: 78.4347 },

{ name: 'Hyderabad', area: 'Jubilee Hills', state: 'Telangana', temp: 31, weather: 'Clear', humidity: 63, windSpeed: 9, lat: 17.4239, lng: 78.4075 },

{ name: 'Hyderabad', area: 'Secunderabad', state: 'Telangana', temp: 33, weather: 'Mostly Sunny', humidity: 58, windSpeed: 14, lat: 17.4399, lng: 78.4983 },

{ name: 'Hyderabad', area: 'Madhapur', state: 'Telangana', temp: 31, weather: 'Clear Sky', humidity: 64, windSpeed: 11, lat: 17.4484, lng: 78.3908 },

{ name: 'Hyderabad', area: 'Kukatpally', state: 'Telangana', temp: 32, weather: 'Sunny', humidity: 61, windSpeed: 13, lat: 17.4849, lng: 78.4138 },

{ name: 'Hyderabad', area: 'Ameerpet', state: 'Telangana', temp: 32, weather: 'Clear', humidity: 59, windSpeed: 12, lat: 17.4374, lng: 78.4487 },

{ name: 'Hyderabad', area: 'Begumpet', state: 'Telangana', temp: 31, weather: 'Mostly Clear', humidity: 66, windSpeed: 10, lat: 17.4444, lng: 78.4667 },

{ name: 'Hyderabad', area: 'Mehdipatnam', state: 'Telangana', temp: 33, weather: 'Sunny', humidity: 57, windSpeed: 15, lat: 17.3933, lng: 78.4311 },

{ name: 'Hyderabad', area: 'Dilsukhnagar', state: 'Telangana', temp: 32, weather: 'Clear Sky', humidity: 63, windSpeed: 11, lat: 17.3686, lng: 78.5242 },

{ name: 'Hyderabad', area: 'LB Nagar', state: 'Telangana', temp: 32, weather: 'Mostly Sunny', humidity: 62, windSpeed: 12, lat: 17.3457, lng: 78.5477 },

{ name: 'Mumbai', state: 'Maharashtra', temp: 31, weather: 'Humid', humidity: 75, windSpeed: 18, lat: 19.0760, lng: 72.8777 },

{ name: 'Delhi', state: 'Delhi', temp: 28, weather: 'Clear Sky', humidity: 55, windSpeed: 16, lat: 28.6139, lng: 77.2090 },

{ name: 'Bangalore', state: 'Karnataka', temp: 27, weather: 'Partly Cloudy', humidity: 68, windSpeed: 14, lat: 12.9716, lng: 77.5946 },

{ name: 'Chennai', state: 'Tamil Nadu', temp: 30, weather: 'Sunny', humidity: 70, windSpeed: 15, lat: 13.0827, lng: 80.2707 },

{ name: 'Kolkata', state: 'West Bengal', temp: 29, weather: 'Scattered Clouds', humidity: 72, windSpeed: 13, lat: 22.5726, lng: 88.3639 },

{ name: 'Pune', state: 'Maharashtra', temp: 28, weather: 'Clear', humidity: 65, windSpeed: 12, lat: 18.5204, lng: 73.8567 },

```

    { name: 'Ahmedabad', state: 'Gujarat', temp: 33, weather: 'Hot', humidity: 58, windSpeed:
17, lat: 23.0225, lng: 72.5714 }
];

```

```

export default function Home() {
  const [currentTime, setCurrentTime] = useState(new Date());
  const [searchQuery, setSearchQuery] = useState("");
  const [selectedCity, setSelectedCity] = useState(indianCities[0]);
  const [showResults, setShowResults] = useState(false);

  useEffect(() => {
    const timer = setInterval(() => setCurrentTime(new Date()), 1000);
    return () => clearInterval(timer);
  }, []);

  const filteredCities = indianCities.filter(city =>
    (city.name.toLowerCase().includes(searchQuery.toLowerCase()) ||
    (city.area?.toLowerCase() || "").includes(searchQuery.toLowerCase()) ||
    city.state.toLowerCase().includes(searchQuery.toLowerCase()))
  );

  const handleCitySelect = (city: typeof indianCities[0]) => {
    setSelectedCity(city);
    setSearchQuery("");
    setShowResults(false);
  };

  const getWeatherBackground = (weather: string, temp: number) => {
    if (weather.toLowerCase().includes('sunny')) {

```

```

    return 'bg-gradient-to-r from-yellow-50 to-orange-50 dark:from-yellow-900/20
dark:to-orange-900/20';
  }
  if (weather.toLowerCase().includes('cloud')) {
    return 'bg-gradient-to-r from-gray-50 to-blue-50 dark:from-gray-900/20 dark:to-blue-
900/20';
  }
  if (weather.toLowerCase().includes('clear')) {
    return 'bg-gradient-to-r from-blue-50 to-indigo-50 dark:from-blue-900/20 dark:to-
indigo-900/20';
  }
  if (temp >= 30) {
    return 'bg-gradient-to-r from-orange-50 to-red-50 dark:from-orange-900/20 dark:to-
red-900/20';
  }
  return 'bg-gradient-to-r from-blue-50 to-green-50 dark:from-blue-900/20 dark:to-green-
900/20';
};

