WeatherEase

ACCESSIBLE WEATHER APP FOR BLIND AND VISUALLY IMPAIRED USERS

Group Members:

- Tajra Sinanagic,
- Ehab Mohamed,
- Mike A. Khadeida.

Supervisor:

- Basak Taylan (CUNY Brooklyn College)

Course: CISC 4900

Date: Sept 2025

Mission & Problem Statement

Easy Breezy Forecasts

WeatherEase is designed for blind and visually impaired users, making weather updates simple, fast, and inclusive.

Our mission is clear: ***to ensure everyone, rain or shine, can access the forecast with ease.***

GOAL: By replacing visuals with spoken updates and intuitive voice controls, WeatherEase transforms weather checking into an effortless and empowering experience.

WeatherEase: Purpose, Research, and Learning Path

PURPOSE

- Hands-free weather via voice commands
- Progressive Web App with offline support
- Clear, readable forecast cards
- Bridges general and accessibility-focused designs

RESEARCH

- Existing apps: Weather Gods, WeatherWheel, Weather for the Blind
- Mainly audio output & screen reader support
- WeatherEase: UI + voice + offline, general & accessibility-friendly
- Coding examples: Stack Overflow, GitHub, Web.dev

FUTURE STUDENT ROADMAP

Weeks 1–2: HTML/CSS basics, explore API
Weeks 3–4: Display weather, basic UI
Weeks 5–6: Text-to-speech (voice output)
Weeks 7–8: Speech recognition (voice input)
Weeks 9–10: Convert to PWA, offline support
Weeks 11–12: Accessibility & UX polish
Week 13: Final deployment, bug fixes

- **WeatherEase shows how we applied what we've learned in class—like working with data structures, arrays, and APIs—to a real-world application.**
- **Our goal was to take a common app and make it more inclusive and accessible, since most student projects rarely focus on accessibility.**
- **This project gave us hands-on experience in building a PWA, integrating voice commands, and designing for all users, while highlighting the value of thoughtful, inclusive coding.**

ORGANIZATION CHART

TAJRA – Team Lead
sinanagict@gmail.com

CODING
- Repo setup
- API
- Speech features

LEAD
- Creating Weekly Plans
- Coordinating tasks
- Reviewing progress

EHAB – Individual Contributor
ehabm7986@gmail.com

CODING
- UI structure
- Error handling
- Command mapping
- Testing & Debugging
- Mutual collaboration in code with Tajra

MIKE – Individual Contributor
mike.khadeida@gmail.com

RESEARCH
- Accessibility guidelines
- User testing plans
- Reporting findings
- Supporting UI/UX improvements

BASAK TAYLAN – Supervisor
basak.taylan@brooklyn.cuny.edu

Team ClearPath

TEAM COLLABORATION & WORKFLOW

COMMUNICATION

Discord, Google Docs, and Notion

- Held Discord calls and in-person library meetings (avg. 3 hours per call/meeting) to catch up and plan ahead

COLLABORATION

Cross-Collaborated despite defined roles

- Google Docs mirrored Notion but allowed faster access for notes, new tasks, and review markers.

COORDINATION

Task Management

- Assigned tasks based on comfort and adjusted workloads as needed.
- Discussed and agreed on task order to balance fairness, and schedule.

Through open communication, strangers became collaborators, and together we turned our individual strengths into a shared success.

Team ClearPath

Tentative Schedule – Weeks 3–5 (Approx. 15 hrs/week)

Week	Tajra	Ehab	Mike
3: Sep 8 –14	• repo set up • create starter files (index.html, style.css, script.js, manifest.json, service-worker.js) • push to GitHub • enable Pages • basic planning	• build HTML layout (<header>, <main>, <footer>) • add weather button • test keyboard nav • minor CSS • debug layout	• write accessibility guide for blind users • share with team • review HTML semantic usage
4: Sep 15 – 21	• register OpenWeather API key • fetch weather data handle responses • debug integration • update scripts	• display API data on page • style temperature/humidity/wind info • test keyboard access • minor UI fixes	• update accessibility checklist with ARIA attributes • note progressive enhancement • review dynamic content
5: Sep 22 – 28	• implement modular TTS function using SpeechSynthesis API • test with sample text • debug	• connect button to TTS • provide visual feedback while reading • test keyboard/mouse	• suggest concise TTS wording • test voice settings • guide on reading dynamic info clearly

This schedule shows planned tasks and estimated time per team member for Weeks 3–5, including coding, research, testing, and coordination.

Commits on Aug 27, 2025

First commit
tajrasinanagic0 committed on Aug 27

Removed test.txt
gkaiser266 committed on Aug 27

Added test.txt file
gkaiser266 committed on Aug 27

Removed test.txt
gkaiser266 committed on Aug 27

Added test.txt file
gkaiser266 committed on Aug 27

Deleted test.txt
gkaiser266 committed on Aug 27

Added test.txt file
gkaiser266 committed on Aug 27

update 6
Mike Khadeida auth

Commits on Aug 27, 2025 (Week 1)

First meeting: Figuring out GitHub as collaboration together.

Commits on Sep 8, 2025

Changed in PWA short name from WE to WeatherEase
tajrasinanagic0 committed on Sep 8 · ✓ 3 / 3

Updated icon's names
tajrasinanagic0 committed on Sep 8 · ✓ 3 / 3

Updating PWA in index for icon chang
tajrasinanagic0 committed on Sep 8 · ✓ 3 / 3

Update PWA manifest and add weath
tajrasinanagic0 committed on Sep 8 · ✓ 3 / 3

Ignore .DS_Store files going forward
tajrasinanagic0 committed on Sep 8 · ✓ 3 / 3

Update README with project details
tajrasinanagic0 committed on Sep 8 · ✓ 3 / 3

Update README with project details and team info
tajrasinanagic0 committed on Sep 8 · ✓ 3 / 3

Delete CNAME

Commits on Sept 8, (Week II)

Learning to set up basic pages, working directories, potentially needed files

Commits on Sep 16, 2025

Updpate bug fix for Dark mode
Mike Khadeida authored and Mike Khadeida committed on Sep 16 · ✓ 3 / 3

bug fixed for dark mode
Mike Khadeida authored and Mike Khadeida committed on Sep 16

Debugging home page, weathe
tajrasinanagic0 committed on Sep 16 ·

Updated manifest.json to first o
tajrasinanagic0 committed on Sep 16 ·

Created Home Page and Exit pa
tajrasinanagic0 committed on Sep 16 ·

Upgraded API plan for better fe
tajrasinanagic0 committed on Sep 16 ·

Debugging API as error in fetching location
tajrasinanagic0 committed on Sep 16 · ✓ 3 / 3

Debugging API errors of incorrect temperature metric
mitted on Sep 16 · ✓ 3 / 3

ound

Commits on Sept 16, (Week III)

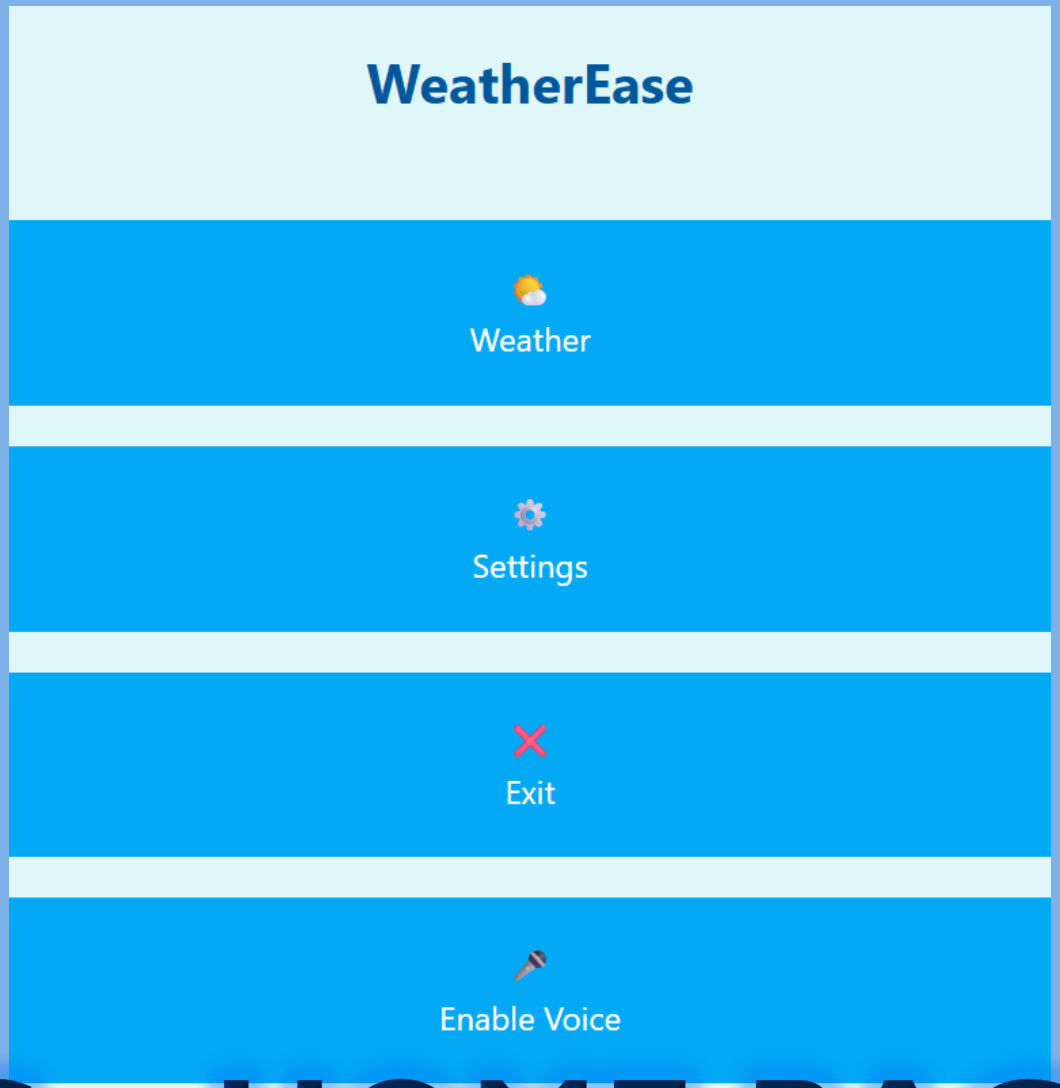
Already implementing API, learning how to debug, adding features we learned in previous courses or from research.

TIMELINE ROADMAP WEEKS 1-3

Why these weeks matter?

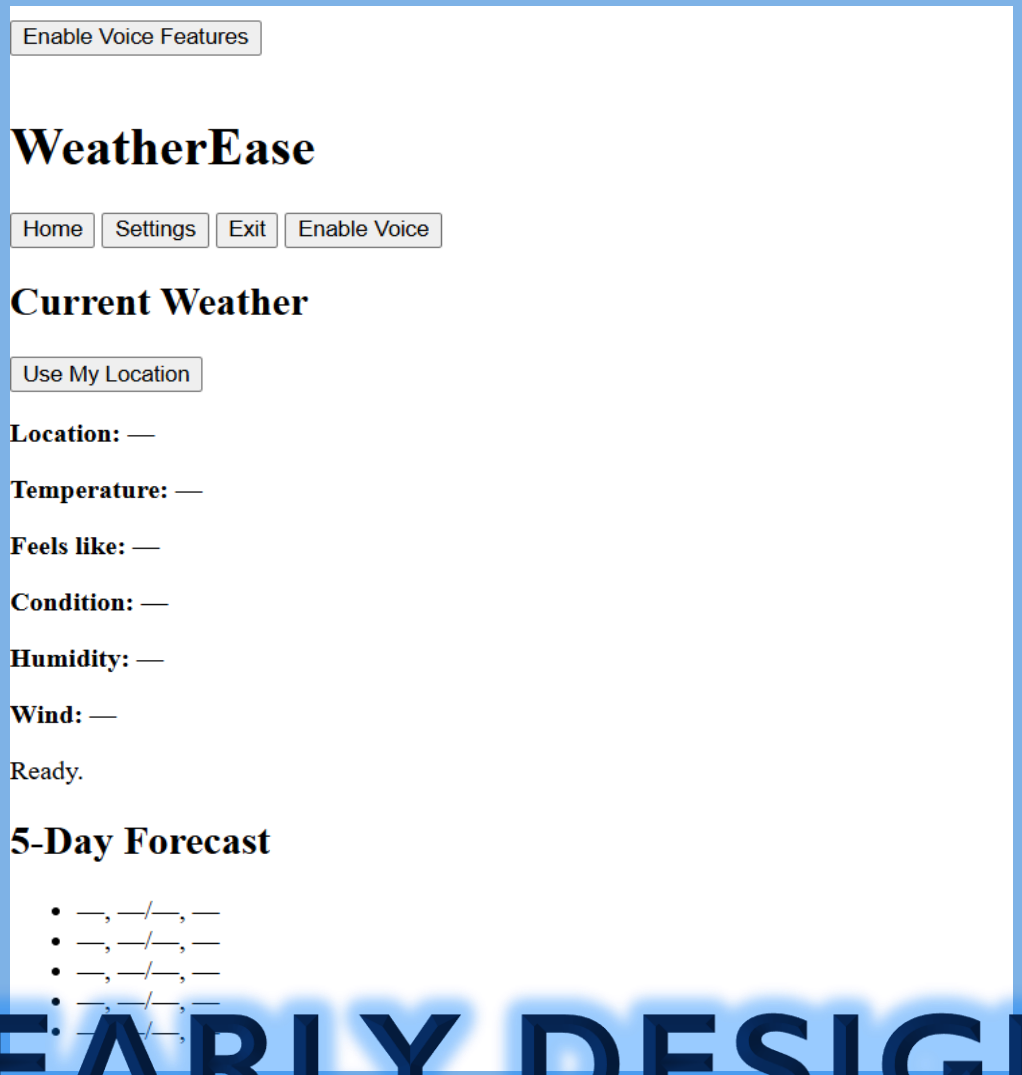
In just three weeks, our team went from zero to building a working app. Week 1 was all about figuring out GitHub and finding our rhythm. **Week 2**, we laid the foundation—setting up files, doing research, and planning how to turn ideas into code. By **Week 3**, we made it real: a progressive web app with live weather data.

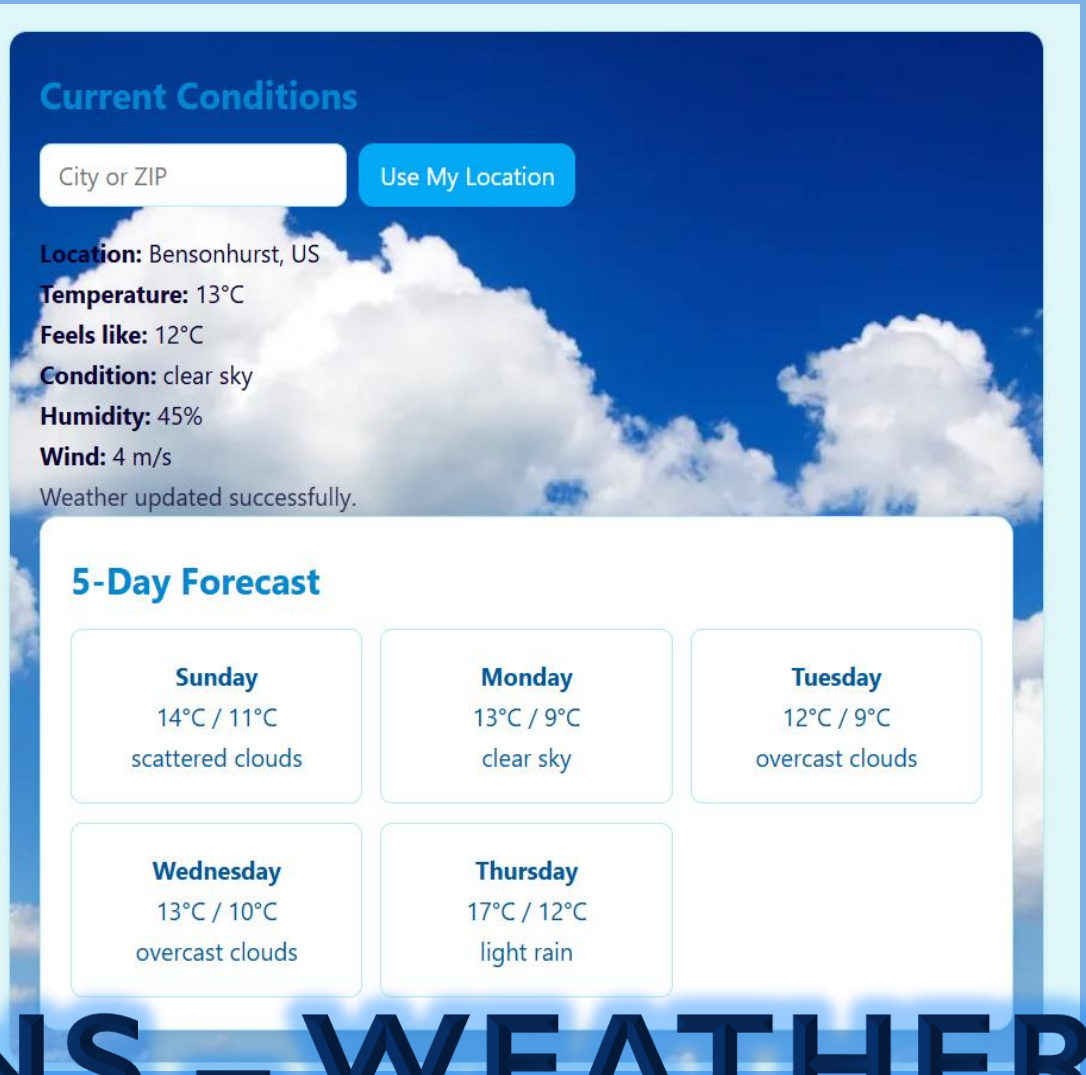
These weeks show more than progress—they show how teamwork, open communication, and persistence can turn beginners into a team capable of creating something functional and meaningful.



EARLY DESIGNS – HOME PAGE

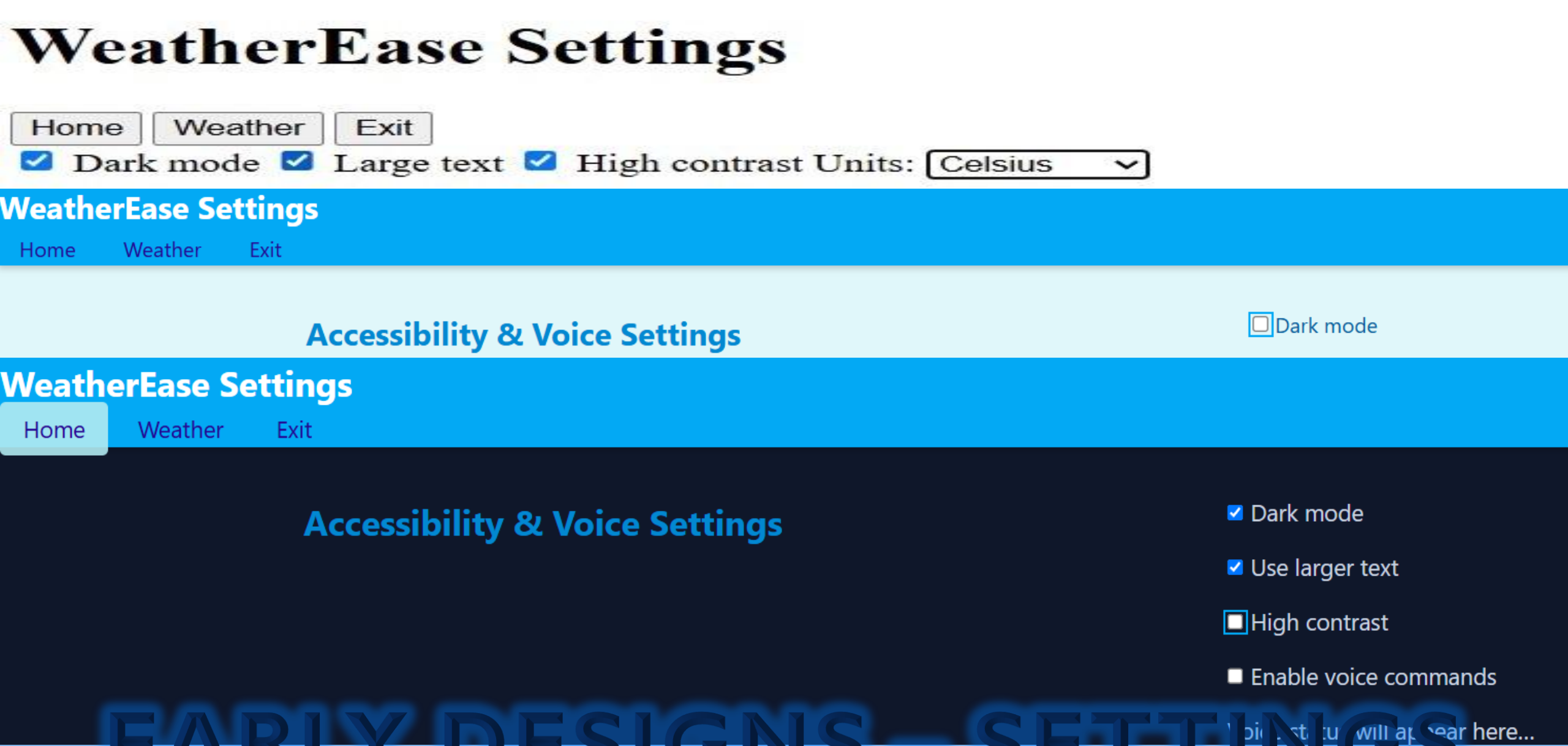
Early vs. Current: Started with basic layout and buttons; now improved navigation and responsive design.





