COLLEGE CODE : 9623

COLLEGE NAME : Amrita College of Engineering And Technology

DEPARTMENT : Computer Science and Engineering

STUDENT NM-ID :F0F3EBA66EADA4782A846E244BC964C3

ROLL NO : 23CS013

DATE : 12-09-2025

Completed the project named as

Phase 2 Solution Design & Architecture

PROJECT NAME : Live Weather Dashboard

SUBMITTED BY,

NAME :AJAY C
MOBILE NO :9597354838

⍰ Phase 2: Solution Design & Architecture

1. Tech Stack Selection

Frontend:

Framework: React.js (for reusable components, SPA behavior)

UI Styling: Tailwind CSS (fast, responsive UI)

Charting: Recharts or Chart.js (for weather trends visualization)

State Management: Redux Toolkit / React Query (to handle API calls + caching)

Backend:

Runtime: Node.js with Express.js (for REST API endpoints)

API Integration: OpenWeatherMap / WeatherAPI / AccuWeather (3rd party weather API)

Data Caching: Redis (optional, for storing frequent weather requests)

Database:

MongoDB (for user preferences like favorite cities, units, theme)

Deployment & Tools:

Hosting: Vercel (frontend), Render/Heroku (backend)

Version Control: GitHub / GitLab

CI/CD: GitHub Actions

Authentication (optional Phase 3): Firebase Auth / JWT

UI Structure

Main Screens / Components:

1. **Dashboard Page**

Search bar (city/location input)

Current weather widget (temperature, condition, icon)

Forecast cards (hourly & 7-day)

Charts (temperature, humidity, wind trends)

2. **Settings Page**

Units (Celsius / Fahrenheit)

Theme (Dark / Light)

Saved locations

3. **Error / Loading States**

Spinner / skeleton UI while fetching

Error banner if API fails

---

3. **API Schema Design**

Base URL (Backend): /api/v1/weather

Endpoints:

GET /current?city={cityName} → Fetch current weather

GET /forecast/daily?city={cityName}&days=7 → 7-day forecast

GET /forecast/hourly?city={cityName}&hours=24 → 24-hour forecast

POST /preferences → Save user preferences (units, favorite cities)

GET /preferences/:userId → Retrieve user preferences

Sample Response (Current Weather):

```
{
  "city": "Chennai",
  "temperature": 31,
  "unit": "C",
  "condition": "Sunny",
  "humidity": 65,
  "wind_speed": 12,
  "icon": "sunny.png"
}
```

---

4. **Data Handling Approach**

Frontend:

Fetch API data via Redux Toolkit Query or React Query.

Use local Storage for theme + last searched city.

Cache results for X minutes to avoid unnecessary calls.

Backend:

Fetch from weather API provider.

Apply rate-limiting to avoid overuse of API keys.

(Optional) Cache common city weather data in Redis for 5–10 mins.

---

5. **Component / Module**

Diagram Frontend Modules:

SearchBar → takes user input

WeatherCard → displays current weather

ForecastList → list of daily/hourly forecasts

WeatherChart → line/bar charts for trends

Settings → theme, units, favorites

Backend Modules:

weatherController.js → Handles API requests

weatherService.js → Calls external weather API

preferencesController.js → Manages user settings

db.js → MongoDB connection

---

6. **Basic Flow Diagram**

Flow (Live Weather Dashboard):

[User]
↓ (enters city name)
[Frontend: React]
↓ (calls API endpoint)
[Backend: Node/Express]
↓ (fetches from OpenWeather API)
↓ (formats response)
↓
[Frontend receives JSON]
↓
[Renders WeatherCard + Forecast + Charts]