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

Applying Classroom Knowledge to a Real-World, Accessible App

WeatherEase allowed us to **apply concepts learned in class**—such as working with **data structures, arrays, and APIs**—to a real-world project.

Our goal was to **take a standard weather app and make it inclusive and accessible** for blind and visually impaired users. Through this project, we gained **hands-on experience building a Progressive Web App (PWA), integrating voice commands**, and **designing UI/UX that serves all users**.

This work demonstrates how **thoughtful, inclusive coding** can transform everyday applications into tools that empower a wider audience.

Compared to a **typical iOS weather app**—which displays mostly numbers with minimal description and forces the user to interpret the conditions.

WeatherEase provides natural, easy-to-understand descriptions, supportive emojis, and full text-to-speech accessibility.

This makes the weather **clearer, more readable, and instantly understandable**, especially for users who rely on audio output or need quick, descriptive insights.

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

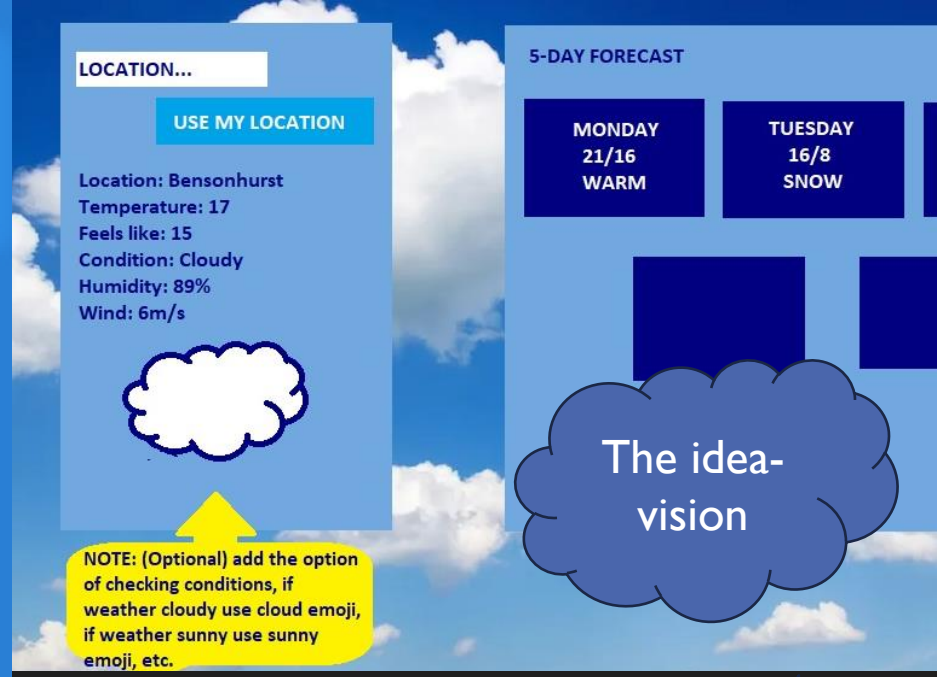
TAJRA – TEAM LEAD

CODING

- Set up project repository and file structure (HTML, CSS, JS, manifest.json, service-worker.js)
- Integrated **OpenWeatherMap API** for live weather and forecast data
- Developed **Text-to-Speech (TTS) module** using the SpeechSynthesis API
- Implemented **voice command recognition** via SpeechRecognition API
- Built dynamic, **interactive UI elements** compatible with keyboard navigation
- Ensured **offline support** using Service Worker for PWA functionality
- Debugged complex issues, including asynchronous API calls and voice command timing

LEADERSHIP

- Created weekly plans and set team milestones to keep project on track
- Coordinated task assignments based on team strengths and availability
- Monitored progress and ensured deadlines were met for each development phase
- Reviewed teammates' work and provided constructive feedback for quality improvements
- Facilitated team communication via Discord, Google Docs, and Notion
- Organized and led team meetings (in-person and virtual) to discuss challenges and next steps
- Balanced collaboration between coding and research roles to maximize efficiency
- Tracked and updated the project roadmap, ensuring alignment with semester goals




```
// ---------------------------
// Emoji helper
// ---------------------------
function getConditionEmoji(desc) {
    if (!desc) return '';
    desc = desc.toLowerCase();
    if (desc.includes('cloud')) return '☁️';
    if (desc.includes('rain')) return '🌧️';
    if (desc.includes('snow')) return '❄️';
    if (desc.includes('sun') || desc.includes('clear')) return '☀️';
    if (desc.includes('mist') || desc.includes('fog') ||
    desc.includes('haze')) return '🌫️';
    return '❓';
}
```

