COLLEGE CODE : 9238

COLLEGE NAME: Mangayarkarasi College of Engineering

DEPARTMENT: CSE

STUDENT NM-ID:
5040D5ACBEEB3EE144AD7AACD13DB424D

ROLL NO.: 923823104023

DATE: 08-09-2025

Completed the project named as Phase 2 - Solution Design & Architecture

FRONT END TECHNOLOGY

PROJECT NAME: LIVE WEATHER DASHBOARD

SUBMITTED BY,

NAME: KARTHIKA DEVI S

MOBILE NO: 7904591497

LIVE WEATHER DASHBOARD

1. Tech Stack Selection

To build a scalable and responsive live weather dashboard, the following technologies are selected:

Frontend

React.js for component-based UI

Tailwind CSS or Bootstrap for styling

Axios for API calls

Backend

Node.js with Express for RESTful API

Python with FastAPI (alternative for data-heavy processing)

API Integration

OpenWeatherMap or WeatherAPI for real-time weather data

GeoLocation API for user-based location detection

Database

MongoDB for storing user preferences and logs

Redis for caching frequent weather queries

Hosting & Deployment

Vercel or Netlify for frontend

Heroku or AWS EC2 for backend services

DevOps & Monitoring

GitHub Actions for CI/CD

Docker for containerization

Prometheus + Grafana for performance monitoring

2. UI Structure & API Schema Design

UI Structure

The dashboard is divided into intuitive sections:

Header: Contains app name, search bar, and location toggle

Main Panel: Displays current weather, temperature, humidity, wind speed

Forecast Section: Hourly and 7-day forecast with icons

Sidebar Widgets: Favorite locations, alerts, and settings

Footer: Credits, API source, and contact info

API Schema Design

Endpoint: `GET /weather?location={city}`

Some Endpoints:

```
GET /forecast?location={city}

POST /favorites

GET /alerts?location={city}
```

3. Data Handling Approach

Efficient data handling ensures performance and reliability:

Fetching Strategy

Real-time data fetched using scheduled polling

Webhooks for alert updates (if supported by API provider)

Caching

Redis used to cache frequent queries

TTL (Time to Live) set based on forecast freshness

Storage

MongoDB stores user preferences, search history, and logs

Weather logs used for analytics and trends

Error Handling

Retry mechanism for failed API calls

Fallback to cached data during API downtime

Graceful UI degradation with user notifications

Security Measures

API key encryption and rotation

HTTPS for secure data transmission

Input validation and rate limiting

4. Component / Module Diagram

Frontend Components

`SearchBar`: Input for city/location

`WeatherDisplay`: Shows current weather

`ForecastPanel`: Hourly and daily forecast

`MapWidget`: Optional weather map integration

`SettingsPanel`: Theme, units, and preferences

Backend Modules

`WeatherController`: Handles API requests

`ForecastService`: Processes forecast data

`APIClient`: Communicates with external weather APIs

`CacheManager`: Manages Redis caching

`UserPreferencesService`: Stores and retrieves user settings

Database Collections

`Users`: Stores user profiles and preferences

`Locations`: Saved locations and search history

`WeatherLogs`: Historical weather data

5. Basic Flow Diagram

The flow of data and interaction is as follows:

Code

```
User → UI → Backend →
Weather API
      ↓       ↓
   Display ← Cache ← DB
```

Explanation

User enters a location in the UI

Request is sent to backend

Backend checks Redis cache

If not found, fetches from external API

Response is cached and stored in DB

UI updates with latest data