Sprint 4: Backend Integration and Final Polish

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.

4.1 - Create and Implement Backend Chat Handler – Manu Redd

- FR 4.1.1: Build a backend endpoint to receive chat messages and forward them to Gemini.
- FR 4.1.2: Log all chat exchanges for debugging and testing.

Progress Summary:

4.2 - Handle Real-time Message Exchange – Manu Redd

- FR 4.2.1: Return Gemini responses to the front-end chat in real time.

Progress Summary:

4.3 - Add Conversational Limits and Constraints – Manu Redd

- FR 4.3.1: Restrict chat replies to weather-related questions only.
- FR 4.3.2: Add checks to ignore unrelated or unsafe inputs.
- FR 4.3.3: Display a short “out-of-scope” message when a user asks a non-weather topic.

Progress Summary:

4.4 - Implement Error Handling for Chat – Jackson Yanek

- FR 4.4.1: Detect failed Gemini API calls and show a user-friendly error message.
- FR 4.4.2: Retry failed requests once before stopping.

- FR 4.4.3: Log connection or response errors for review.

Progress Summary:

4.5 - Verify Topic and Conversational Flow with Test Cases – Cole Cooper

- FR 4.5.1: Create test conversations for valid and invalid weather topics.
- FR 4.5.2: Verify Gemini replies stay within the weather domain.
- FR 4.5.3: Check that error messages appear correctly when issues occur.

Progress Summary:

4.6 - Polish UI (Ensure Website is Professional) – Evans Chigweshe

- FR 4.6.1: Improve chat and weather panel layout for readability.
- FR 4.6.2: Replace placeholder text and ensure consistent design tone.
- FR 4.6.3: Ensure spelling and grammar correctness.

Progress Summary:

4.7 - Review Color Palette and Spacing for Consistency – Cole Cooper

- FR 4.7.1: Create a spacing scale (e.g., 4/8/12/16 px) and apply across pages.
- FR 4.7.2: Ensure contrast ratios meet WCAG AA for text/icons.

Progress Summary:

4.8 - Final Responsiveness – Jackson Yanek

- FR 4.8.1: Test responsiveness on desktop, tablet, and mobile.
- FR 4.8.2: Optimize loading animations and transitions.
- FR 4.8.3: Ensure all links, buttons, and routes work before final deployment.

Progress Summary:

4.9 - Test Load Times and Transitions – Cole Cooper

- FR 4.9.1: Measure and record initial paint and weather fetch latency.
- FR 4.9.2: Verify loading indicators appear during any delay > 300 ms.

Progress Summary:

4.10 - Update README and Documentation – Cole Cooper

- FR 4.10.1: Update README with setup, usage, and deployment instructions.
- FR 4.10.2: Add a section summarizing all sprints and key features.
- FR 4.10.3: Include contributor credits and project structure overview.

Progress Summary:

4.11 - Create Demo Video showing Final Product – Everyone

- FR 4.11.1: Record a ≤ 3-minute walkthrough: search, display, chat summary.
- FR 4.11.2: Include one error/retry and one unit toggle demonstration.

Progress Summary:

4.12 - Write Short Team Retrospective and Project Reflection – Cole Cooper

- FR 4.11.1: Present key accomplishments and challenges faced during development.
- FR 4.11.2: Compile lessons learned and improvement ideas for future versions.

Progress Summary:

4.13 - Create and Complete Sprint 4 Artifacts - Riley England

- FR 4.13.1: Compile Requirements for Sprint 4 into formatted documentation.
- FR 4.13.2: Write functional requirements for each requirement, leaving space for progress summaries.
- FR 4.13.3: Verify artifact alignment with course rubric and project objectives.
- FR 4.13.4: Upload all Sprint 4 documents to github.

Progress Summary: Compiled all Sprint 4 Requirements into the specified format of the Artifacts document. Broke each requirement down into its Functional Requirements, and gave a description for each. Ensured consistent formatting and completeness across all team members' sections before submission. Finally, compiled all Sprint 4 deliverables and uploaded them to the github, ensuring the final version was pushed.

4.14 - Add Prologue Comments to All Code Files - Cole Cooper

- FR 4.14.1: Read and Understand each program file in the Github.

- FR 4.14.2: Write detailed prologue comments for each file.
- FR 4.14.3: Upload all adjusted files to Github.

Progress Summary: