1. 014-7
2. Weathermen
3. Names
    a. Daniel Iyasu, daiy1557@colorado.edu, daiy1557
    b. Sebastian Martinez, sema5085@colorado.edu sebastianmtzz
    c. Max Wang, mawa5097@colorado.edu, MaxWang9528
    d. Brody Wingrove, brwi8575@colorado.edu, BroD54
4. We-ather
5. Our application has all the features that a standard weather application would require. This includes the ability to search for location-specific weather forecasts using the National Weather Service's API (https://api.weather.gov/) and the ability to set favorite locations. Users will be able to create an account to save locations. In addition to these standard features, users will be able to upload location-specific pictures and comments about the weather.
6. Our target audience is the typical weather app user looking to find a more trustworthy source of information with community-shared images to actually see the weather.
7. To be Earth's most customer-centric weather app, which puts weather in the hands of the people.
8. https://github.com/CU-CSCI3308-Fall2025/group-project-We-ather
9. We will be using the Agile methodology with weekly sprints
10. We have a Discord channel set up for communication throughout the project, with a weekly meeting.
11. We will have team meetings over Discord on Tuesday at 1 PM. We will have weekly meetings with our TA on Wednesdays at 5 PM.
12. In Github
13.
    a. Pages:
        i. Home Weather
        ii. Account
        iii. Upload
    b. In Github

Features Brainstorm:

- Live Cams
- Collaborative Planning between users for Events based on forecast
- What to wear based on what the weather is like
- Nearest location

- Game features
- **Crowd Sourcing/reporting from users to show an accurate depiction of weather**

Discord: https://discord.gg/RNxwaBB3

Risk Management Plan

1. Risk: External API Failure or Rate Limiting

- **Description:** The application relies entirely on the National Weather Service (NWS) API for forecast data. If the API goes down, changes its data structure, or if the application hits rate limits due to high traffic, the core feature of the app will break.
- **Severity: High** (The app loses its primary utility without this data).
- **Mitigation Strategy:** Implement local data caching so users can see the last known weather if the API fails. Implement distinct error handling to inform the user if the service is down rather than the app just crashing. Limit API calls by only refreshing data after a set time interval (e.g., 15 minutes) rather than on every page load.

2. Risk: Inappropriate User-Generated Content (Malicious Uploads)

- **Description:** A core feature of "We-ather" is allowing users to upload pictures and comments. There is a risk that users will upload offensive images (NSFW), spam, or hate speech instead of weather photos, which violates typical hosting policies and ruins the "trustworthy" goal of the app.
- **Severity: High** (Reputational damage and potential removal from hosting platforms).
- **Mitigation Strategy:** Implement a "Report" button on all user posts so the community can flag bad content. Create a simple "Admin" view where team members can manually delete flagged posts from the database.

3. Risk: Scope Creep (Over-ambitious Features)

- **Description:** The "Features Brainstorm" list includes complex items like "Live Cams," "Collaborative Planning," and "Game features." Trying to implement these alongside the core MVP (Minimum Viable Product) within a single semester could lead to half-finished features and a buggy final product.
- **Severity: Medium** (Could result in a poor grade or incomplete project).
- **Mitigation Strategy:** Strictly adhere to Agile methodology. The team will prioritize the MVP (Login, Weather API, Image Upload) first. All brainstormed features (Games, Live

Cams) will be placed in the "Product Backlog" and only attempted if the MVP is 100% stable and time permits.

4. Risk: Database Storage Scalability (Image Hosting)

- **Description:** Storing user-uploaded images can quickly consume available storage space or database limits, especially on free-tier hosting often used for student projects. Storing images directly as BLOBs in a database can also slow down query performance significantly.
- **Severity: Medium** (Can cause app performance issues or crash the database).
- **Mitigation Strategy:** We will limit the file size of uploads (e.g., max 2MB) and file types (JPEG/PNG only). If possible, we will store image *references* (URLs) in the database and store the actual files in a separate object storage system (like an S3 bucket or a dedicated file folder) rather than the database itself.

5. Risk: Version Control Conflicts (Git Merge Issues)

- **Description:** With four developers (Daniel, Sebastian, Max, Brody) working on the same repository simultaneously, there is a high risk of merge conflicts, overwriting each other's work, or breaking the "main" build, specifically when connecting the frontend to the backend.
- **Severity: Medium** (Causes delays and frustration).
- **Mitigation Strategy:** Enforce a strict Branching Strategy. No one pushes directly to the `main` branch. Developers must work on feature branches (e.g., `feature/login-page`) and submit Pull Requests. These PRs must be reviewed by at least one other team member before merging.