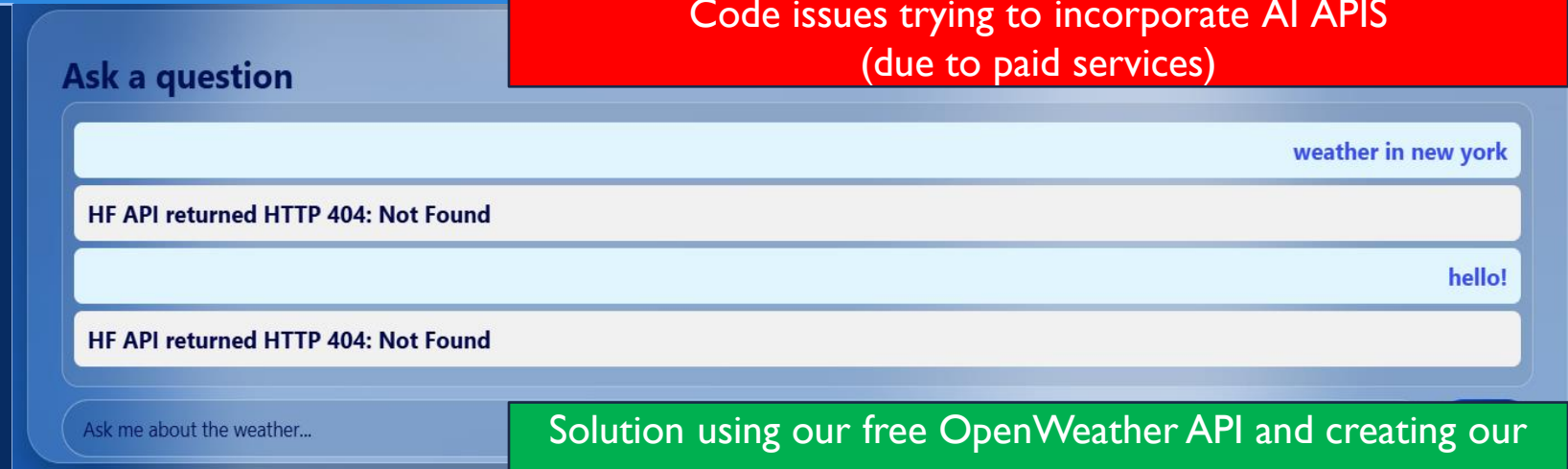
The execution

EHAB – INDIVIDUAL CONTRIBUTOR

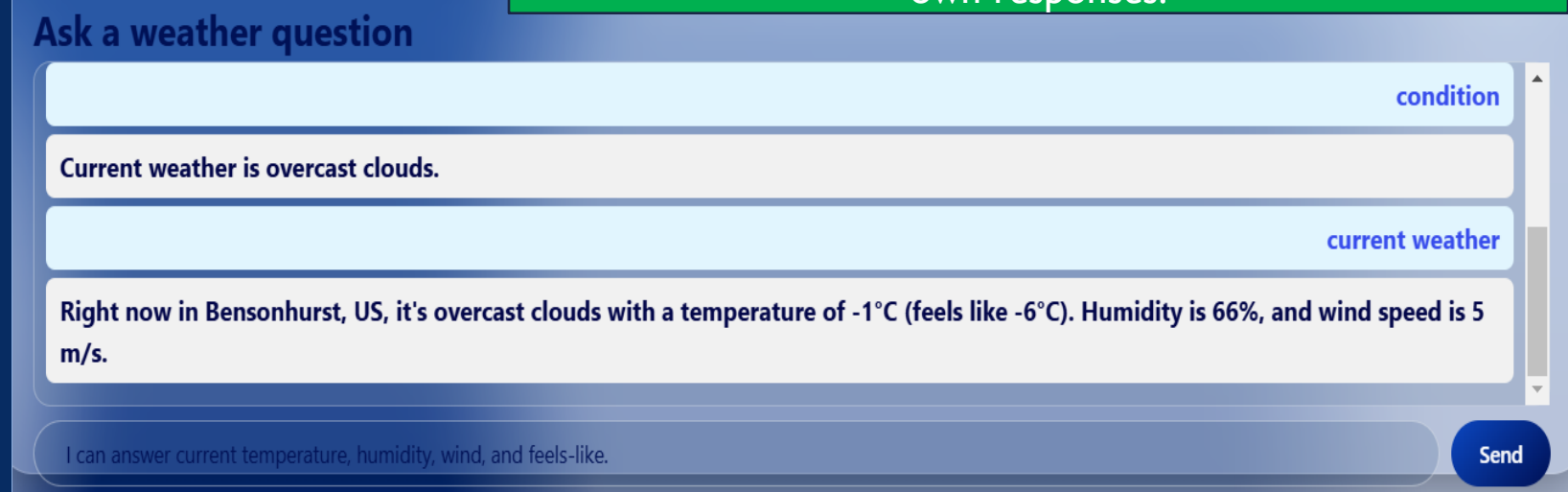
CODING

- Built and structured the UI layout for accessibility and responsiveness
- Implemented error handling for invalid user input and API failures
- Mapped voice commands to actions for smooth user interaction
- Conducted testing and debugging across multiple browsers and devices
- Collaborated with Tajra on integrating TTS and API modules
- Suggested concise TTS wording for better voice output clarity
- Updated accessibility features including ARIA attributes and keyboard navigation
- Proposed AI-responsive chatbot for conversational weather queries
    - Adapted plan to use custom scripted responses due to paid AI limitations

Code issues trying to incorporate AI APIS (due to paid services)



Solution using our free OpenWeather API and creating our own responses.



MIKE – INDIVIDUAL CONTRIBUTOR

RESEARCH AND DESIGN

- Conducted accessibility research and gathered best practices for blind and visually impaired users
- Developed user testing plans to validate UI/UX decisions
- Reported findings and recommendations to improve app accessibility and usability
- Assisted in UI/UX design creation, including layout suggestions, button placement, and color choices
- Supported feature design iterations such as dark mode, large text, and simplified navigation
- Collaborated with team to ensure research insights were applied in code

3. Test Instructions

Read to participant:

During this session, I will ask you to complete several tasks.
There are no right or wrong answers — we are testing the design, not your skills.

4. Tasks for the Usability Test

The link of Websites: https://project4900.github.io/4900-Project-/index.html

Task 1: Visit the website using the link: https://project4900.github.io/4900-Project-/index

Task 2: Locate and explore the main navigation buttons

- **Weather**
- **Settings**
- **Exit**
- **Enable Voice**

Task 3: On the **Weather** page:

- **Enter a location by city or ZIP code**
- **OR select Use My Location**
- **View the weather details provided.**

Review the displayed weather information, including:

- **Current temperature**
- **Weather conditions**
- **5-day forecast**
- **Additional details (humidity, wind, etc.)**

Testing questions examples

Building Apps Accessible for Blind Users

Links for example:

- How Blind People Use Technology (My Apple Products - An Introduction to Voice …
- https://www.bairesdev.com/blog/how-to-build-a-handy-app-for-a-blind-person/

inks: How to use ARIA in HTML to make content accessible for blind users

- What is ARIA-Label and How to Use It Effectively | Web Accessibility Basics
- ARIA HTML Tutorial - What is ARIA & Why it's Important to Use!

Introduction
Making apps accessible for blind users means everyone can use your app.
Accessibility helps people who rely on screen readers and improves usability for Everyone.

1. Support Screen Readers
   - Add **alt text** to all images, icons, and buttons so screen readers can describe them.
   - Use **clear labels** for buttons and links (e.g
     "Button 1"). For example, **Clear labels"** m
     descriptive name that tells the user exactly
2. Make Navigation Easy
   - Users should be able to move around usin
   - Keep the **order of buttons and links log**
     reading with a screen reader.
   - On mobile, make **buttons large** and easy
     gestures.
3. Keep Design Simple and Clear
   - Use **headings and simple layouts** so screen readers can understand

Research with videos and links.

1. What is `aria-label`?

- `aria-label` is an HTML attribute that gives a **text description** to elements that might not have visible text.
- Screen readers use it to tell users what the element does.

2. When should we use it?

- Use it on **custom buttons or interactive elements** that do
- Example: a `<div>` used as a button, or an icon-only button.
- Helps blind users understand the purpose of elements.

