Sprint 2: Initial Setup

1. System Overview

The WeatherSensor AI aims to provide users with real-time weather information enhanced by natural-language summaries generated through a large language model.

2. Functional Requirements

User Story: As a user, I want the application to retrieve real-time weather data, handle errors smoothly, and generate a clear, natural-language summary so I can quickly understand current weather conditions without reading raw data.

2.1 - Format/Structure Weather API Information –

- FR 2.1.1: Parse API response to pull out key weather values (temperature, wind, humidity, etc.).
- FR 2.1.2: Implement key weather values into the website.
- FR 2.1.3: Store extracted data in a clean, labeled JSON format.

Progress Summary:

2.2 - Implement Error Handling for Weather API –

- FR 2.2.1: Detect and report API request errors (timeouts, bad responses).
- FR 2.2.2: Print clear messages for missing or invalid data.
- FR 2.2.3: Add simple retry logic if a request fails once.

Progress Summary:

2.3 - Test Cases / Verify Correctness of API Responses –

- FR 2.3.1: Create test cases to check that valid data is parsed correctly.
- FR 2.3.2: Test with both normal and error responses.
- FR 2.3.3: Confirm output matches expected weather values.
- FR 2.3.4: Collect group feedback and adjust the prompt or parameters as needed.

Progress Summary:

2.4 - Choose LLM API and Configure LLM Provider – Manu Redd

- FR 2.4.1: Research available LLM APIs and select one (e.g., Gemini).
- FR 2.4.2: Set up environment variables for API key and model name.
- FR 2.4.3: Verify connection with a simple test prompt.

Progress Summary: I researched the most optimal gemini models for our use-case and learned how to use an API key for sending requests for text generation. I used sample code from google and the API requests are working.

2.5 - Design Prompt Structure for Weather Summary –

- FR 2.5.1: Write a basic prompt format that includes weather data fields.
- FR 2.5.2: Keep the prompt concise and easy to read.
- FR 2.5.3: Add examples for different weather types (clear, rain, storm).

Progress Summary:

2.6 - Setup of Google Gemini API Pipeline –

- FR 2.6.1: Install and configure Google Gemini API client.
- FR 2.6.2: Build a function to send prompts and receive responses.
- FR 2.6.3: Log the results and confirm that output text is returned.

Progress Summary:

2.7 - Implement Summary within Gemini, Return Generated Text –

- FR 2.7.1: Combine extracted weather data with the summary prompt.
- FR 2.7.2: Generate a text summary through Gemini and display it.
- FR 2.7.3: Test several examples to confirm correct output.

Progress Summary:

2.8 - Verify Summary Quality with Test Cases –

- FR 2.8.1: Create sample weather inputs (clear, rain, storm, extreme heat) to test summaries.

- FR 2.8.2: Check that each summary accurately reflects key weather values (temperature, wind, precipitation).
- FR 2.8.3: Review summaries for clarity, grammar, and tone consistency.
- FR 2.8.4: Compare generated text to a short “ideal” reference summary for quality scoring.
- FR 2.8.5: Collect group feedback and adjust the prompt or parameters as needed.

Progress Summary: