Team-8: WeatherSensor AI

Team Members

Last name	First name
Cooper	Cole
England	Riley
Chigweshe	Evans
Redd	Manu
Yanek	Jackson

Project Name

WeatherSensor AI

Project Synopsis

A responsive web application integrating real-time weather data and Google Gemini AI for intelligent weather summaries, interactive chat, and dynamic user-location forecasting.

Architecture Description

System overview

The WeatherSensor AI is a client-side web application that delivers real-time weather data combined with conversational AI capabilities. It's built for static hosting (GitHub Pages) and emphasizes design simplicity, data clarity, and intelligent user engagement through Google Gemini integration**.**

The architecture is organized into three main layers:

Presentation layer (Frontend/UI)

The frontend is built using HTML5, CSS, and JavaScript, providing a robust foundation for managing all user interactions, including location searches, weather data display, and chatting with Gemini. It features a fully responsive design that adapts seamlessly to mobile, tablet, and desktop screens, while dynamically handling error messages and loading states to ensure a smooth user experience.

Integration layer (API interaction)

The integration layer acts as a bridge between the application and external APIs, keeping the frontend separate from the complexities of third-party services. It handles fetching weather data, including temperature, precipitation, wind speed, UV index, and air quality, through RESTful API calls, as well as obtaining summaries and chat responses from the Gemini API. Additionally, this layer manages request limits and gracefully handles network errors, ensuring that any issues are logged and presented to the user clearly and understandably.

Logic Layer

This layer takes the raw data returned by the APIs and prepares it for display in a user-friendly format. It handles tasks like converting timestamps and measurement units (such as °C/°F or km/h/mph) and translating air quality index values into readable formats. It also generates clear, natural-language summaries of the weather, such as “Light rain expected this afternoon,”.

Data flow

When a user opens the site:

1. The UI requests geolocation (or user input for city name).
2. The Weather API module fetches real-time data for the chosen location.
3. Parsed data is displayed on the main dashboard.
4. Simultaneously, the weather summary is sent to the Gemini API, which returns a conversational summary (e.g., “It’s a warm, breezy afternoon with clear skies.”).
5. The Chat UI allows ongoing dialogue with Gemini. Users can ask for extended forecasts, clothing advice, or activity recommendations.
6. All modules communicate through lightweight asynchronous events within the browser.

Component architecture

Each component is loosely coupled, allowing independent updates:

Component	Description
idex.html	Entry point for the application
styles.css	Layout and theming (colors, spacing, responsive grid)
weather.js	Handles API calls and data parsing for weather information
gemini.js	Handles AI API requests and responses.
chat.js	Manages chat interface, message rendering, and Gemini responses.
main.js	Initialization, event listeners, and app control flow.

UML Diagrams

USE Case diagram

- User interaction overview: Shows how the user interacts with the three main system capabilities: viewing weather, searching for a location, and chatting with the AI