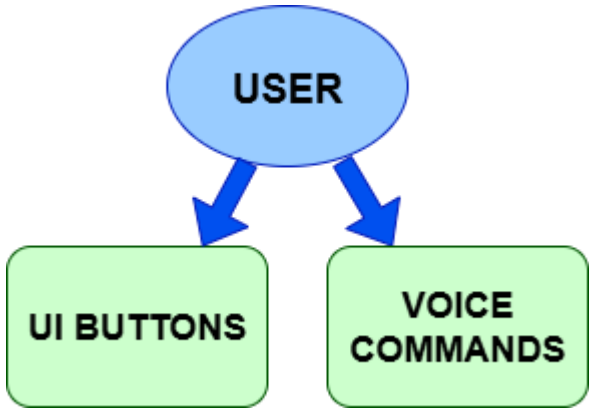
EARLY DESIGNS – WEATHER

Early vs. Current: From a static design to fully functional weather API integration with interactive search and forecast



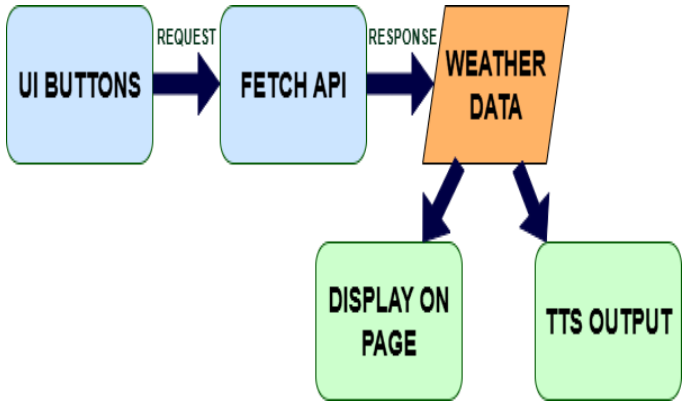
Early vs. Current: Initial static options evolved into user-friendly settings with functioning dark mode and large text.

User Input Modalities



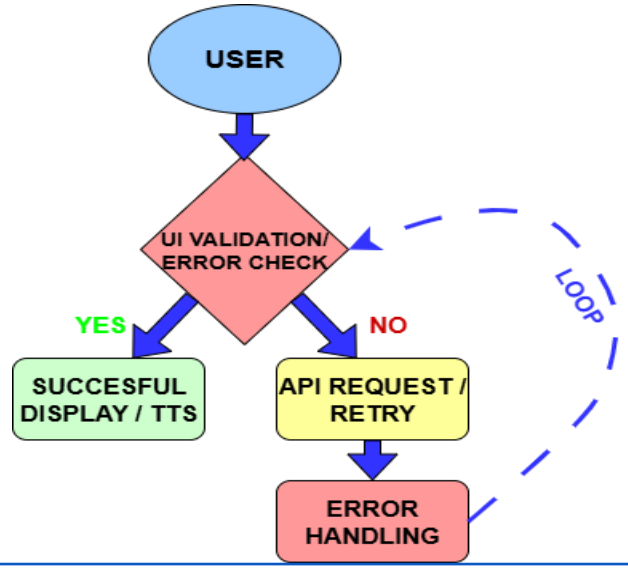
This diagram shows how the user interacts with the app, either by pressing buttons to get weather or listen, or by issuing voice commands for specific information.

Data Flow Diagram



This diagram shows how data flows through the app: the user interacts with UI buttons, which trigger API requests. The API retrieves weather data, which is then displayed on the page and read aloud via text-to-speech

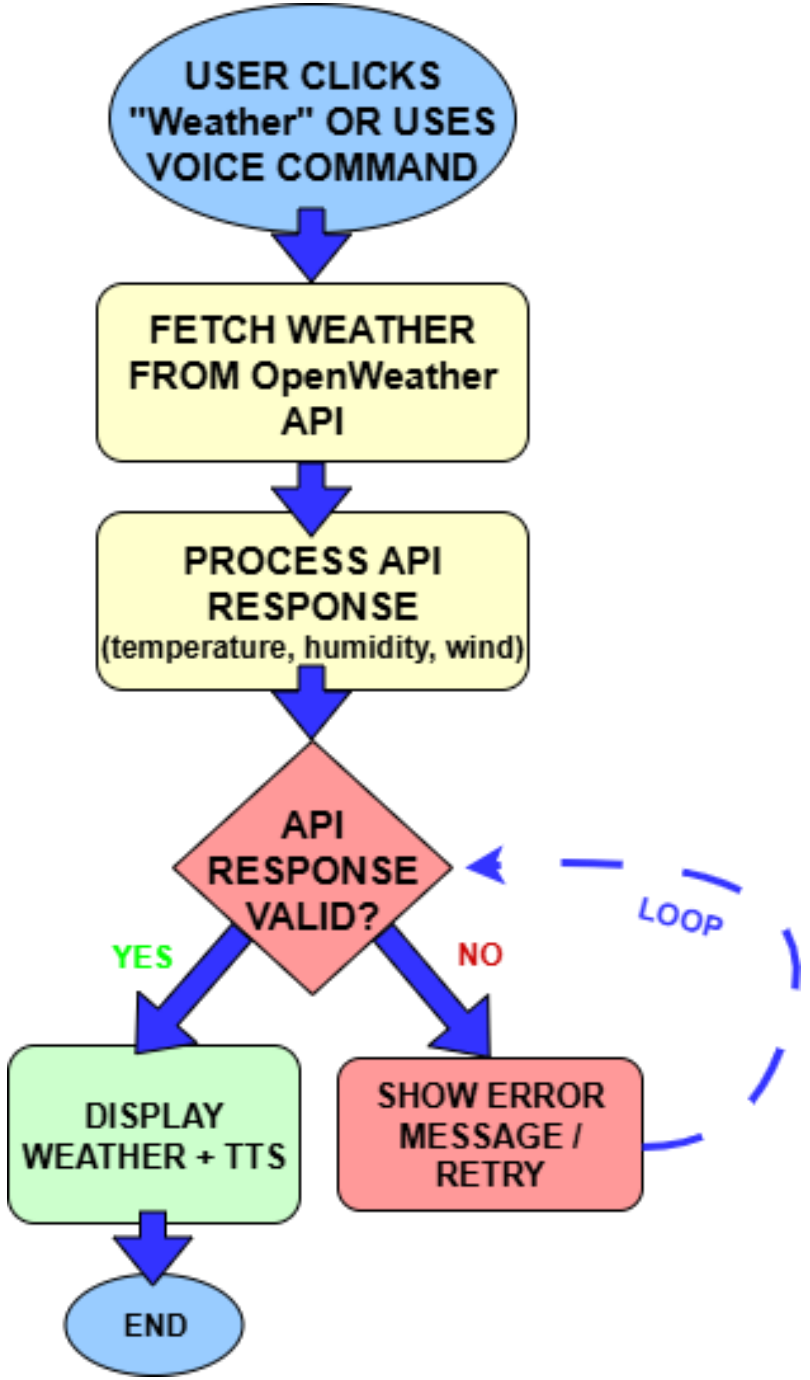
Error Handling Flow



This diagram shows how the app handles errors: input is validated, and if successful, data is displayed and read aloud. If validation fails, the API request handles errors and loops back to re-check the input.

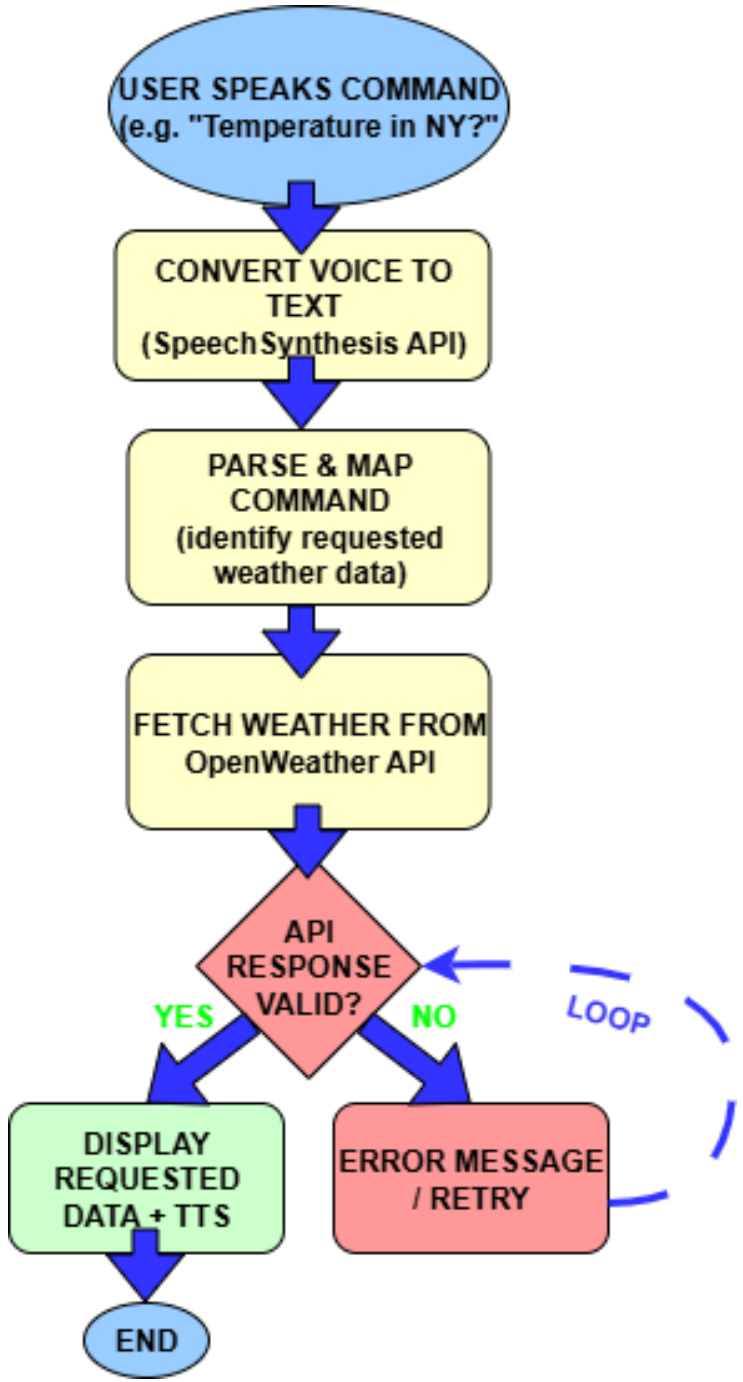
CASE 1: CHECK CURRENT WEATHER

- In this use case, the user checks the current weather by clicking the 'Get Weather' button or using a voice command.
- The app fetches data from the OpenWeatherMap API, processes the temperature, humidity, and wind speed, and displays the results on the screen.
- Optionally, the system reads the information aloud using text-to-speech.
- If the API response is invalid, an error message is shown, and the system automatically retries the request.



