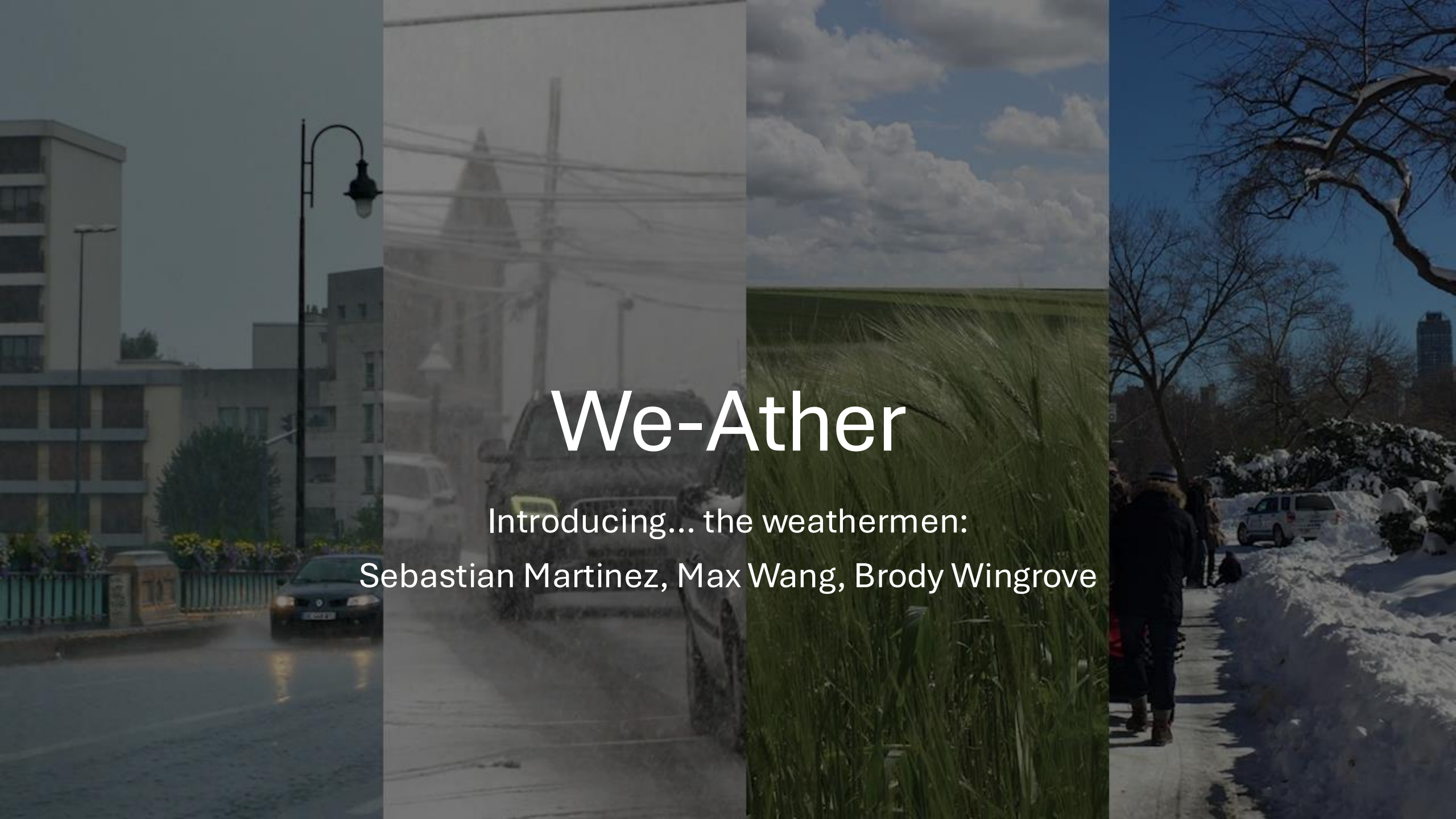


We-Ather

Introducing... the weathermen:

Sebastian Martinez, Max Wang, Brody Wingrove

We-ather Description

We-ather is a crowd-sourced weather platform that puts forecasting in the hands of the people. Traditional weather apps often miss what's actually happening at a local level, so our site collects real-time observations from users, like temperature, wind, rain, and photos, and combines them with official data from Weather.gov. This creates a constantly updated, community-driven snapshot of current conditions that is more accurate, relevant, and local than standard forecasts. We-ather makes weather collaborative, turning users into contributors instead of passive receivers, and helping people make better decisions based on what's truly happening around them, not just what a model predicts.

What we worked on

- Seb:
  - Back end, uploading photos, deleting posts, photo feed
- Brody:
  - Front end, handlebars, search bar, autocomplete
- Max:
  - Back end, database, weather api

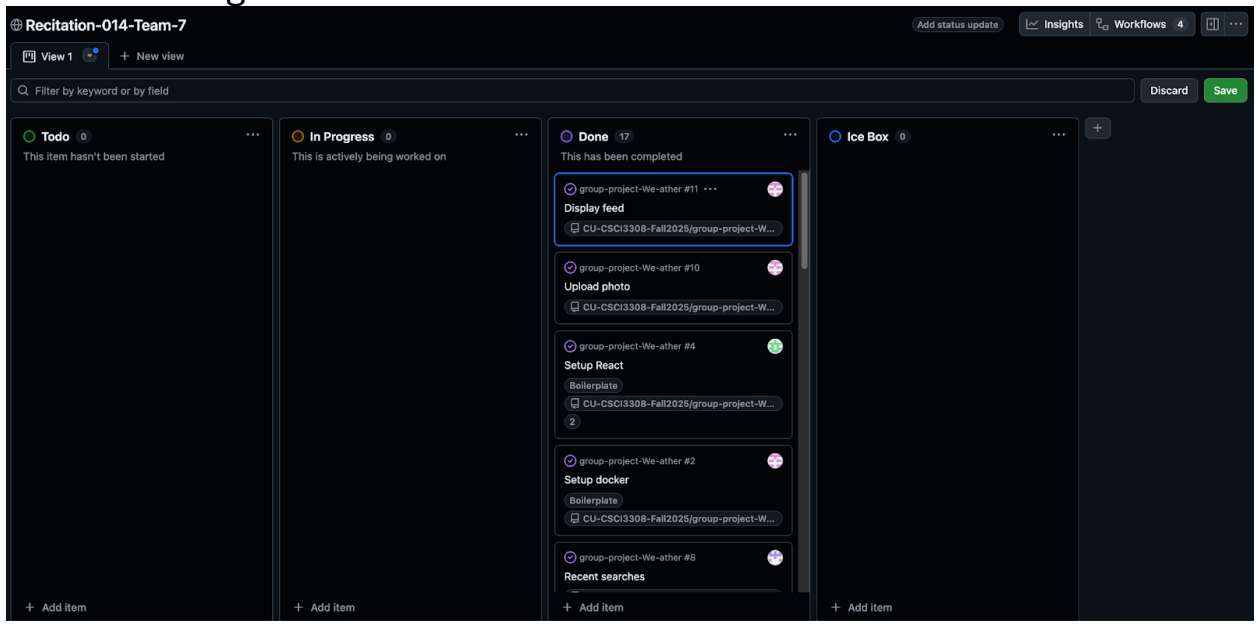
Tools



- Docker : We used Docker to containerize our application, making it easy to run the same environment on every machine and deploy consistently without configuration issues.
- Mocha: Mocha was used for automated testing, helping us verify that core features worked correctly and preventing bugs from breaking our app as we developed.
- Handlebars: Handlebars served as our template engine, allowing us to dynamically generate HTML pages and display user-submitted weather data in a clean, organized way.
- NodeJS: NodeJS was the backend runtime for our application, enabling us to build a fast, scalable server that handled requests, data processing, and API interactions.
- Render: Render was our hosting and deployment platform, allowing us to deploy our app online so users can access it without needing local installation.
- **Weather.gov API(4): The Weather.gov API provided official weather data, which we combined with crowd-sourced reports to compare and enhance forecasting accuracy.**
- **Multer(5): Multer handled file uploads, letting users submit photos or images alongside their weather reports.**
- **OpenStreetMap(3): Used for autocompleting locations**
- PostgresSQL: PostgreSQL was our database, storing user submissions and weather data so it could be retrieved, visualized, and analyzed in real time.
- Github: GitHub was our remote repository and collaboration platform, where we stored our code, tracked changes, managed issues, and worked as a team.
- Git: Git was our version control system, helping us manage code changes, roll back mistakes, and coordinate development without overwriting each other’s work.
- Express: Express was our framework for handling routes and requests, helping us build and connect our backend quickly and efficiently.

Project Management

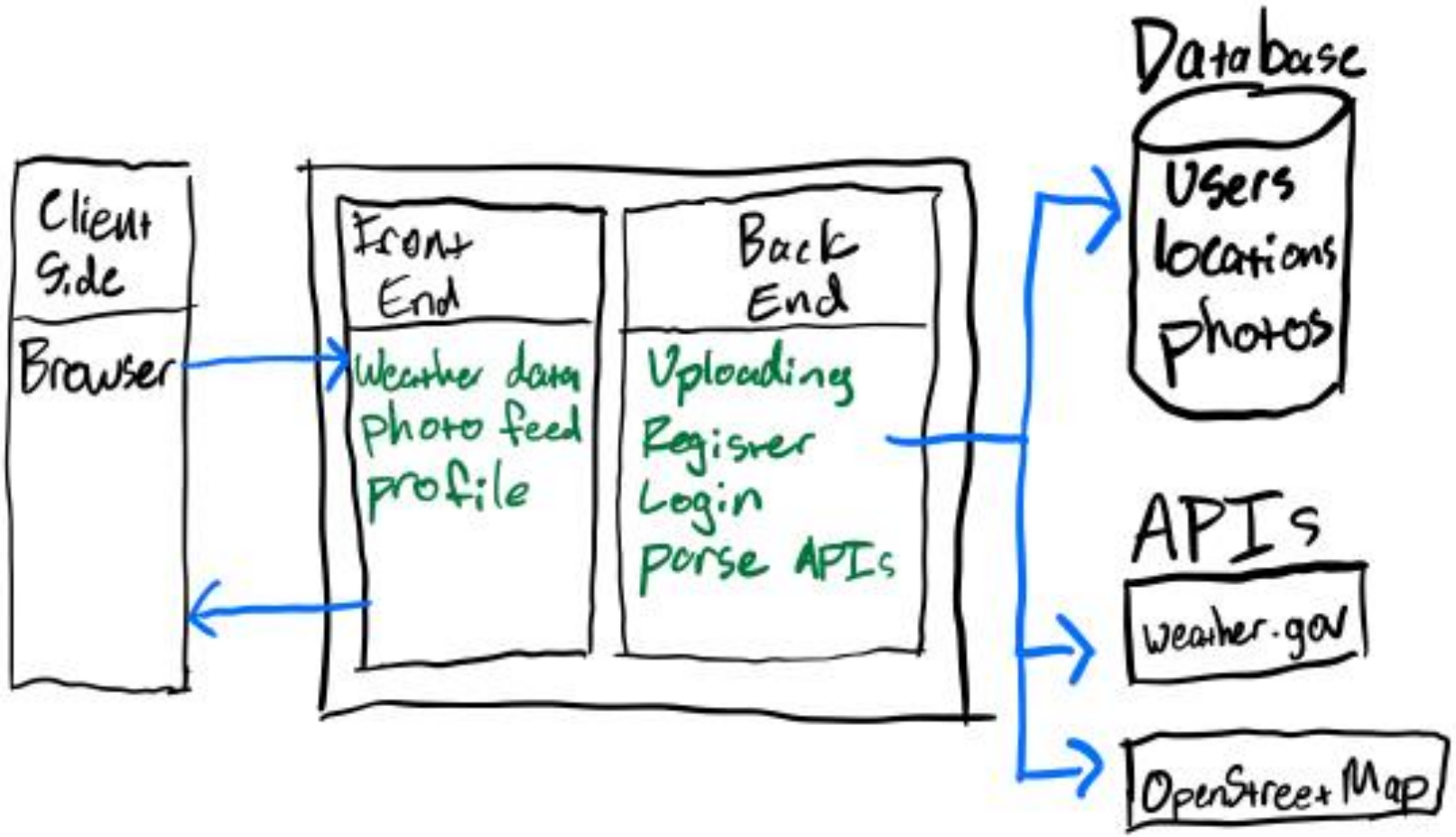
- **Agile Methodology with weekly sprints**
- Teem meetings every Tuesday at 1PM
- Weekly meetings with our TA every Wednesday at 5PM
- Discord: Discord was our team communication hub, used for quick updates, planning, sending resources, and coordinating development tasks.
- Zoom: Zoom allowed us to meet remotely, discuss progress, assign tasks, and work through problems when we couldn't meet in person.
- GitHub: Project board allowed us to break down the project into manageable user stories.



Challenges

- **Time Management & Scheduling**
  - **Challenge:** Coordinating meetings and dividing work was difficult with different schedules, classes, and workloads.
    How we overcame it: We used **Discord, Zoom, and Messages** to stay connected, assign tasks, and hold quick check-ins.
- **Code Merge Conflicts**
  - **Challenge:** Multiple people working on the same files led to broken builds and merge conflicts.
    How we overcame it: We used **Git branching and pull requests**, reviewed code before merging, and communicated changes better.
- **Handling User-Submitted Data**
  - **Challenge:** Processing and storing user weather submissions and file uploads was complex, especially validating input and avoiding crashes.
    How we overcame it: We used **Multer for file handling**, added validation, and built error handling to prevent bad data from breaking the app.

Architecture Diagram



DEMO

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

Future Scope / Enhancements

1. Most users would use this app from their phone, so we should implement the ability to upload photos from camera roll or take photo from phone camera.
2. Adding more social features to the photo feed, such as comments, likes, adding friends, etc.
3. Expanding the profile page to add quality of life features, such as sorting old posts, or seeing a photo feed of a location from a different day.

QUESTIONS?

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