COLLEGE CODE : 9509

COLLEGE NAME:HOLYCROSS ENGINEERING COLLEGE

DEPARTMENT:CSE

STUDENT NM-ID:A21ED64081D8052AD7D4921642CB1DB7

Roll No:950923104015

Date: 22.09.2025

Completed the project named as Phase 3

TECHNOLOGY PROJECT NAME:IBM-FE-Live Weather Dashboard

Submitted by,

Name:Isool Rabiya N

Mobile No:6369428920

1. Project Setup

Tech Stack Selection:

Frontend: React.js (for UI) or simple HTML/CSS/JS for MVP.

Backend (optional for MVP): Node.js/Express or direct API calls.

API: OpenWeatherMap / WeatherAPI for live weather data.

Environment Setup:

Install necessary dependencies (React, Tailwind/Bootstrap, Axios/Fetch API).

Create project repository on GitHub.

Configure .gitignore and basic project structure (components, services, assets).

Basic UI Wireframe Setup:

Header with project title.

Search bar for city/location.

Display container for weather data.

2. Core Features Implementation

Weather Search:

Implement city-based search to fetch current weather using API.

Weather Data Display:

Show temperature, humidity, wind speed, and weather condition (e.g., sunny, cloudy).

Add icons for different weather conditions.

Location Detection (optional MVP add-on):

Use browser geolocation to fetch weather of current location.

Responsive Design:

Ensure dashboard works on mobile and desktop.

3. Data Storage (Local State / Database)

Local State (MVP focus):

Use React useState / useEffect to manage fetched weather data.

Cache recent search results in local state for quick re-access.

Database (Future scope, not required for MVP):

Store user preferences or recent searches in a lightweight DB (SQLite/Firebase/MongoDB).

4. Testing Core Features

Unit Testing:

Test API call function (mock weather API).

Test search input validation (empty search, invalid city name).

UI Testing:

Ensure weather data updates correctly when a new city is searched.

Cross-Browser/Device Testing:

Verify responsive layout on desktop and mobile browsers.

5. Version Control (GitHub)

Repository Setup:

Create GitHub repo with main branch.

Push initial boilerplate code.

Branching Strategy:

Use feature branches (feature/search-bar, feature/weather-card).

Merge to main after testing.

Commit Practices:

Write meaningful commit messages (Added weather API integration, Fixed search validation).

L

Use GitHub issues for task tracking.Pull requests for peer review before merging.