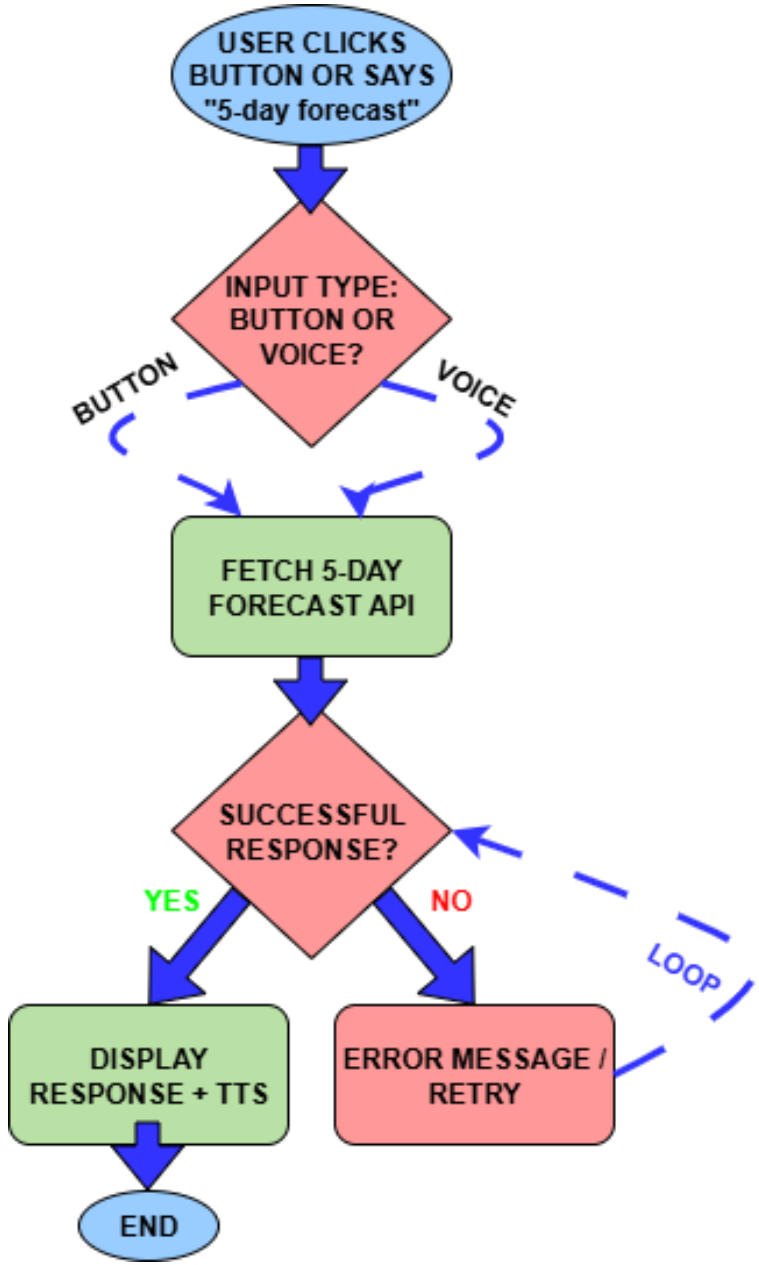
CASE II: VOICE QUERY FOR WEATHER DETAILS

- In this scenario, the user speaks a command, such as “What’s the temperature in New York?”.
- The system first uses the SpeechSynthesis API to convert the spoken input into text, then parses and maps the command to identify the requested weather data.
- A call is made to the OpenWeatherMap API, and if the response is valid, the result is displayed on the screen and also read aloud using text-to-speech for convenience.
- If the API request fails or the command cannot be understood, the system provides an error message and prompts the user to retry.



CASE III: FETCHING A 5-DAY FORECAST

- In this scenario, the user requests a “5-day weather forecast.”
- The system calls the OpenWeatherMap Forecast API and retrieves hourly and daily forecast data for the next five days.
- The information is displayed in a clean format on the screen (such as date, temperature, humidity, and conditions).
- At the same time, a summary of each day’s weather is read aloud using text-to-speech for accessibility.
- If the API call fails, the user is shown an error message with the option to retry.



CODE HIGHLIGHT – VOICE COMMAND BUG

```
// ---------------------------
// Handle voice commands
// voice-control-weather.js
// Trimmed version for readability
// ---------------------------
function handleCommand(cmd) {
  if (!window.weatherDataReady) {
    speak("Weather data is not ready yet.");
    return;
  }

  switch(cmd) {
    case '1':
    case 'one':
    case 'temperature':
    case 'current temperature':
      speakField('temp');
      break;
    default:
      speak("Command not recognized. Please try
      break;
  }
}
```

Prevents user voice commands from running before weather data loads.

BUG – What was the problem?

- Users could trigger voice commands before the weather API returned data.
- The app would try to speak temperature or weather conditions that didn't exist yet, resulting in empty or wrong output.

FIX – Solution to the problem

- Added a check for window.weatherDataReady at the start of handleCommand.
- If the data isn't ready, the app notifies the user and exits early.
- This ensures voice commands only respond once valid weather info is available

DATA SOURCE: OPENWEATHERMAP API

DATASET NATURE

- Provides **current weather** and **forecasts** (hourly up to 4 days, daily up to 16 days).
- Includes **historical data** and statistical weather parameters.
- Offers **weather maps** with multiple layers and air pollution data.
- Accessible via **geocoding API** for location-based queries.
- Responses are structured in **JSON format** and updated in real-time.
- API usage limits: 3,000 calls/minute, 100 million calls/month for current/forecast data; 50,000 calls/day for historical data.