```

```

const getWeatherColor = (temp: number) => {
  if (temp >= 35) return 'text-red-600 dark:text-red-400';
  if (temp >= 30) return 'text-orange-500 dark:text-orange-400';
  if (temp >= 25) return 'text-yellow-500 dark:text-yellow-400';
  return 'text-blue-500 dark:text-blue-400';
};

```

```

return (
  <div className="min-h-[calc(100vh-80px)]">
    <Navigation />

    <main className="container mx-auto px-4 py-8">

```

```

<div className="grid grid-cols-1 md:grid-cols-2 gap-8">
  <div className={`card dark:bg-gray-800/50
    ${getWeatherBackground(selectedCity.weather, selectedCity.temp)} `}>
    <div className="flex items-center justify-between mb-6">
      <div>
        <h2 className="text-2xl font-bold text-gray-800 dark:text-white">Current
Weather</h2>
        <p className="text-gray-600 dark:text-gray-300">
          {currentTime.toLocaleDateString('en-IN', {
            weekday: 'long',
            year: 'numeric',
            month: 'long',
            day: 'numeric'
          })}
        </p>
        <p className="text-gray-500 dark:text-gray-400">
          {currentTime.toLocaleTimeString('en-IN')}
        </p>
      </div>
      <ThermometerSun size={32} className="text-blue-700 dark:text-blue-400" />
    </div>

    <div className="space-y-6">
      <div className="flex items-center space-x-2">
        <MapPin className="text-blue-700 dark:text-blue-400" />
        <span className="text-gray-700 dark:text-gray-200">
          {selectedCity.area}
          ? `${selectedCity.area}, ${selectedCity.name}`
          : `${selectedCity.name}`, {selectedCity.state}
        </span>
      </div>
    </div>
  </div>
</div>

```

```

</div>

<div className="flex items-center justify-between">
  <div>
    <div className={`text-6xl font-bold
    ${getWeatherColor(selectedCity.temp)} `}>
      {selectedCity.temp} °C
    </div>
    <p className="text-gray-600 dark:text-gray-300 mt-
    2">{selectedCity.weather}</p>
  </div>

  <div className="space-y-4">
    <div className="flex items-center space-x-2 text-gray-600 dark:text-gray-
    300">
      <Droplets size={20} />
      <span>Humidity: {selectedCity.humidity}%</span>
    </div>

    <div className="flex items-center space-x-2 text-gray-600 dark:text-gray-
    300">
      <Wind size={20} />
      <span>Wind: {selectedCity.windSpeed} km/h</span>
    </div>
  </div>
</div>
</div>
</div>
</div>

<div className="card dark:bg-gray-800">
  <div className="mb-6">
    <div className="relative">

```

```

<input
  type="text"
  placeholder="Search cities or areas in India..."
  className="input-field pl-10 dark:bg-gray-700 dark:text-white dark:border-
gray-600"
  value={searchQuery}
  onChange={(e) => {
    setSearchQuery(e.target.value);
    setShowResults(true);
  }}
  onFocus={() => setShowResults(true)}
/>
<Search className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-
400" />

```

```

{showResults && searchQuery && (
  <div className="absolute z-10 w-full mt-1 bg-white dark:bg-gray-700
rounded-lg shadow-lg max-h-60 overflow-auto">
    {filteredCities.length > 0 ? (
      filteredCities.map((city, index) => (
        <div
          key={index}
          className="p-3 hover:bg-blue-50 dark:hover:bg-gray-600 cursor-pointer
flex items-center justify-between"
          onClick={() => handleCitySelect(city)}
        >
          <div>
            <div className="font-medium dark:text-white">
              {city.area}
              ? `${city.area}, ${city.name}`
              : city.name}

```

```

        </div>

        <div className="text-sm text-gray-600 dark:text-gray-
300">{city.state}</div>

    </div>

    <div className={`font-semibold
${getWeatherColor(city.temp)}`} >{city.temp}°C</div>

    </div>

    ))

    ): (

        <div className="p-3 text-gray-600 dark:text-gray-300">No locations
found</div>

    )}

</div>

)}

</div>

</div>

<div className="h-[300px] rounded-lg overflow-hidden">

    <MapContainer

        center={[selectedCity.lat, selectedCity.lng]}

        zoom={12}

        style={{ height: '100%', width: '100%' }}

        key={` ${selectedCity.lat}-${selectedCity.lng}`}

    >

        <TileLayer

            attribution='&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

            url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"

        />

        <Marker position={[selectedCity.lat, selectedCity.lng]} icon={customIcon}>

        <Popup>

