**COLLEGE CODE : 9623**

**COLLEGE NAME : AMRITA COLLEGE OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT : CSE**

**STUDENT NM-ID : CD35C39AB3E04C7BE0540250C279E9E9**

**ROLL NO : 8**

**DATE : 06/10/2025**

**COMPLETED THE PROJECT NAMED AS PHASE 4**

**TECHNOLOGY PROJECT NAME : LIVE WEATHER BROADCAST**

**SUBMITTED BY,**

**NAME:ADHARSH P**

**MOBILE NO : 9384758738**

# Live Weather Dashboard

## 1. Project Overview

A Live Weather Dashboard is a web-based or mobile application that provides real-time weather data, forecasts, and visual insights to users. It typically fetches data from APIs like OpenWeatherMap, WeatherAPI, or AccuWeather and presents it in an interactive interface.

## 2. Enhancements (Improvements to Existing Dashboard)

If you already have a basic dashboard (that shows temperature, humidity, etc.), the following enhancements can be made:

### ¯† A. UI/UX Improvements

Responsive design: Make the dashboard mobile-friendly using CSS frameworks like Bootstrap or Tailwind CSS.

Theming: Add dark/light mode toggle.

Animations: Use libraries like Framer Motion or React Spring for smooth transitions.

Dashboard cards: Use visually appealing cards for each city's weather data.

### ¯† B. Advanced Weather Data

Hourly forecast: Display next 24 hours forecast graphically.

7-day forecast: Show upcoming weather trends using line/bar charts.

Air quality index (AQI): Integrate AQI data with color-coded indicators.

Sunrise/sunset times, wind speed, UV index, visibility.

🔹 C. Data Visualization

Integrate charts using libraries like Chart.js or Recharts.

Use icons for weather conditions ( ☀️ 🌧️ ❄️ ) from libraries like Weather Icons.

🔹 D. Geolocation and Search

Auto-detect user location using browser geolocation API.

City search bar with autocomplete suggestions.

🔹 E. Alerts & Notifications

Severe weather alerts (storms, rain warnings).

Push notifications for significant changes.

🔹 F. Performance Enhancements

Use caching with localStorage or IndexedDB to reduce API calls.

Lazy loading for assets.

Use service workers for offline support (PWA).

🔹 G. Integration with Maps

Show weather overlay on Google Maps or Leaflet map.

Allow users to click on map points to view weather.

¯† H. Accessibility

Add ARIA labels, contrast, and keyboard navigation.

## 3. Technology Stack

| *Layer* | *Tools* |
| --- | --- |
| Front-End | React.js / Angular / Vue.js |
| Backend | Node.js / Express (optional if API consumed directly) |
| API Source | OpenWeatherMap / WeatherAPI / AccuWeather |
| Styling | Tailwind CSS / Bootstrap |
| Charts | Chart.js / Recharts |
| Deployment | Netlify / Vercel / AWS Amplify / GitHub Pages |
| Version Control | Git & GitHub |
| Optional DB | Firebase / MongoDB (if storing user preferences) |

## 4. Deployment Steps

¯† A. Build Preparation

1. Ensure all API keys are stored securely (.env files).
2. Run tests (npm run test) and build (npm run build).
3. Optimize assets and check Lighthouse performance.

¯† B. Choose Deployment Platform

Static Front-End:

Netlify / Vercel: Simple drag-and-drop or GitHub integration.

Steps:

Connect GitHub repo

Set build command (npm run build)

Deploy automatically on each commit

Full Stack (Backend + Frontend):

Use Render, Railway, or AWS EC2.

Use Docker for containerization (optional).

¯† C. Environment Variables

Configure API keys in environment settings (never commit .env file).

¯† D. Continuous Deployment (CI/CD)

Automate deployment using GitHub Actions:

```
name: Deploy to Netlify
on:
  push:
    branches:
      - main
jobs:
  build-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - run: npm install
```

```
- run: npm run build
- uses: netlify/actions@v1
  with:
    publish-dir: ./build
```

- E. Testing After Deployment

Check responsiveness, data loading, and API performance.

Monitor logs for errors.

**5. Testing**

Unit Testing: Using Jest (React) or Jasmine (Angular).

Integration Testing: Test API calls and data binding.

UI Testing: Cypress / Playwright.

Performance Testing: Lighthouse, GTmetrix.

**6. Documentation**

Include README.md with setup steps.

Document APIs used, features, and deployment process.