TOOLS & TECHNOLOGIES

LANGUAGES
- HTML5
- CSS3
- JavaScript

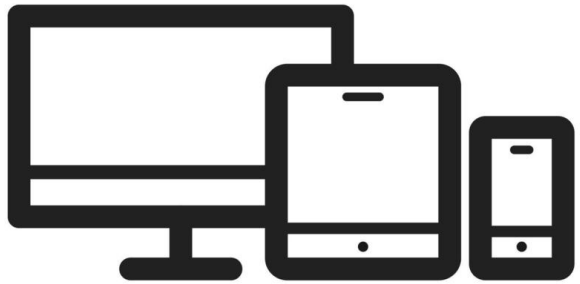
API
- OpenWeatherMap
- Web Speech API
- SpeechSynthesis API

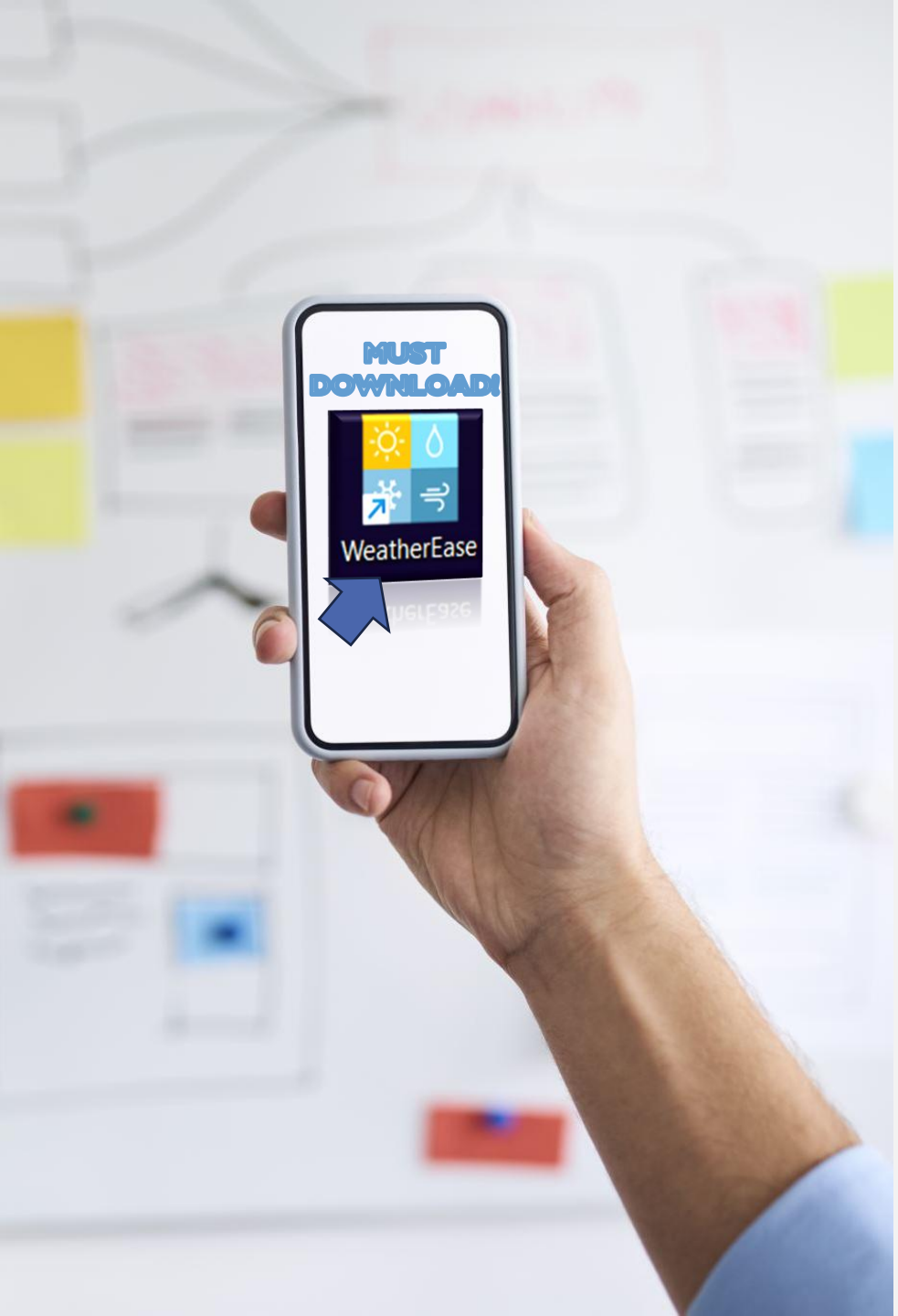
PWA COMPONENTS
- manifest.json (app metadata & install behavior)
- Service Worker (offline caching)

HOSTING
- GitHub
- GitHub Pages

PROJECT MANAGEMENT
- Notion
- Google Docs

DEVICE TESTING
- Chrome
- Microsoft Edge
- Safari (iOS)
- Android browsers (Google Play devices)





LINKS

- **GITHUB:** https://github.com/Project4900
- **PROJECT MANAGEMENT BOARD:** WeatherEase | Kanban
- **GOOGLE DOCS:** https://docs.google.com/document/d/1IGvmm2wwOGerVoRkflSTbxjR2IwU7BI5_7Lgd7Z1JMg/edit?usp=sharing
- **WEBSITE:** WeatherEase