

Max Wang
Sebastian Martinez
Brody Wingrove

Demo

<https://www.youtube.com/watch?v=2ann3AnD0-s>

We-ather

Our project, We-Ather, started with a frustration we all shared: weather is almost always wrong. We constantly checked forecasts, only for them to fail us in the moments that mattered. Whether it was unexpected rain, inaccurate wind, or a “warm day” that felt freezing. Traditional forecasting works on broad models, but it often can’t capture what people actually experience on the ground, in real time.

That led us to a question: What if weather wasn’t just predicted from the top down, but reported from the bottom up? What if the people experiencing the weather could help create the forecast?

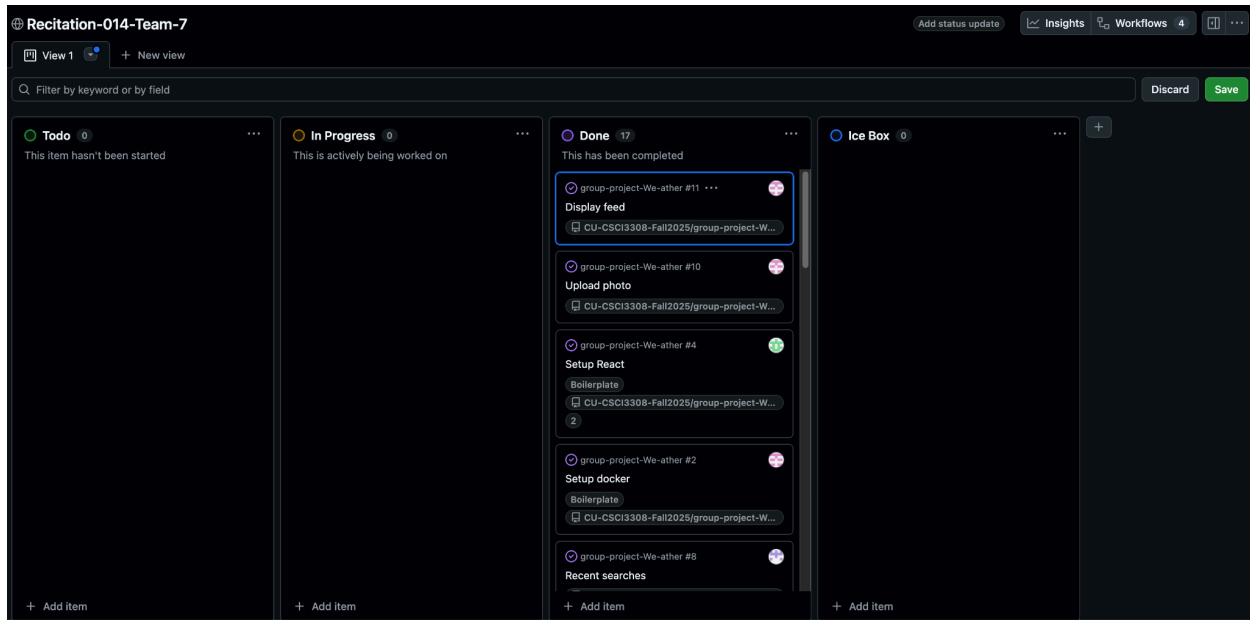
This idea became We-Eather, a crowdsourced weather platform that gives users the ability to submit real-time observations like temperature, rain, wind, and even subjective descriptions such as “humid,” “icy,” or “feels like summer.” These reports form a constantly updated map powered by the people actually outside.

We-Eather reflects a belief that weather is personal, local, and collaborative. Crowdsourcing can improve accuracy, lived experience matters, and community data can reveal patterns models miss.

We built We-Ather because we’re tired of inaccurate forecasts, and because we believe the best weather information comes from the people living in it.