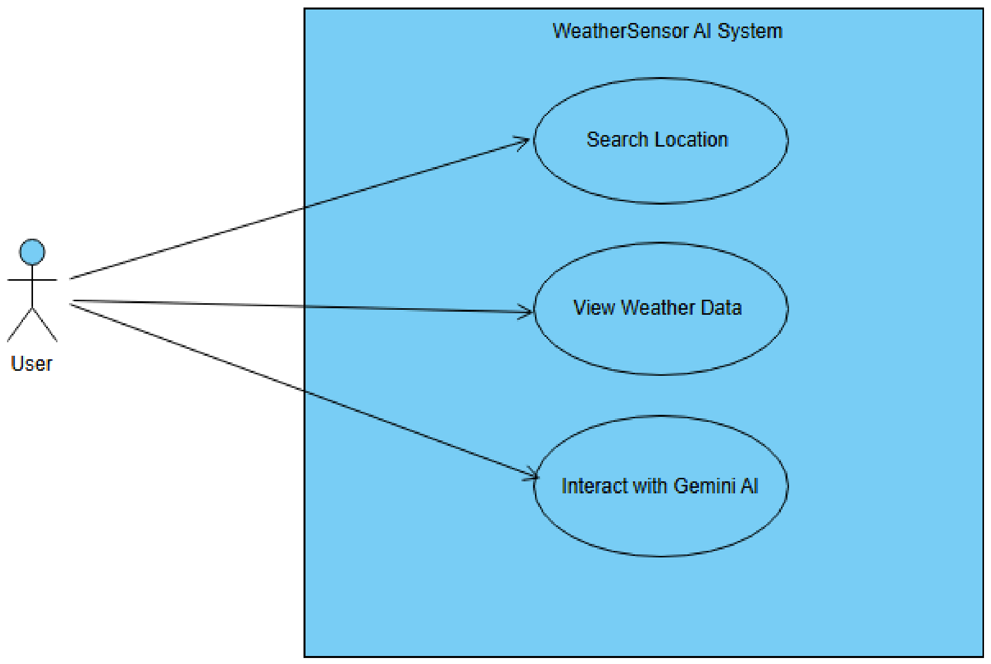

figure1.0

Component diagram

- Shows how major system components communicate and depend on one another.

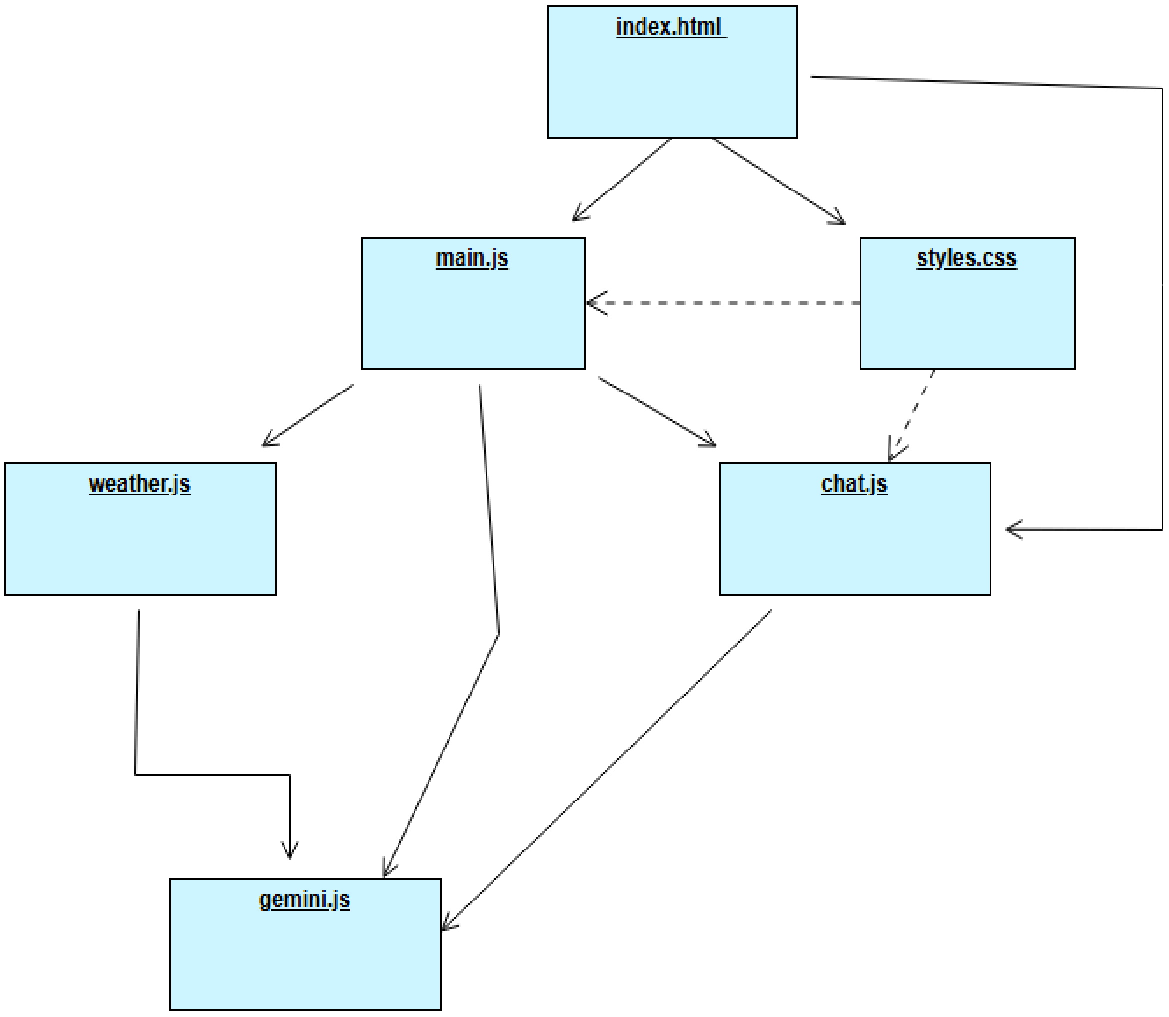

figure 1.2

Data Flow Diagram

- Shows how data flows through the system, from user input to weather data retrieval, Gemini processing, and display.

Figure

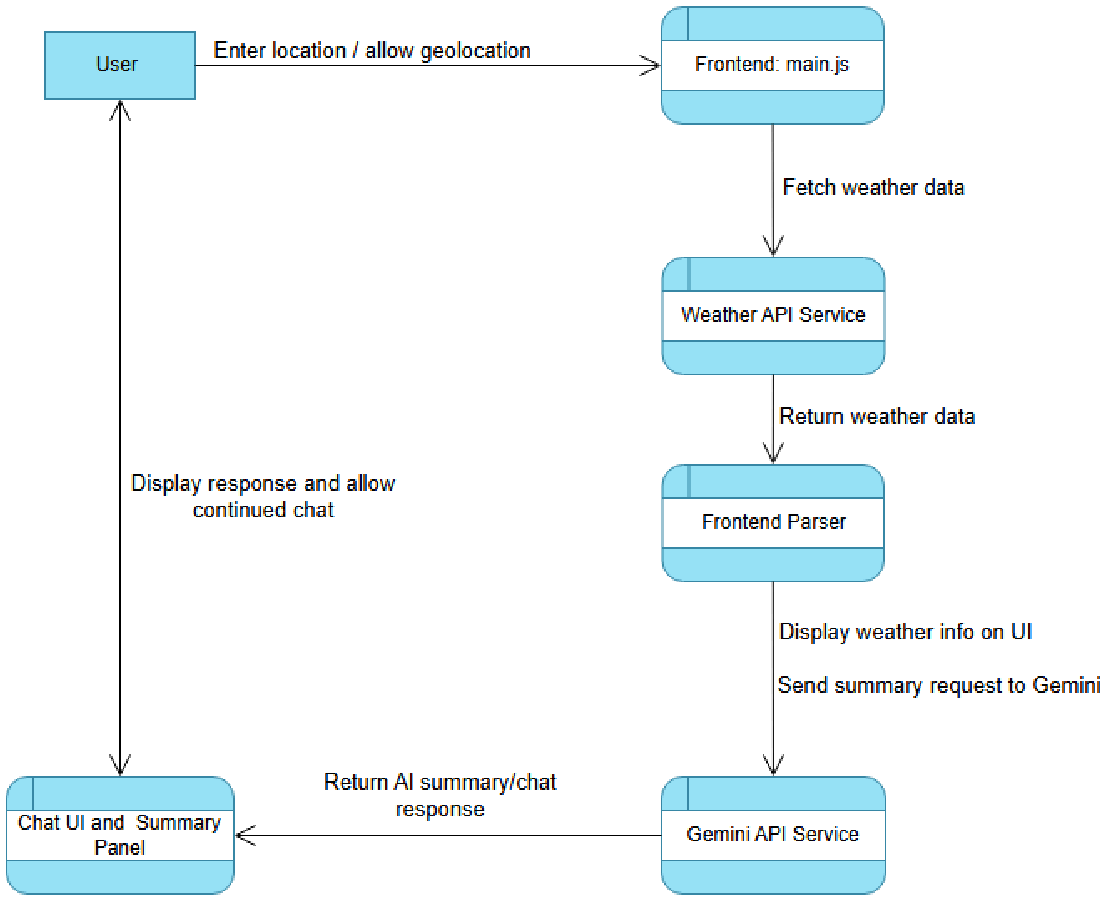
User
Enter location / allow geolocation
Frontend: main.js
Fetch weather data
Weather API Service
Return weather data
Frontend Parser
Display weather info on UI
Send summary request to Gemini
Gemini API Service
Return AI summary/chat response
Chat UI and Summary Panel
Display response and allow continued chat

figure 1.3