```

```

        {selectedCity.area
        ? `${selectedCity.area}, ${selectedCity.name}`
        : selectedCity.name}
      <br />
      Temperature: {selectedCity.temp}°C
    <br />
    Weather: {selectedCity.weather}
  </Popup>
</Marker>
</MapContainer>
</div>
</div>
</div>
</main>
</div>
);
}

import React, { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { CloudSun } from 'lucide-react';

export default function Login() {
  const [identifier, setIdentifier] = useState("");
  const [password, setPassword] = useState("");
  const navigate = useNavigate();

  const handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    navigate('/home');
  }

```

```
};
```

```
return (
```

```
<div className="min-h-[calc(100vh-80px)] flex items-center justify-center px-4">  
  <div className="card max-w-md w-full">  
    <div className="flex flex-col items-center mb-8">  
      <CloudSun size={48} className="text-blue-700" />  
      <h1 className="text-3xl font-bold text-blue-700 mt-2">AERO FORECAST</h1>  
      <p className="text-gray-600 mt-2">Welcome back</p>  
    </div>
```

```
<form onSubmit={handleSubmit} className="space-y-4">  
  <div>  
    <label className="block text-gray-700 mb-2">Email or Phone Number</label>  
    <input  
      type="text"  
      className="input-field"  
      value={identifier}  
      onChange={(e) => setIdentifier(e.target.value)}  
      required  
    />  
  </div>
```

```
<div>  
  <label className="block text-gray-700 mb-2">Password</label>  
  <input  
    type="password"  
    className="input-field"  
    value={password}
```

```

        onChange={(e) => setPassword(e.target.value)}
        required
      />
    </div>

    <button type="submit" className="btn-primary w-full">
      Log In
    </button>
  </form>

  <p className="text-center mt-6 text-gray-600">
    Don't have an account?{' '}
    <Link to="/" className="text-blue-700 hover:underline">
      Sign up
    </Link>
  </p>
</div>
</div>

);
}

import React, { useState } from 'react';
import Navigation from '../components/Navigation';

const seasonalDiseases = {
  summer: [
    { name: 'Heat Exhaustion', medicines: ['Oral rehydration solutions'], precautions: ['Stay hydrated', 'Avoid direct sunlight'] },
    { name: 'Food Poisoning', medicines: ['Antibiotics', 'ORS'], precautions: ['Proper food storage', 'Hand hygiene'] }
  ],

```

```

winter: [
  { name: 'Common Cold', medicines: ['Antihistamines', 'Decongestants'], precautions:
['Keep warm', 'Boost immunity'] },
  { name: 'Flu', medicines: ['Antiviral medications'], precautions: ['Flu vaccination',
'Regular hand washing'] }
],
monsoon: [
  { name: 'Dengue', medicines: ['Paracetamol'], precautions: ['Mosquito control', 'Clean
surroundings'] },
  { name: 'Malaria', medicines: ['Antimalarial drugs'], precautions: ['Use mosquito nets',
'Remove stagnant water'] }
]
};

```

```

export default function Medical() {
  const [selectedSeason, setSelectedSeason] = useState("");

  return (
    <div className="min-h-[calc(100vh-80px)]">
      <Navigation />

      <main className="container mx-auto px-4 py-8">
        <div className="card">
          <h2 className="text-2xl font-bold text-gray-800 mb-6">Seasonal Health
Guide</h2>

          <select
            className="input-field mb-6"
            value={selectedSeason}
            onChange={(e) => setSelectedSeason(e.target.value)}
          >

```

```

<option value="">Select a season</option>
<option value="summer">Summer</option>
<option value="winter">Winter</option>
<option value="monsoon">Monsoon</option>
</select>

{selectedSeason && (
  <div className="space-y-6">
    {seasonalDiseases[selectedSeason as keyof typeof
seasonalDiseases].map((disease, index) => (
      <div key={index} className="p-4 bg-white rounded-lg shadow">
        <h3 className="text-xl font-semibold text-blue-700 mb-
3">{disease.name}</h3>

        <div className="space-y-2">
          <div>
            <h4 className="font-medium text-gray-700">Recommended
Medicines:</h4>
            <ul className="list-disc list-inside text-gray-600">
              {disease.medicines.map((medicine, idx) => (
                <li key={idx}>{medicine}</li>
              ))}
            </ul>
          </div>

          <div>
            <h4 className="font-medium text-gray-700">Precautions:</h4>
            <ul className="list-disc list-inside text-gray-600">
              {disease.precautions.map((precaution, idx) => (
                <li key={idx}>{precaution}</li>

```

```

        )))}
      </ul>
    </div>
  </div>
</div>
  )))}
</div>
  })
</div>
</main>
</div>
);
}
import React, { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { CloudSun } from 'lucide-react';

export default function SignUp() {
  const [identifier, setIdentifier] = useState("");
  const [password, setPassword] = useState("");
  const navigate = useNavigate();

  const handleSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    navigate('/home');
  };

  return (
    <div className="min-h-[calc(100vh-80px)] flex items-center justify-center px-4">

```

```

<div className="card max-w-md w-full">
  <div className="flex flex-col items-center mb-8">
    <CloudSun size={48} className="text-blue-700" />
    <h1 className="text-3xl font-bold text-blue-700 mt-2">AERO FORECAST</h1>
    <p className="text-gray-600 mt-2">Create your account</p>
  </div>

```

```

<form onSubmit={handleSubmit} className="space-y-4">
  <div>
    <label className="block text-gray-700 mb-2">Email or Phone Number</label>
    <input
      type="text"
      className="input-field"
      value={identifier}
      onChange={(e) => setIdentifier(e.target.value)}
      required
    />
  </div>

```

```

  <div>
    <label className="block text-gray-700 mb-2">Password</label>
    <input
      type="password"
      className="input-field"
      value={password}
      onChange={(e) => setPassword(e.target.value)}
      required
    />
  </div>

```

```

    <button type="submit" className="btn-primary w-full">
      Sign Up
    </button>
  </form>

  <p className="text-center mt-6 text-gray-600">
    Already have an account? { ' '}
    <Link to="/login" className="text-blue-700 hover:underline">
      Log in
    </Link>
  </p>
</div>
</div>
);
}

import React from 'react';
import { Phone, AlertTriangle } from 'lucide-react';
import Navigation from '../components/Navigation';

export default function SOS() {
  return (
    <div className="min-h-[calc(100vh-80px)]">
      <Navigation />

      <main className="container mx-auto px-4 py-8">
        <div className="card">
          <div className="flex items-center space-x-3 mb-6">
            <AlertTriangle size={32} className="text-red-600" />

```

```

    <h2 className="text-2xl font-bold text-gray-800">Emergency Services</h2>
  </div>

  <div className="space-y-6">
    <div className="p-6 bg-red-50 rounded-lg border border-red-200">
      <h3 className="text-xl font-semibold text-red-700 mb-4">Disaster Management Helpline</h3>
      <div className="flex items-center space-x-3">
        <Phone className="text-red-600" />
        <a href="tel:112" className="text-2xl font-bold text-red-600 hover:underline">112</a>
      </div>
      <p className="mt-2 text-gray-600">National Emergency Number (24/7 Available)</p>
    </div>

    <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
      <div className="p-4 bg-white rounded-lg shadow">
        <h4 className="font-semibold text-gray-800 mb-2">Fire Emergency</h4>
        <a href="tel:101" className="text-blue-700 hover:underline flex items-center space-x-2">
          <Phone size={18} />
          <span>101</span>
        </a>
      </div>

      <div className="p-4 bg-white rounded-lg shadow">
        <h4 className="font-semibold text-gray-800 mb-2">Police</h4>
        <a href="tel:100" className="text-blue-700 hover:underline flex items-center space-x-2">
          <Phone size={18} />

```

```

        <span>100</span>
      </a>
    </div>

    <div className="p-4 bg-white rounded-lg shadow">
      <h4 className="font-semibold text-gray-800 mb-2">Ambulance</h4>
      <a href="tel:102" className="text-blue-700 hover:underline flex items-center
space-x-2">
        <Phone size={18} />
        <span>102</span>
      </a>
    </div>

    <div className="p-4 bg-white rounded-lg shadow">
      <h4 className="font-semibold text-gray-800 mb-2">Natural Disaster</h4>
      <a href="tel:108" className="text-blue-700 hover:underline flex items-center
space-x-2">
        <Phone size={18} />
        <span>108</span>
      </a>
    </div>
  </div>
</div>
</main>
</div>

);
}
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';

```

```

import { CloudSun } from 'lucide-react';
import { ThemeProvider } from './context/ThemeContext';
import SignUp from './pages/SignUp';
import Login from './pages/Login';
import Home from './pages/Home';
import Medical from './pages/Medical';
import AgriCare from './pages/AgriCare';
import SOS from './pages/SOS';

function App() {
  return (
    <ThemeProvider>
      <Router>
        <div className="min-h-screen dark:bg-gray-900 transition-colors duration-200">
          <header className="bg-white/90 dark:bg-gray-800/90 shadow-md backdrop-blur-sm">
            <div className="container mx-auto px-4 py-4 flex items-center justify-between">
              <Link to="/" className="flex items-center space-x-2 text-2xl font-bold text-blue-700 dark:text-blue-400">
                <CloudSun size={32} />
                <span>AERO FORECAST</span>
              </Link>
            </div>
          </header>

          <Routes>
            <Route path="/" element={<SignUp />} />
            <Route path="/login" element={<Login />} />
            <Route path="/home" element={<Home />} />
            <Route path="/medical" element={<Medical />} />
          </Routes>
        </div>
      </Router>
    </ThemeProvider>
  );
}

```

```

        <Route path="/agri-care" element={<AgriCare />} />
        <Route path="/sos" element={<SOS />} />
    </Routes>
</div>
</Router>
</ThemeProvider>
);
}

export default App;
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer base {
    body {
        background-image: url('https://images.unsplash.com/photo-1505533321630-975218a5f66f?auto=format&fit=crop&q=80');
        background-size: cover;
        background-position: center;
        background-attachment: fixed;
        background-repeat: no-repeat;
    }
}

@layer components {
    .btn-primary {
        @apply bg-blue-700 text-white px-6 py-2 rounded-lg hover:bg-blue-800 transition-colors;
    }
}

```

```

.input-field {
  @apply w-full px-4 py-2 border border-gray-300 rounded-lg focus:outline-none
  focus:ring-2 focus:ring-blue-500;
}

.card {
  @apply bg-white/90 backdrop-blur-sm rounded-lg shadow-lg p-6;
}

/* Leaflet specific styles */
.leaflet-container {
  z-index: 10;
}

.leaflet-popup-content-wrapper {
  @apply bg-white dark:bg-gray-800 dark:text-white;
}

.leaflet-popup-tip {
  @apply bg-white dark:bg-gray-800;
}

import { StrictMode } from 'react';
import { createRoot } from 'react-dom/client';
import App from './App.tsx';
import './index.css';

createRoot(document.getElementById('root')!).render(
  <StrictMode>

```

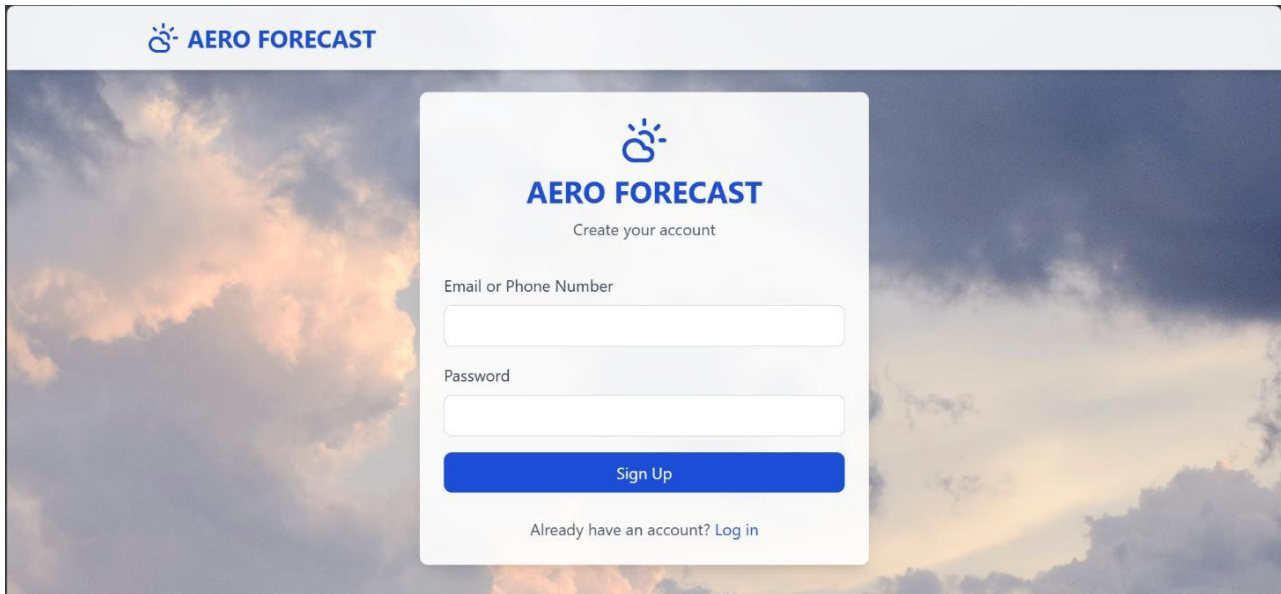
```
<App />
</StrictMode>
);

<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React + TS</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>
```

## 4.7 RESULTS

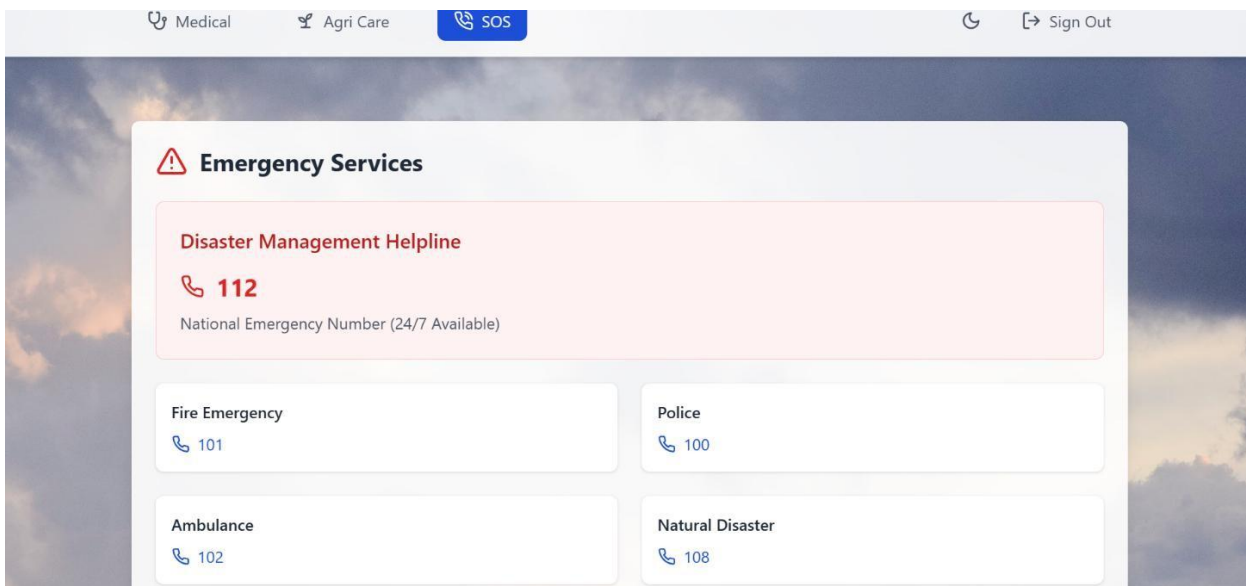
### 4.7.1 Weather Prediction Results

- **Outcome:** High accuracy in short-term and seasonal weather forecasts, assisting users in planning and decision-making.



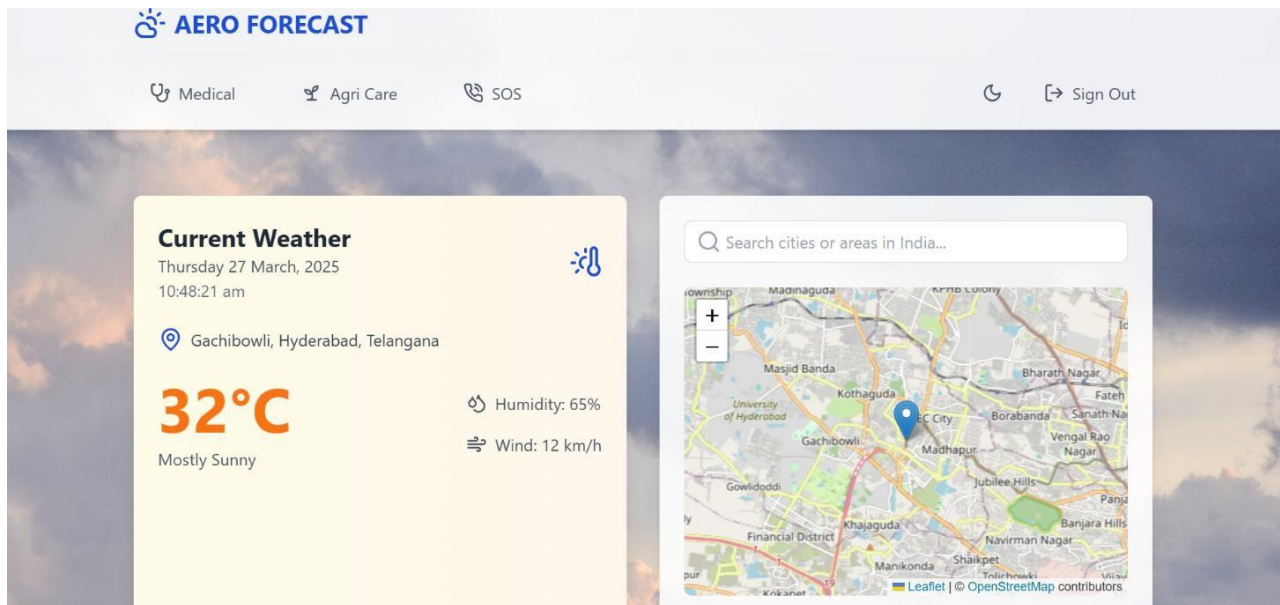
The screenshot shows the AERO FORECAST login page. At the top, there is a header with the AERO FORECAST logo and name. Below the header, there is a large background image of a cloudy sky. In the center, there is a white login box with the AERO FORECAST logo and the text "Create your account". Below this, there are two input fields: "Email or Phone Number" and "Password". A blue "Sign Up" button is located below the password field. At the bottom of the login box, there is a link that says "Already have an account? Log in".

### EMERGENCY SOS :

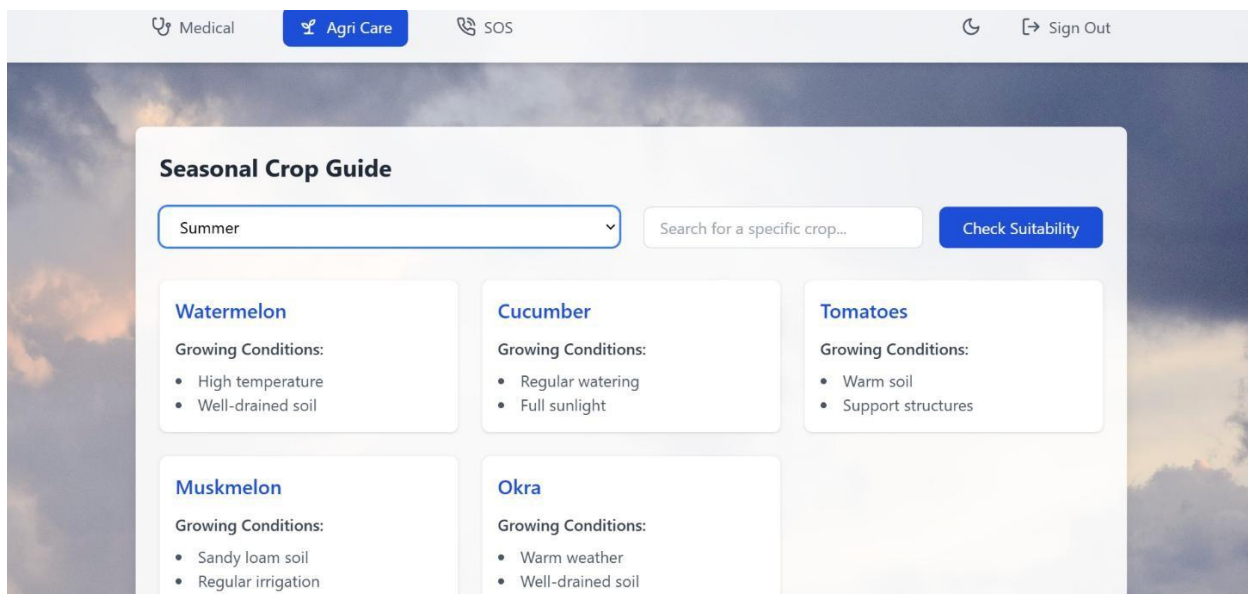


The screenshot shows the Emergency SOS page. At the top, there is a header with navigation links: "Medical", "Agri Care", and "SOS" (which is highlighted in blue). There is also a "Sign Out" link on the right. Below the header, there is a large background image of a cloudy sky. In the center, there is a white box with the title "Emergency Services" and a red warning icon. Below the title, there is a red box with the text "Disaster Management Helpline" and a large red "112" with a phone icon. Below this, there is a link that says "National Emergency Number (24/7 Available)". Below the red box, there are four white boxes arranged in a 2x2 grid. The top-left box is labeled "Fire Emergency" and has a phone icon and the number "101". The top-right box is labeled "Police" and has a phone icon and the number "100". The bottom-left box is labeled "Ambulance" and has a phone icon and the number "102". The bottom-right box is labeled "Natural Disaster" and has a phone icon and the number "108".

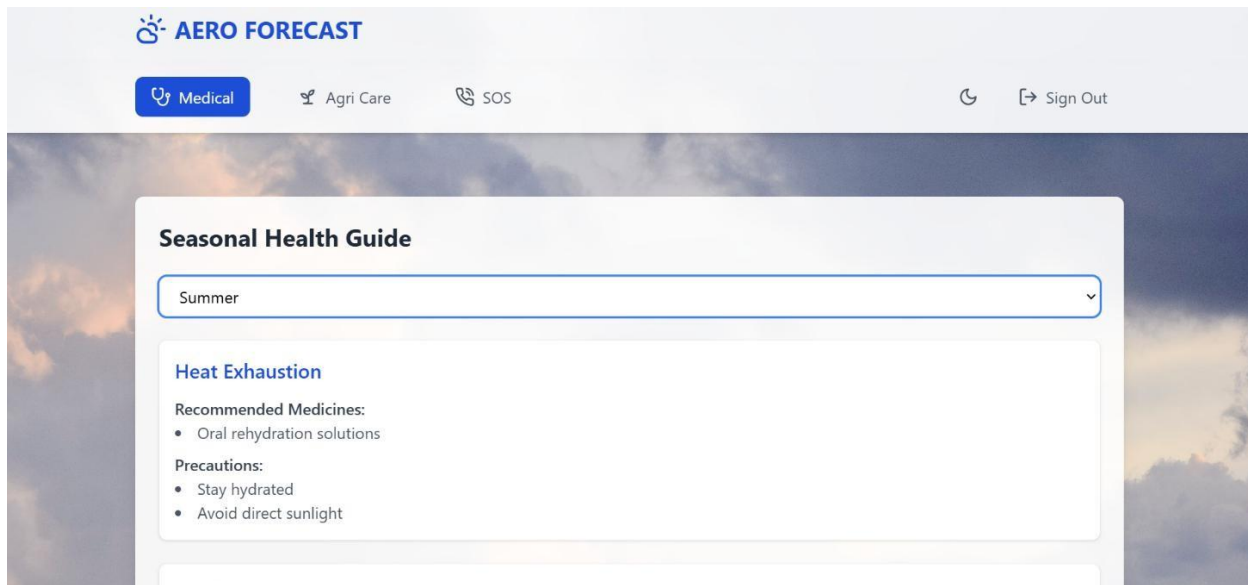
## HOME PAGE :



## AGRICULTURE SUPPORT PAGE :



## HEALTH CARE PAGE :



## **CHAPTER-5**

### **CONCLUSION**

#### **5.1 PROJECT CONCLUSION**

The development of Aero Forecast, a weather application designed to provide real-time weather updates, seasonal forecasts, agricultural insights, and emergency assistance, marks a significant step forward in public utility applications. This app is tailored to serve diverse user needs by leveraging weather data to offer insights for planning, safety, and health.