Project Tracker

<https://github.com/orgs/CU-CSCI3308-Fall2025/projects/20>



Github Repo

<https://github.com/CU-CSCI3308-Fall2025/group-project-We-ather>

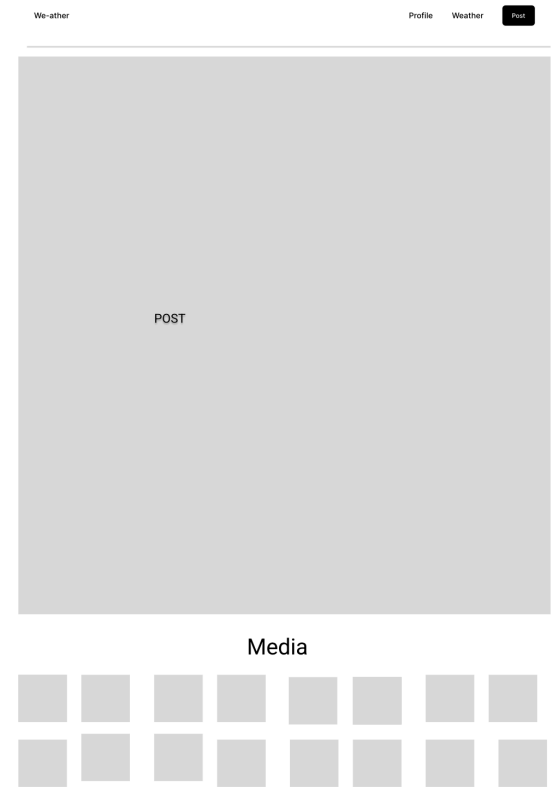
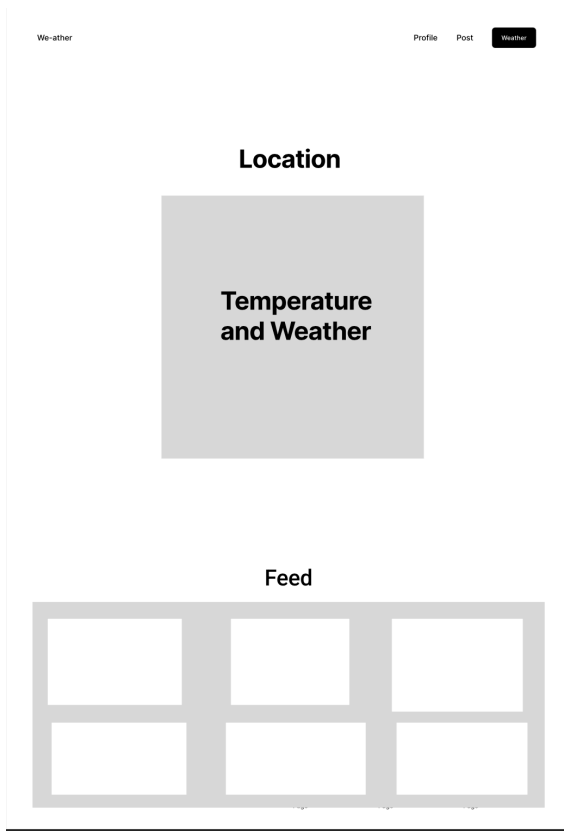
Contributions

Sebastian - I worked mostly on the back end working on the uploading posts and images features, autocomplete feature, current location feature, delete post feature. I also did the front end for the posts feed and profile display.

Max - Worked mostly on the back end, working on the initial setup, login and registration, and the integration with the weather API. Also worked on the front end for the weather display.

Brody Wingrove - I worked mostly on the front end, using HTML, Handlebars, Bootstrap, and JavaScript. The main features I worked on were the location search bar, which included using an API as a middleman between a location search and the weather API to autocomplete the search, and the profile page.

Wireframes



We-ather

Post

Weather

Profile



Full Name

Country, username

Bio

Body text

Follow

member since ...

Related products



Date

weather

analytics and interactions



Date

weather

analytics and interactions



Date

weather

analytics and interactions



Date

weather

analytics and interactions



D

w

a & i



D

w

a & i

Site name



Topic

Page

Page

Page

Topic

Page

Page

Page

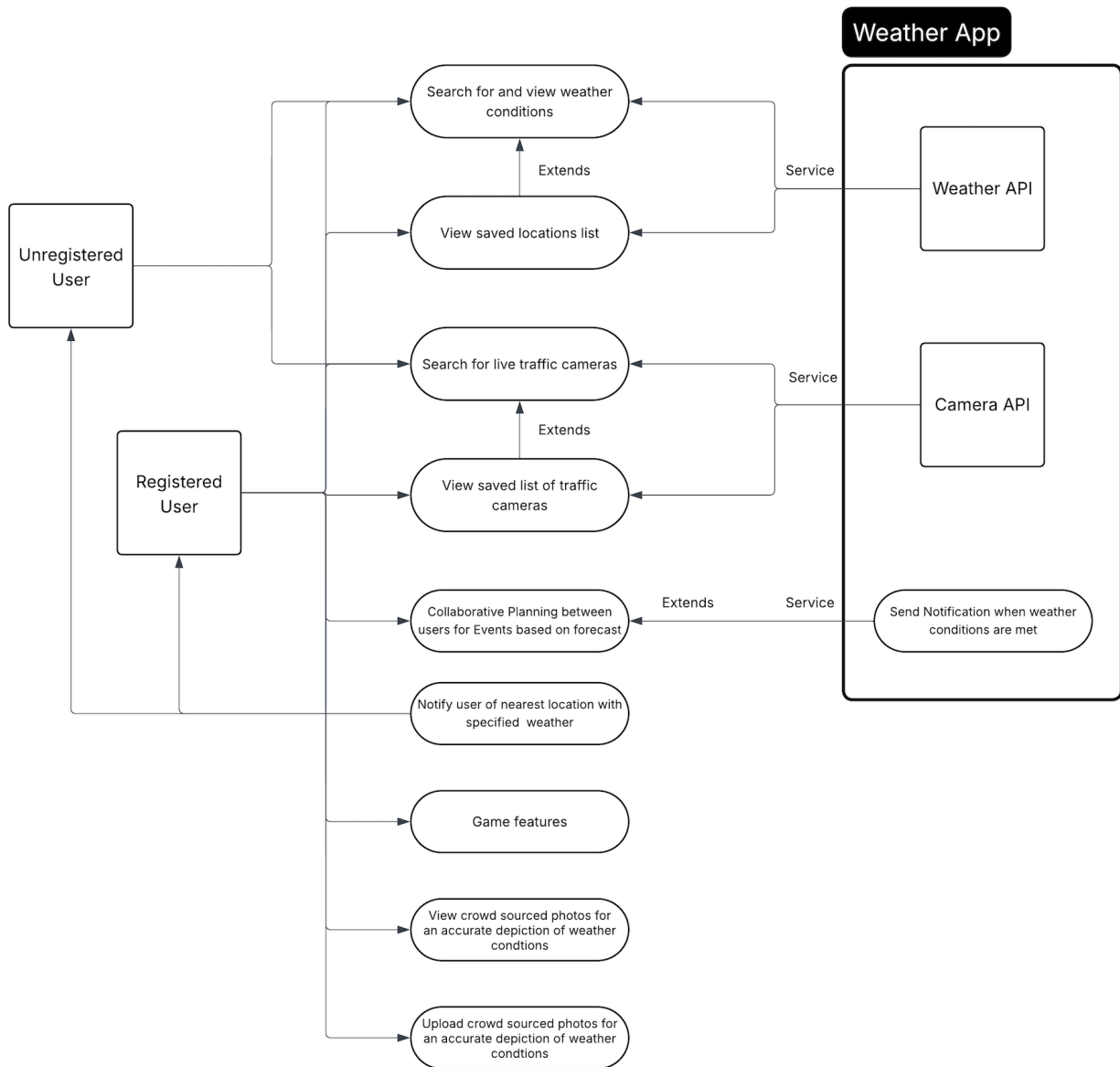
Topic

Page

Page

Page

Use Case Diagram



Deployment link

<https://group-project-we-ather.onrender.com/>

User Acceptance Testing (UAT) plan

Project: We-ather
Team Members: Sebastian, Brody, Max, Daniel
Date: Nov. 6 2025

Feature 1: User Login

Objective:
Ensure that registered users can log in successfully and that incorrect credentials are handled properly.

Test Case 1: Valid Login

- **Test Data:**
 - Username: testuser
 - Password: password123
- **Steps:**
 1. Navigate to the login page (/login).
 2. Enter the correct username and password.
 3. Click "Login."
- **Expected Result:**
User is redirected to the Home page and greeted by their username.
- **Actual Result:**
Works as expected.
- **Test Environment:**
Localhost, Node.js/Express backend, PostgreSQL database, Chrome browser.
- **Tester:** Sebastian Martinez

Test Case 2: Invalid Login

- **Test Data:**
 - Username: wronguser
 - Password: wrongpassword
- **Steps:**

- **Expected Result:**
Displays message: "Invalid username or password." User is not redirected.
- **Actual Result:**
Error message shown properly.
- **Test Environment:**
Localhost, Node.js/Express backend, PostgreSQL database, Chrome browser.
- **Tester:**

- **Expected Result:**
Displays message: "Username already exists." Registration fails.
- **Actual Result:**
Duplicate registration correctly rejected.
- **Test Environment:**
Localhost, Node.js/Express backend, PostgreSQL database, Chrome browser.
- **Tester:**

Feature 3: Weather Search and Autocomplete

Objective:
Ensure that users can search for weather data by city name, see autocomplete suggestions, and get accurate weather information.

Test Case 1: Valid City Search

- **Test Data:**
 - City: Boulder, CO
- **Steps:**
 1. Go to /home.
 2. Type "Boulder" into the search bar.
 3. Select "Boulder, CO" from autocomplete dropdown.
 4. Click "Search."
- **Expected Result:**
Displays current weather information for Boulder (temperature, humidity, conditions).
- **Actual Result:**
Weather data loads and displays correctly.
- **Test Environment:**
Localhost, Node.js/Express backend, OpenWeather API, Chrome browser.
- **Tester:** Sebastian Martinez

Test Case 2: Invalid City Search

- **Test Data:**
 - City: asdfsdfjkl (nonexistent)
- **Steps:**
 1. Go to /home.
 2. Type a random invalid city name.
 3. Click "Search."
- **Expected Result:**
Displays message: "City not found" or "Invalid input."
- **Actual Result:**
Error message shown properly, no crash.
- **Test Environment:**
Localhost, Node.js/Express backend, OpenWeather API, Chrome browser.
- **Tester:**

Feature 2: User Registration

Objective:
Ensure that users can register new accounts with unique usernames and that errors display for duplicate or incomplete input.

Test Case 1: Successful Registration

- **Test Data:**
 - Username: newuser
 - Password: newpassword123
- **Steps:**
 1. Go to /register.
 2. Fill out all fields and click "Register."
- **Expected Result:**
User is redirected to the login page after successful registration.
- **Actual Result:**
Works as expected. User appears in the database.
- **Test Environment:**
Localhost, Node.js/Express backend, PostgreSQL database, Chrome browser.
- **Tester:**

Test Case 2: Duplicate Username Registration

- **Test Data:**
 - Username: testuser (already registered)
 - Password: password123
- **Steps:**
 1. Go to /register.
 2. Enter a username that already exists.
 3. Click "Register."