**Jeffrey Project Proposal**

# **Statement of Purpose**

**Title:** YourWeatherNow App  
**Purpose:** Build a simple C# application that collects real-time weather data, shows the user’s current location, and displays the local time at that location. The app demonstrates coding knowledge while being practically useful for everyday weather checks.

# **Runtime Environment**

**Option 1: Cloud (Primary)**

* Designed to run as a **cloud-first application** on AWS.
* Backend deployed as a **.NET 8 Lambda function** behind **API Gateway**, retrieving weather data from the **OpenWeatherMap API (free tier)**.
* **Secrets Manager (amplify)** stores the API key securely, ensuring no sensitive data is hard-coded.
* Scales automatically and is accessible from both desktop and mobile clients through a web API endpoint.

**Option 2: Local (Secondary)**

* A lightweight **.NET console app** run locally for development, testing, or offline demonstrations.
* API key stored as an **environment variable**.
* Requires **.NET 6+ installed**, same core logic used in the cloud deployment.

# **Information Needed**

**Residence (for demo purposes this is “location”):**

* **Address/Location:** Enter a city name (e.g., Atlanta, London) or latitude/longitude coordinates.

# **Data to be Persisted (Database)**

In the production version, the app could persist:

* User’s last searched city or coordinates.
* User’s preferred units (°F or °C).
* Cached recent weather results to avoid repeat API calls.
* Error logs or diagnostic information.

For this demo project, **no database persistence** is required; values are retrieved live from the API.

# **App Concerns**

* **Access to data in a major emergency:** If connectivity is lost, cached results could be used. Future versions could tie into local/offline data sources.
* **Access to data by emergency personnel:** Not applicable for this weather demo, but the framework could be extended to provide community hazard/weather info to first responders.

# User Interface Outline and Functional Flow

## A) Cloud Web App (Primary)

**Screens**

1. **Home / Search**
   * Search box (city/ZIP) + “Use My Location” (GPS).
   * Unit toggle (°F / °C).
   * “Get Weather” button.
2. **Results**
   * Header: City, Country, Last Updated (local time at location).
   * Cards:
     + **Current Conditions:** description, temp, feels-like, humidity.
     + **Location:** coordinates (lat/lon), time zone/local time.
     + **Times:** sunrise, sunset.
   * Action row: Refresh | Change Location | Save as Favorite (optional).
3. **Settings**
   * Default units, recent searches, enable/disable geolocation.

**Flow**

1. User opens web app (**Progressive Web App (PWA)**) → Home.
2. User enters city/ZIP **or** taps **Use My Location** (consent prompt).
3. App calls **API Gateway → Lambda (.NET)** → **OpenWeatherMap** (API key in Secrets Manager).
4. API returns normalized JSON → UI renders **Current**, **Location**, **Sun & Sky**.
5. User can **toggle units**, **refresh**, or **search again**.
6. (Optional) Cache last result for quick reload and offline fallback.

## B) Console App (Secondary / Local)

**Outline**

1. Launch App → print banner: === YourWeatherNow ===
2. Prompt: enter **city** (or leave blank to enter **latitude/longitude**).
3. Build request → call **OpenWeatherMap** (units per default setting).
4. Display formatted output:
   * Location & country
   * Coordinates
   * Local time at that location
   * Weather description
   * Temperature (actual & feels like)
   * Humidity
   * Sunrise & Sunset
   * Timestamp of data

**Flow**  
User input → API call → Print formatted weather info → Exit (or prompt to query again).

# **Special Features**

* **GPS tie-in:** Future for mobile version can auto-detect location (via GPS or IP).
* **Emergency communications:** Could integrate with notifications or satellite services in future versions.
* **Customization:** Toggle between °F/°C, add forecast, and support offline caching.