Aero Forecast not only provides up-to-date weather reports but also anticipates weather changes that impact various sectors, including agriculture, public health, and disaster management. For farmers, the application is a crucial tool that assists in crop planning, offering advice on the best times for planting and predicting potential pest or crop disease risks based on forecasted weather patterns.

For health-conscious users, the application identifies potential human health risks associated with specific seasonal conditions, allowing individuals to take preventive actions. Additionally, the app's built-in emergency alert system, which works even offline, provides life-saving capabilities during sudden disasters, offering peace of mind and direct support for users facing extreme weather events.

The culmination of this project represents an integration of modern technology with practical use cases that enhance public safety, health, and productivity. Aero Forecast is a prime example of how weather information, when made accessible and actionable, can positively impact everyday decision-making, emergency responsiveness, and proactive health measures.

#### **5.2 FUTURE SCOPE**

The Aero Forecast project has the potential to expand in multiple dimensions, offering even greater value to users and adapting to future technological advancements. Some prospective areas of expansion include:

### 5.2.1 Enhanced Predictive Capabilities

**Machine Learning Integration:** Future iterations of Aero Forecast could integrate machine learning algorithms to improve weather and crop prediction accuracy, enabling personalized recommendations based on historical weather patterns and specific crop needs.

**AI-Based Disease Predictions:** Leveraging AI for predictive health monitoring would allow the app to offer more personalized health alerts by analyzing environmental factors that contribute to disease prevalence in specific regions.

### 5.2.2 Expanded Agricultural Assistance

- **Regional Crop Recommendations:** By incorporating local agricultural data, Aero Forecast could provide farmers with hyper-localized insights, suggesting specific crop varieties suited to regional climates and market demands.
- **Soil and Water Management Insights:** Integrating data on soil types, water availability, and seasonal weather could help Aero Forecast offer advice on irrigation and fertilization practices, thus promoting sustainable farming.

### 5.2.3 Real-Time Health Alerts

- **Community Health Reporting:** In partnership with healthcare organizations, Atmos could issue real-time alerts on outbreaks or rising illness trends, allowing users to take timely precautions.
- **Allergy and Pollution Monitoring:** With the addition of air quality and pollen data, the app could provide alerts for individuals with respiratory conditions or allergies, improving public health monitoring.

### 5.2.4 Enhanced Emergency Features

- **Location-Based Rescue Services:** In addition to the SOS button, Aero Forecast could provide real-time data to local rescue teams and offer geolocation features for efficient user tracking during disasters.
- **Offline Map Navigation:** Adding an offline map feature would allow users to access critical navigation information during disasters, even without internet access.

### 5.2.5 Global Expansion and Localization

- **Support for Multiple Languages and Regions:** Expanding Aero Forecast to include regional languages and localized data would make it more accessible to diverse populations worldwide.
- **Cultural and Seasonal Customization:** Customizing the app for different climates and cultural needs would enhance its relevance to global users, making it a valuable resource for seasonal predictions and agriculture across various ecosystems.
- The future scope for Aero Forecast holds significant potential for growth and impact. By expanding its predictive and emergency-response capabilities, it can continue to evolve into a vital resource for individuals, farmers, health professionals, and rescue teams alike.

## REFERENCES

- [1] Weather API Documentation. “Weather API Reference.” Accessed October 30, 2024. Available at: <https://www.weatherapi.com/docs/>
- [2] Center for Disease Control and Prevention (CDC). “Seasonal Illness Trends and Prevention.” Accessed October 30, 2024. Available at: <https://www.cdc.gov/seasonal-trends>
- [3] PyTorch Foundation. “Developing Machine Learning Models for Weather Prediction.” Accessed October 30, 2024. Available at: <https://pytorch.org/>
- [4] OpenAI Research Team. “AI Models for Predictive Weather Forecasting and Health Risk Detection.” 2025.
- [5] Indian Meteorological Department (IMD). “Annual Weather Summary and Forecast Reports 2024–2025.” Government of India, 2025.
- [6] World Health Organization (WHO). “Climate Change and Seasonal Disease Impact 2024–2025.” WHO Climate Division Report, 2025.
- [7] Google Cloud AI. “Leveraging Cloud Machine Learning for Real-Time Forecasting.” Technical Whitepaper, 2024. Available at: <https://cloud.google.com/ai>
- [8] Microsoft Azure. “Intelligent Weather Data Analysis Using Azure Machine Learning.” Microsoft Documentation, 2024.
- [9] National Aeronautics and Space Administration (NASA). “Earth Observation Data for Weather and Climate Analytics.” NASA Earth Data Portal, 2024. Available at: <https://earthdata.nasa.gov/>