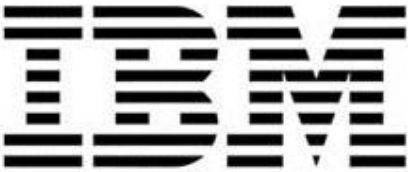

COLLEGE CODE : 9623

COLLEGE NAME : Amrita College of Engineering And Technology

DEPARTMENT : Computer Science and Engineering

STUDENT NM-ID : 62463B9F22FE1538AD5428F1EABBE329

ROLL NO : 23CS016

DATE : 25-09-2025

Completed the project named as

Phase 3 MVP Implementation

PROJECT NAME : LIVE WEATHER BROADCAST

SUBMITTED BY,

NAME : Ananthan N P
MOBILE NO :7708742601

Phase 3 - MVP Implementation

---

1. Project Setup

- Define project scope: Build a lightweight weather broadcast MVP that fetches live weather updates and displays them to users.
- Tools and technologies:
  * Frontend: HTML, CSS, JavaScript (or React if scaling).
  * Backend: None required for MVP (can use API directly).
  * Weather API: OpenWeatherMap / WeatherAPI.
  * Hosting: GitHub Pages / Firebase Hosting.
- Initialize project repository on GitHub.
- Install required dependencies if using frameworks.

2. Core Features Implementation

- Fetch live weather data from an external API using API key.
- Display temperature, humidity, wind speed, and condition (sunny, cloudy, rainy, etc.).
- Implement location-based weather (via user input or geolocation).
- Refresh weather data periodically (e.g., every 5–10 minutes).
- Basic UI with responsive design for desktop and mobile.

3. Data Storage (Local State / Database)

- Local State: Store current weather data temporarily in JavaScript variables or state management (React useState).
- Optional: Use LocalStorage to cache last weather data to display while new data loads. - Database (Future enhancement): Firebase Realtime Database or Firestore for saving user preferences (e.g., favorite cities).

4. Testing Core Features

- Unit Testing: Verify API response parsing and UI rendering.
- Integration Testing: Ensure weather data updates correctly on the interface.
- Edge Cases: Handle invalid API key, no internet connection, or unavailable city names.
- Manual Testing: Test on different devices (mobile/desktop).

5. Version Control (GitHub)

- Initialize Git repository and push code to GitHub.
- Use branches for new features (e.g., `feature/api-integration`).
- Commit messages should follow a convention (e.g., "feat: add API integration").
- Use GitHub Issues for bug tracking and project board for task management.
- Enable GitHub Actions for automated testing and deployment.