3. Code Example:

❌ <!-- Custom button using a div (bad for accessibility with

✅ <!-- Good: use aria-label so screen reader knows what it does -->

Research for coders and good coding practices

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

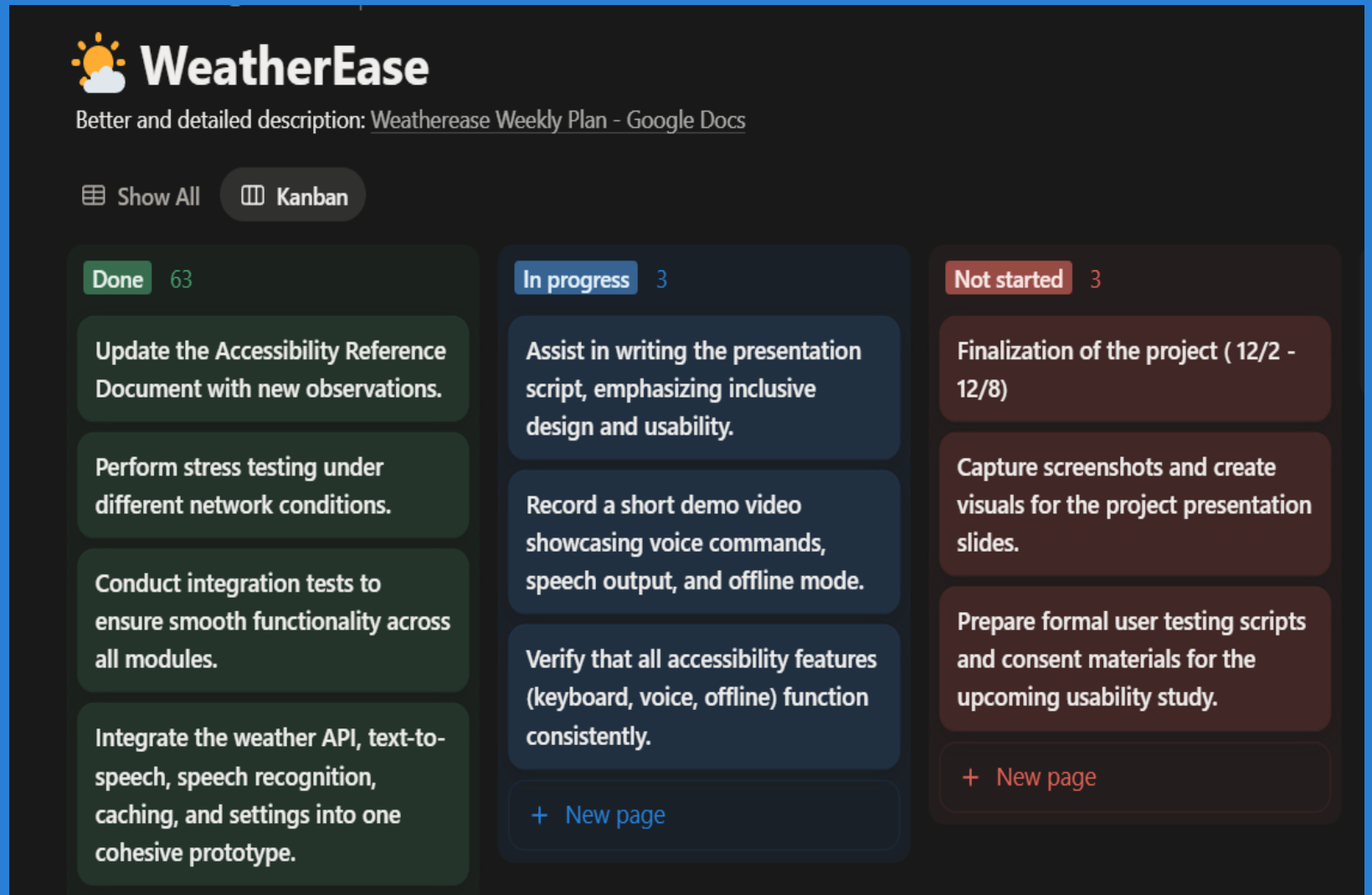
- Assigned tasks based on comfort and adjusted workloads as needed.
- Discussed and agreed on task order to balance fairness, and schedule.

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

Team ClearPath

TEAM COLLABORATION & WORKFLOW

Notion was used to provide brief task descriptions and, more importantly, to visually track progress, showing which tasks were completed, in progress, or not started, and allowing the team to quickly check off items as they were finished.



Team ClearPath

TEAM COLLABORATION & WORKFLOW

Google Docs was used for more detailed task descriptions, organizing weekly tasks for easier navigation, and providing a space for team members to add notes, research, and comments.

GREEN HIGHLIGHT - DONE
NO HIGHLIGHT - NOT STARTED
BLUE HIGHLIGHT - IN PROGRESS OR NEED HELP

MIKE'S GOOGLE DOCS FOR RESEARCH:
https://docs.google.com/document/d/1IdXMVugB1DGSvNd2YNCDB-_55Pqj8XeSBFTlhw96iBc/edit?tab=t.0#heading=h.emxa40sum64s

Week 6 — Integrate API with Text-to-Speech, Add Error Handling, Deploy PoC (Sept 29–Oct 5)

Goal: Connect all previous features into a full workflow and deploy the Proof-of-Concept.

- **Tajra (Lead / Coding):**
  - Integrate the API fetch with the text-to-speech function.
  - Add a service worker to cache the app and API responses.
  - Deploy the working app to GitHub Pages.
- **Ehab (Coding):**
  - Add error handling for cases such as offline status or failed API responses.
  - Improve keyboard navigation and screen reader behavior.
  - Test that the text-to-speech function works correctly in different scenarios.
- **Mike (Research / Accessibility):**
  - Complete the Accessibility Reference Document and write a detailed user testing plan for the Proof-of-Concept.

Deliverables: Deployed Proof-of-Concept, completed accessibility reference, and user testing plan.

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)

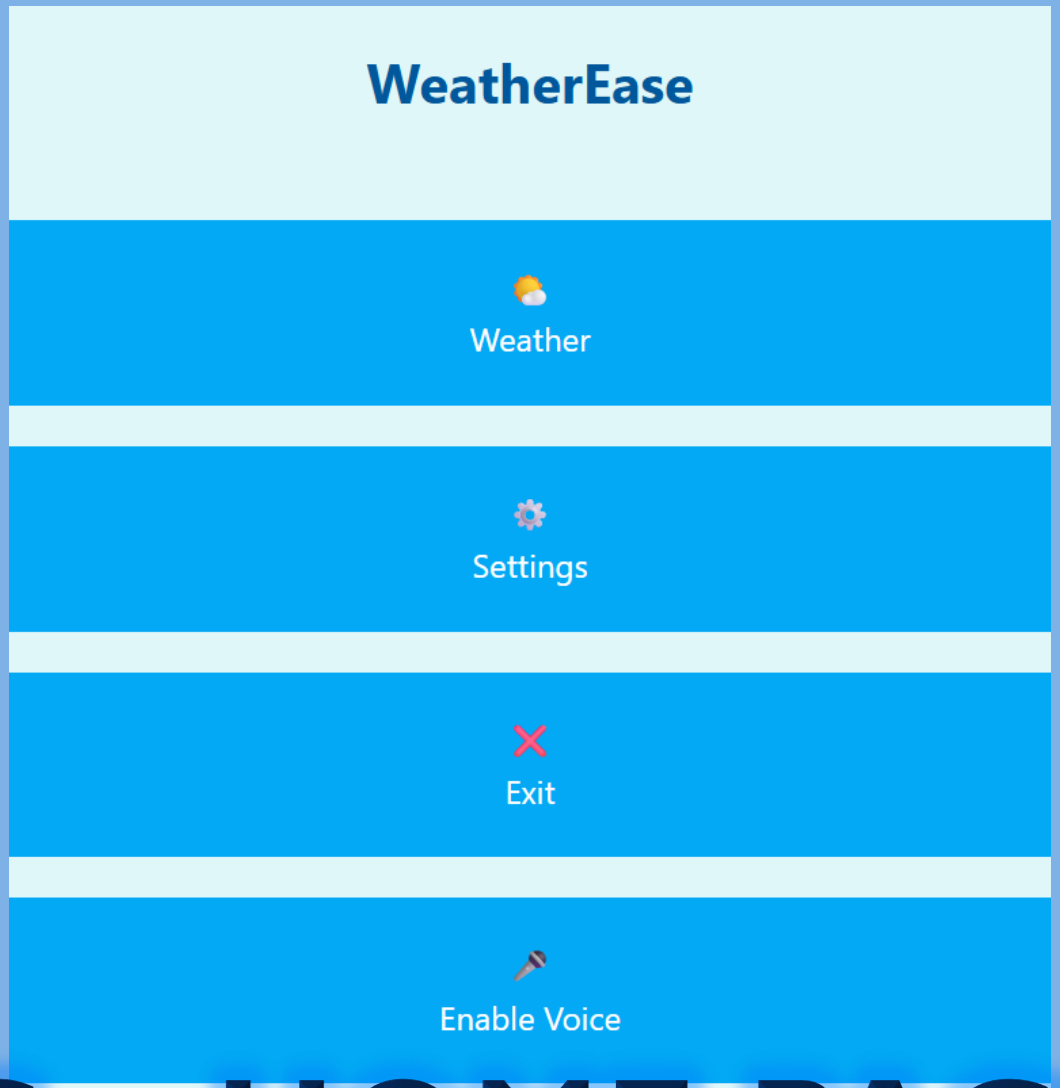
Already implementing API, learning how to debug, adding features we learned in previous courses or from research.

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.

FINAL DESIGN – HOME PAGE

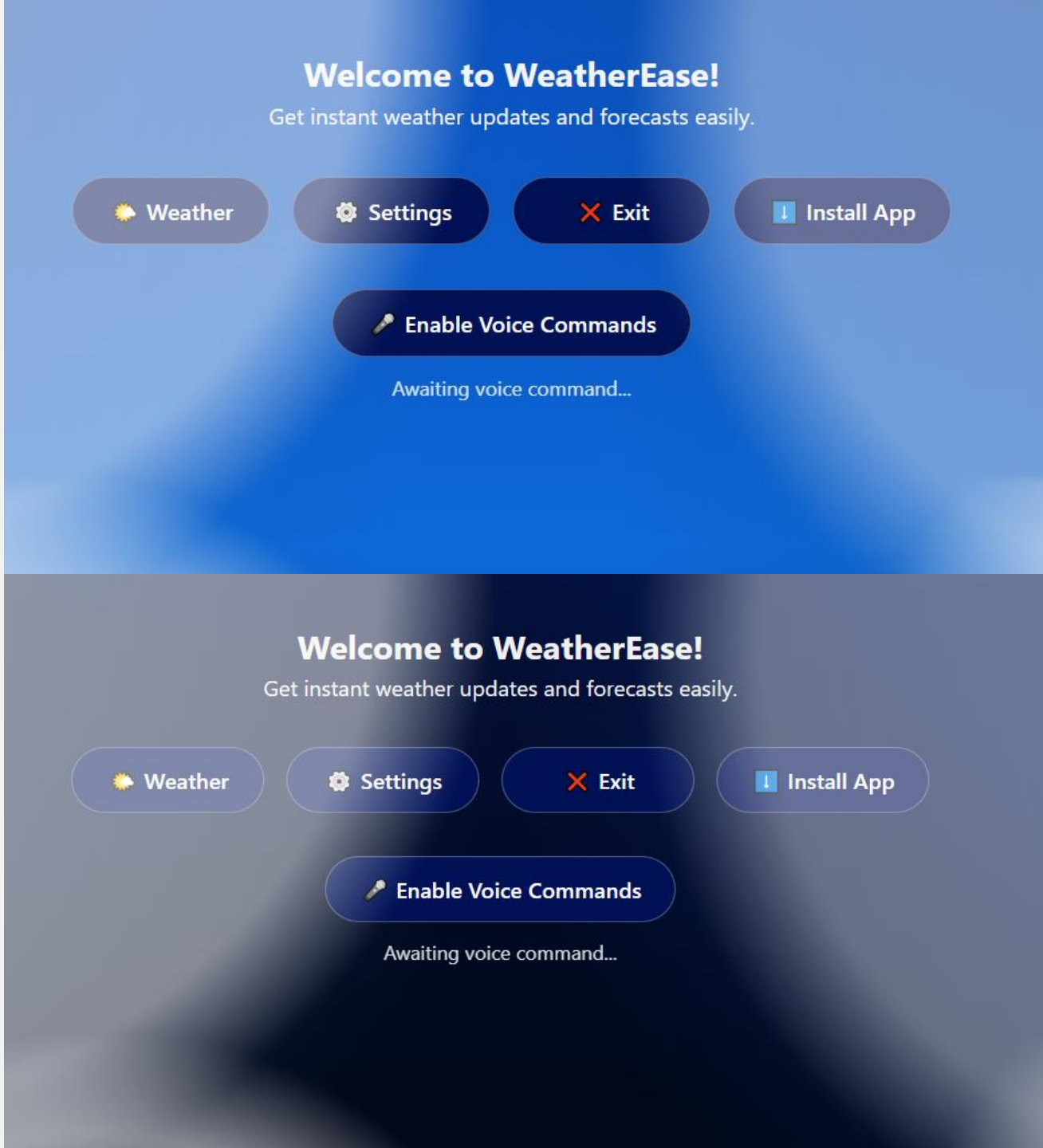
Our final home page design embraces a futuristic, modern, and minimal aesthetic that prioritizes both usability and accessibility.

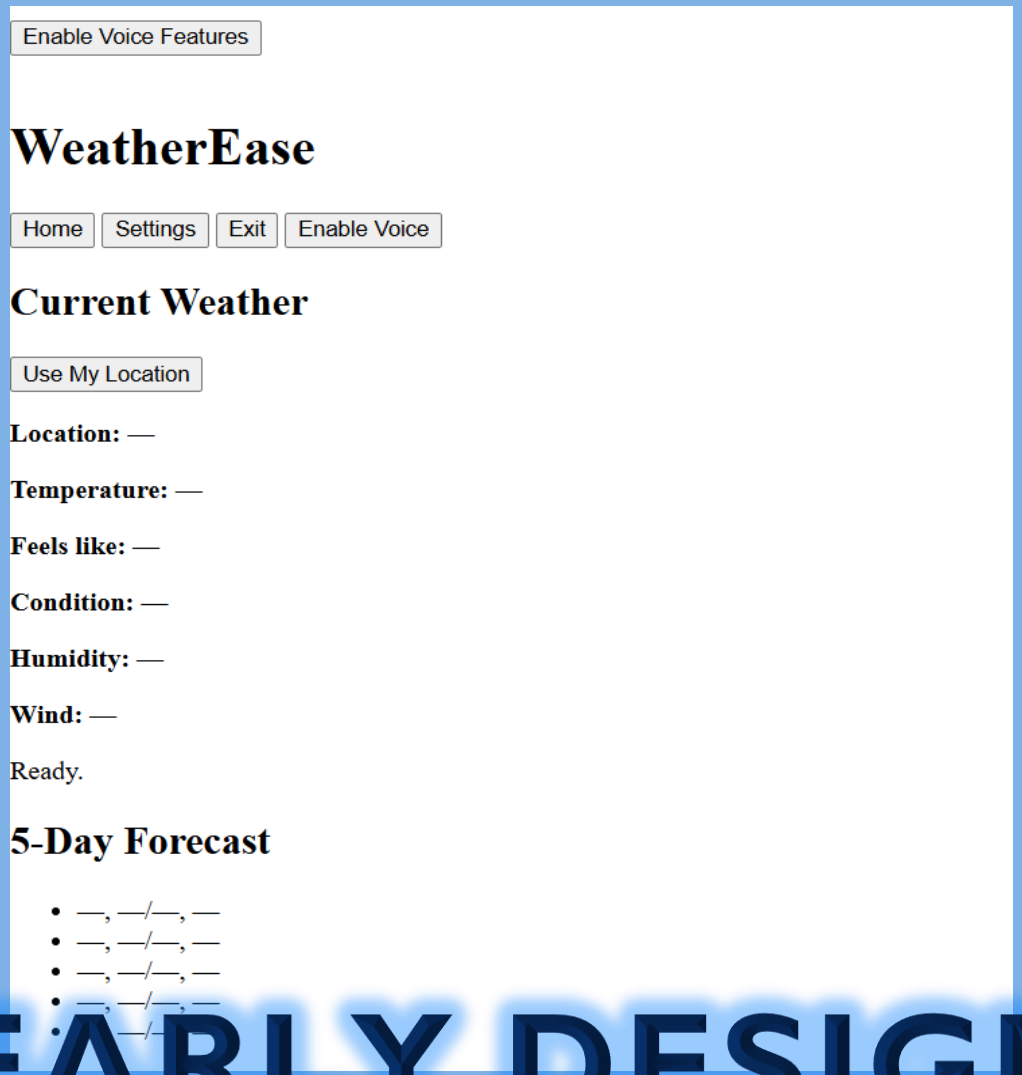
The layout is simple and intuitive, with large, clearly labeled buttons and high-contrast text to support users who rely on screen readers or voice commands.

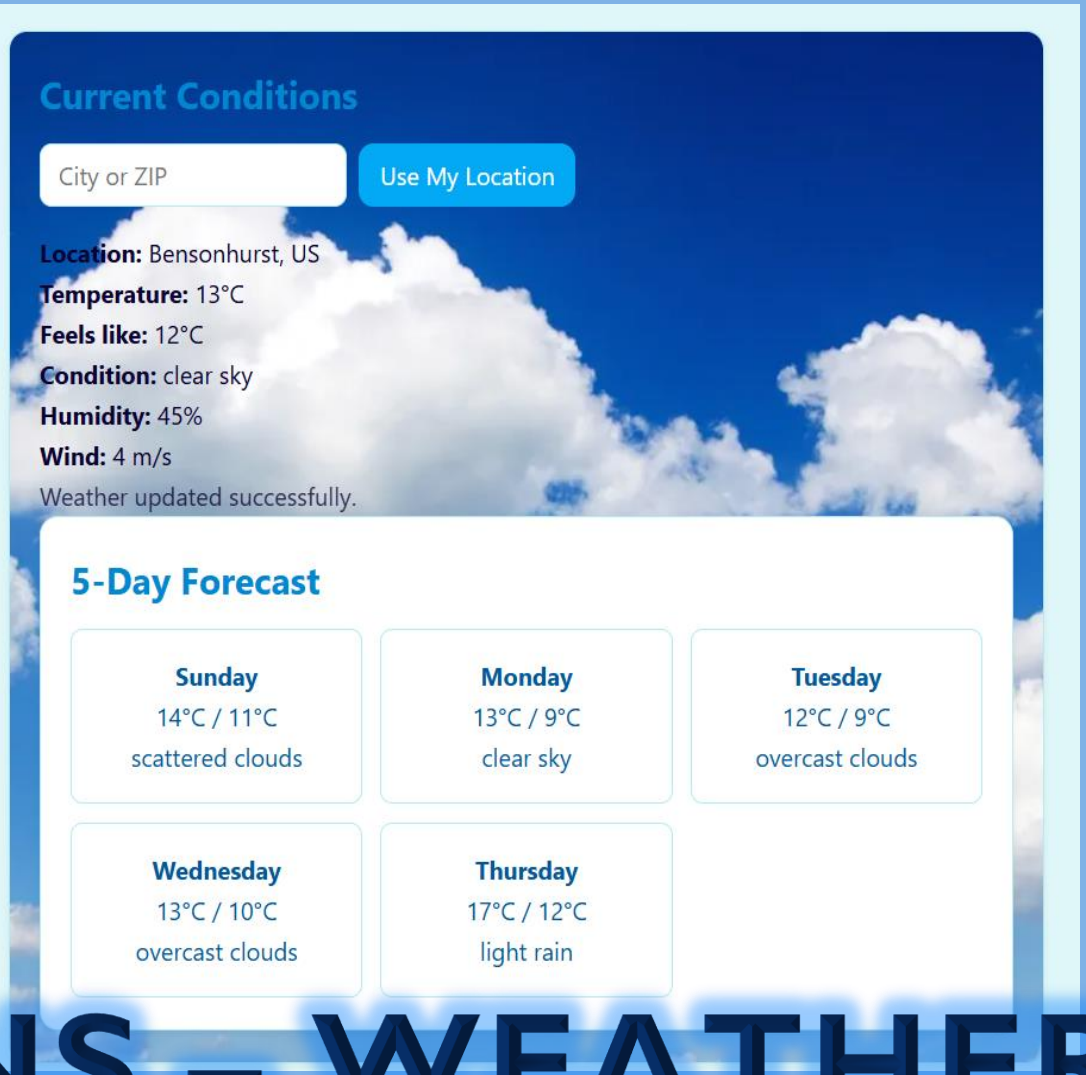
By emphasizing a clean visual hierarchy, users can quickly access current weather, forecasts, and settings without distraction.

The prominent voice command feature reinforces the hands-free experience at the core of WeatherEase, while subtle animations and smooth transitions give the interface a polished, modern feel.

Every design choice was guided by the principle of inclusive, user-focused interaction, ensuring that all users, regardless of visual ability, can navigate and engage with the app effortlessly.







EARLY DESIGNS – WEATHER

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

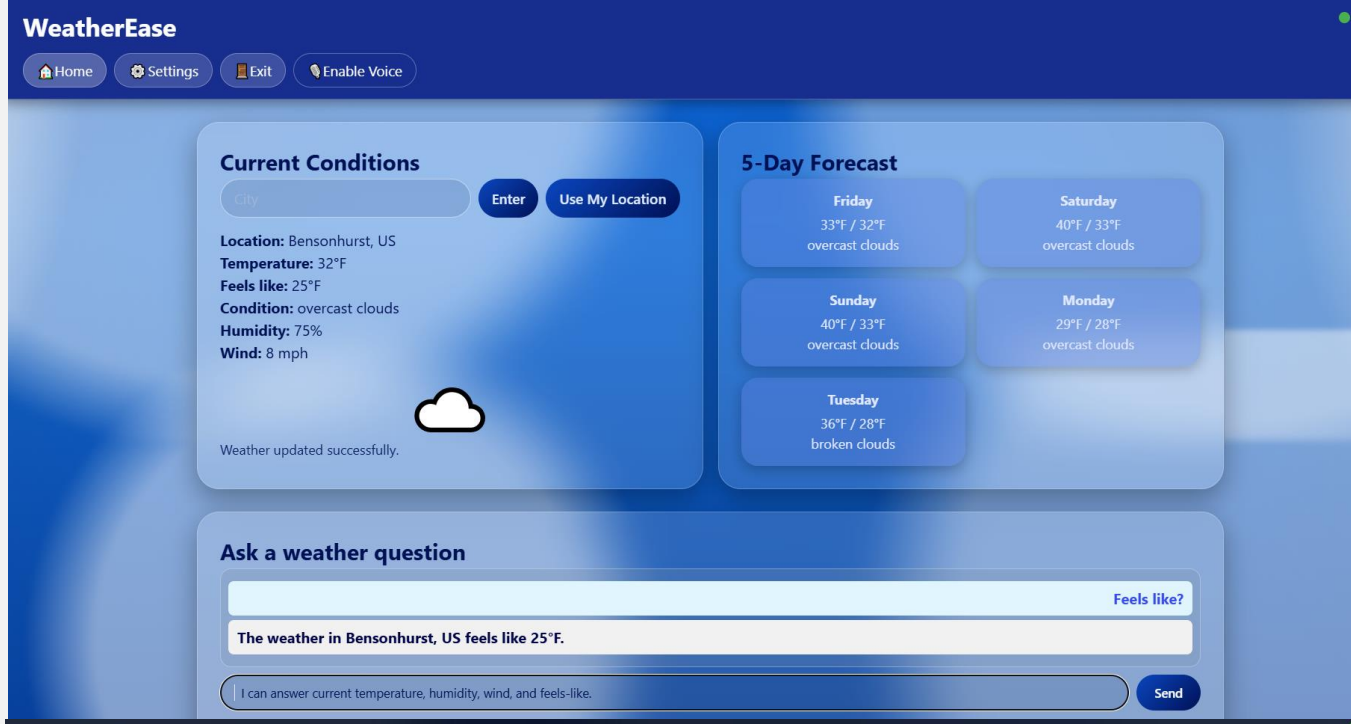
FINAL DESIGN – WEATHER

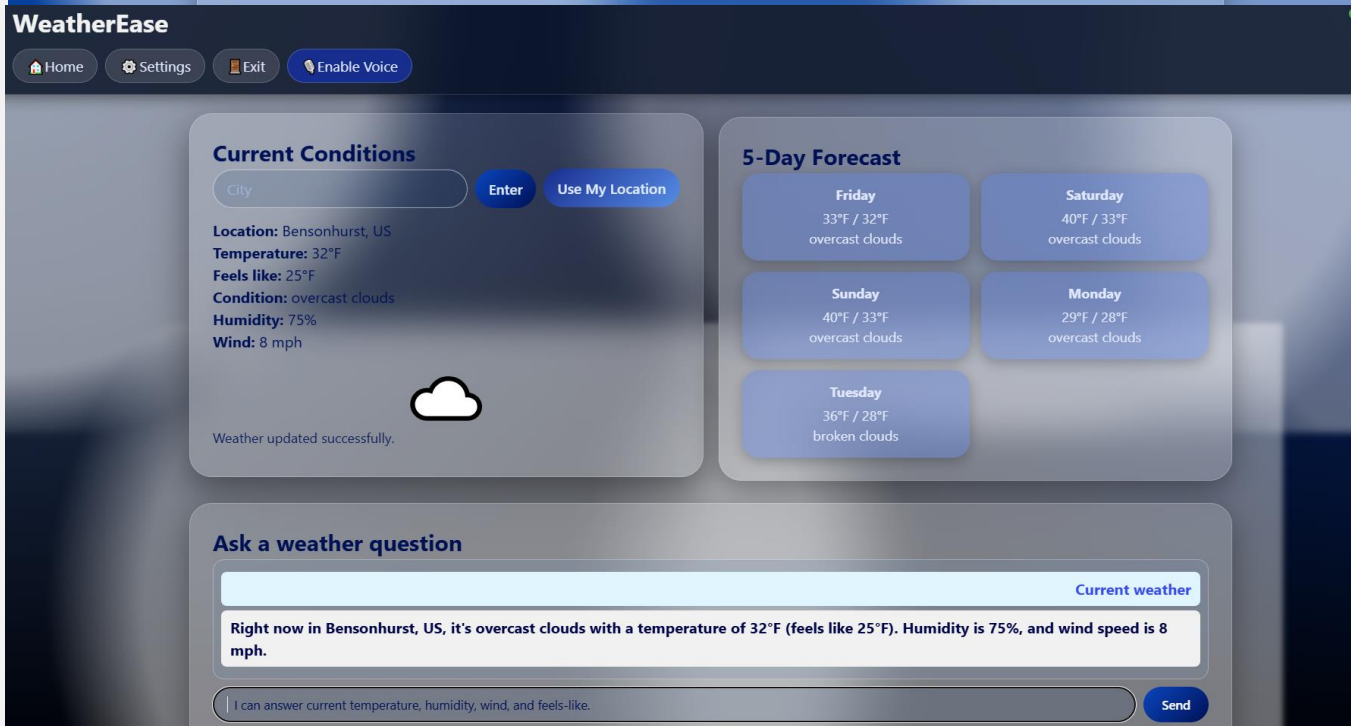
The final weather interface delivers information in a clean, modern, and highly readable layout, making it easy for all users to understand at a glance.

Weather data is clearly structured, with real-time updates and an online/offline indicator so users always know the app's connectivity status.

The addition of a chatbot-style interaction allows users to ask questions naturally and receive scripted responses, creating a conversational, hands-free experience.

Overall, the design balances simplicity, clarity, and accessibility, ensuring that even complex weather information is delivered in a way that is intuitive, inclusive, and visually appealing.



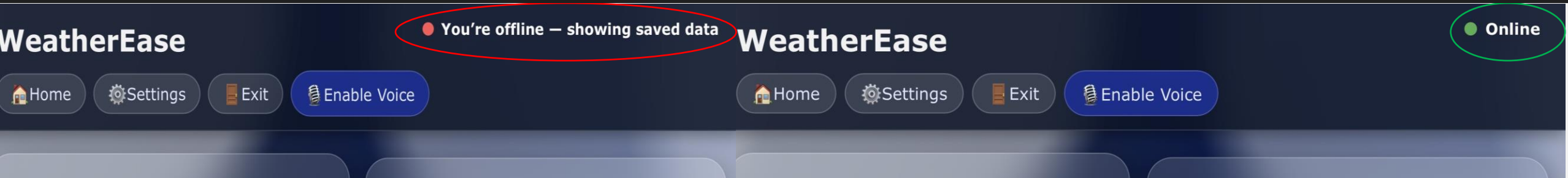


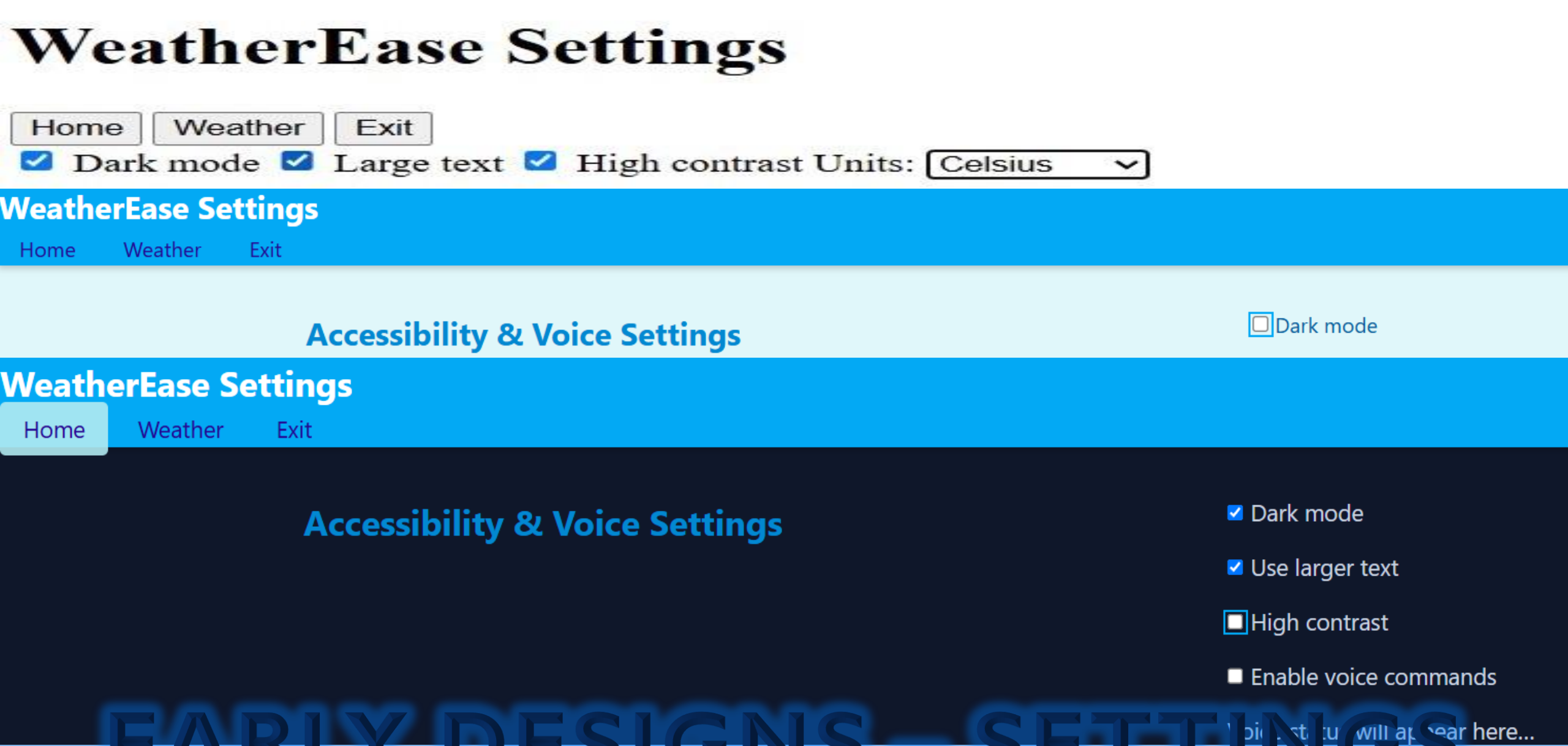
FINAL DESIGN – WEATHER

ONLINE/OFFLINE STATUS & CODE

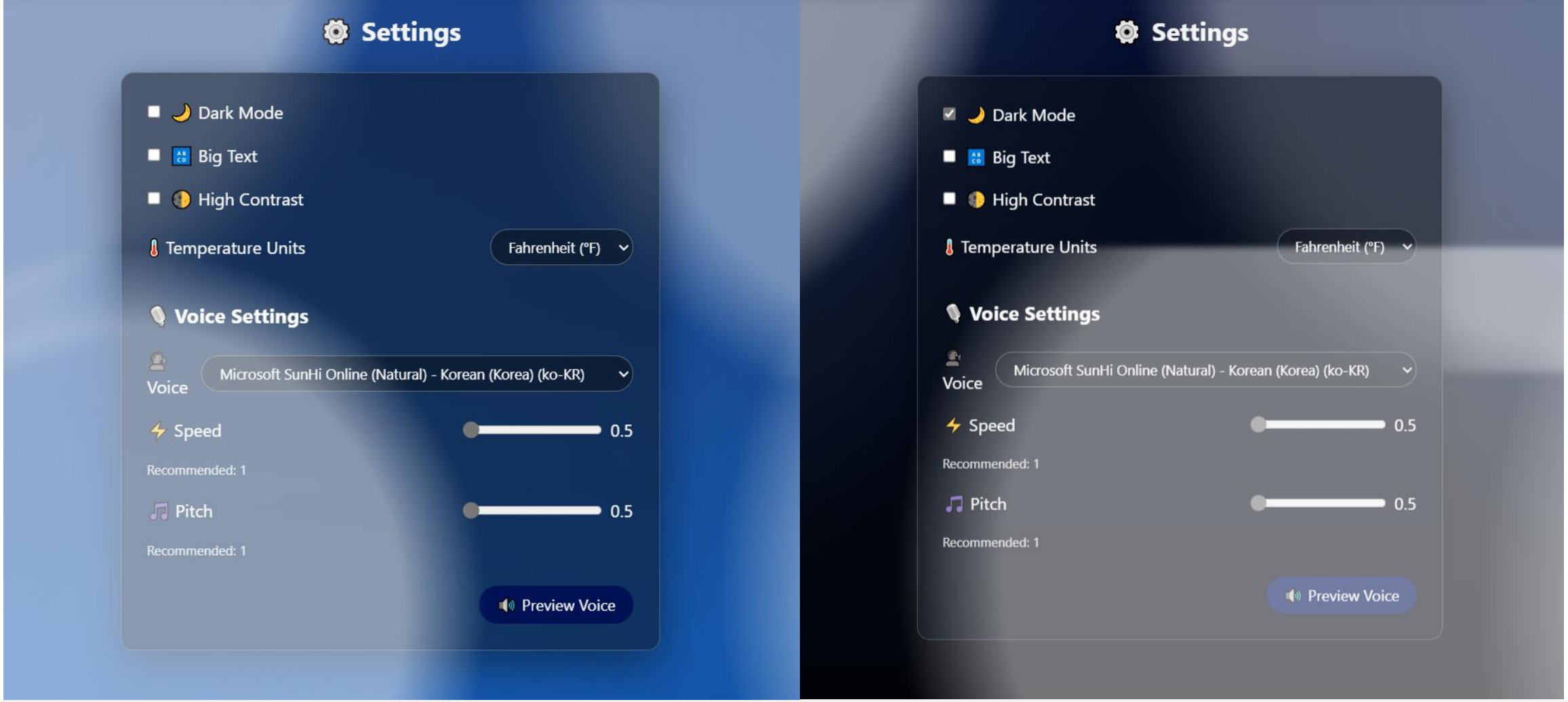
```
// ----------------------------
// Offline/online alerts (visual + voice)
// ----------------------------
window.addEventListener('offline', () => {
    const alertEl = document.getElementById('current-status') || document.getElementById('voice-status');
    if (alertEl) alertEl.textContent = '⚠️ You are offline. Showing cached data.';
    if (window.speechSynthesis) {
        const speech = new SpeechSynthesisUtterance('You are offline. Showing cached data.');
        window.speechSynthesis.speak(speech);
    }
});

window.addEventListener('online', () => {
    const alertEl = document.getElementById('current-status') || document.getElementById('voice-status');
    if (alertEl) alertEl.textContent = '✅ Back online.';
    if (window.speechSynthesis) {
        const speech = new SpeechSynthesisUtterance('You are back online.');
        window.speechSynthesis.speak(speech);
    }
});
```





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



FINAL DESIGN – SETTINGS

The Settings page is simple and user-friendly, giving users control over their experience.

Users can switch between temperature units, adjust voice options and speech speed, and use pitch gestures for speech scaling.

A preview feature lets users immediately hear changes, ensuring that all adjustments are accessible, intuitive, and tailored to individual needs.

TESTING CHECKLISTS

We performed two testing checklists, one for week of Nov 10-16 and one for the final week Dec 2-8.

This checklist is from Nov 10-16, where we have two tasks that are in progress – meaning needs more work done and refinement.

5. Chat / AI Assistant (Tajra)

Task	Status	Notes
User message submission works	☐ **Passed**	Users can submit any message.
AI reply appears in chat log	☐ **In progress**	There is no AI API yet, hence it will keep responding with a **Sorry, I could not answer that right now.**
Scroll auto-follows new messages	☐ **Passed**	Any new prompt and response will result in auto-follow
Chat formatting (user vs bot) correct	☐ **Passed**	Messages display correctly for user and bot, alignment and colors are fine.

6. Deployment & Offline Testing (Ehab)

Task	Status	Notes
Site deployed to GitHub Pages	☐ **Passed**	The prototype is live and accessible.
Offline testing: reload while offline → app shell loads	☐ **Passed**	The app shell loads correctly offline.
Offline caching: last weather data appears	☐ **Passed**	Cached weather data displays as expected.
Online status restored → alerts & TTS work	☐ **In progress**	The feature to appear online needs to be added, but TTS works correctly.

Additional Notes / Bugs

The AI Chat Bot was a quick, experimental idea that wasn't part of the original project plan.

It was something new and interesting we wanted to try out.

If time allows, we would like to explore it to its full potential so it works smoothly, but if not, that's perfectly fine since it wasn't part of our main project idea plan.

TESTING CHECKLISTS

We performed two testing checklists, one for week of Nov 10-16 and one for the final week Dec 2-8.

This checklist is from Dec 2-8, where previously two in progress task have become of passed status.

5. Chat Weather Assistant (Tajra)

Task	Status	Notes
User input submits correctly	☐ Passed	User input goes through without any issues. And if any, prompts ensure navigation.
Bot replies with structured weather sentences	☐ Passed	The bot gives weather replies in clear sentences.

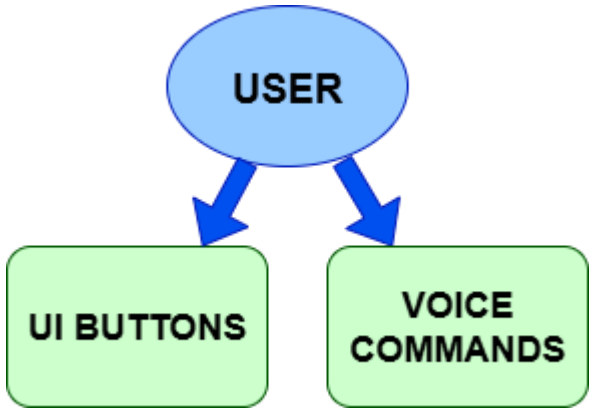
6. Deployment & Offline Testing (Ehab)

Task	Status	Notes
Site loads fully on GitHub Pages	☐ Passed	The site loads completely on GitHub Pages.
Offline reload loads app shell	☐ Passed	Reloading offline still shows the main app structure.
Cached weather data displays offline	☐ Passed	Weather data shows correctly even when offline
Online status detection + TTS alerts work	☐ Passed	Online status and voice alerts work properly.

Additional Notes / Bugs

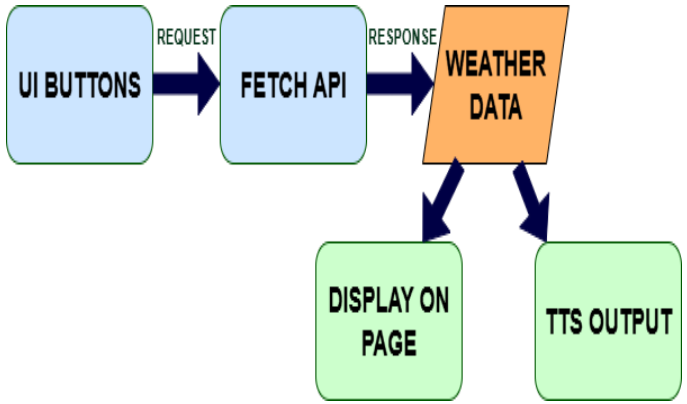
- AI Chat Bot replaced with a simplified weather-response assistant.
- Voice permissions no longer spam; recognition runs only when manually enabled.
- PWA install button now appears properly.

User Input Modalities



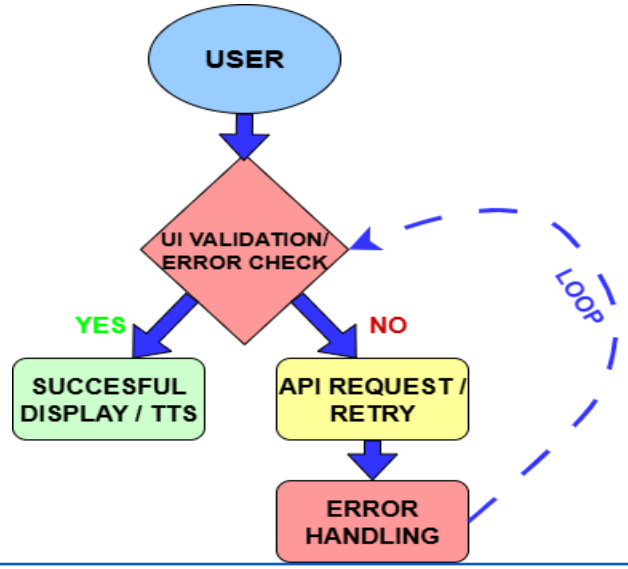
This diagram shows how the user interacts with the app, either by pressing buttons to get weather or listen, or by issuing voice commands for specific information.

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

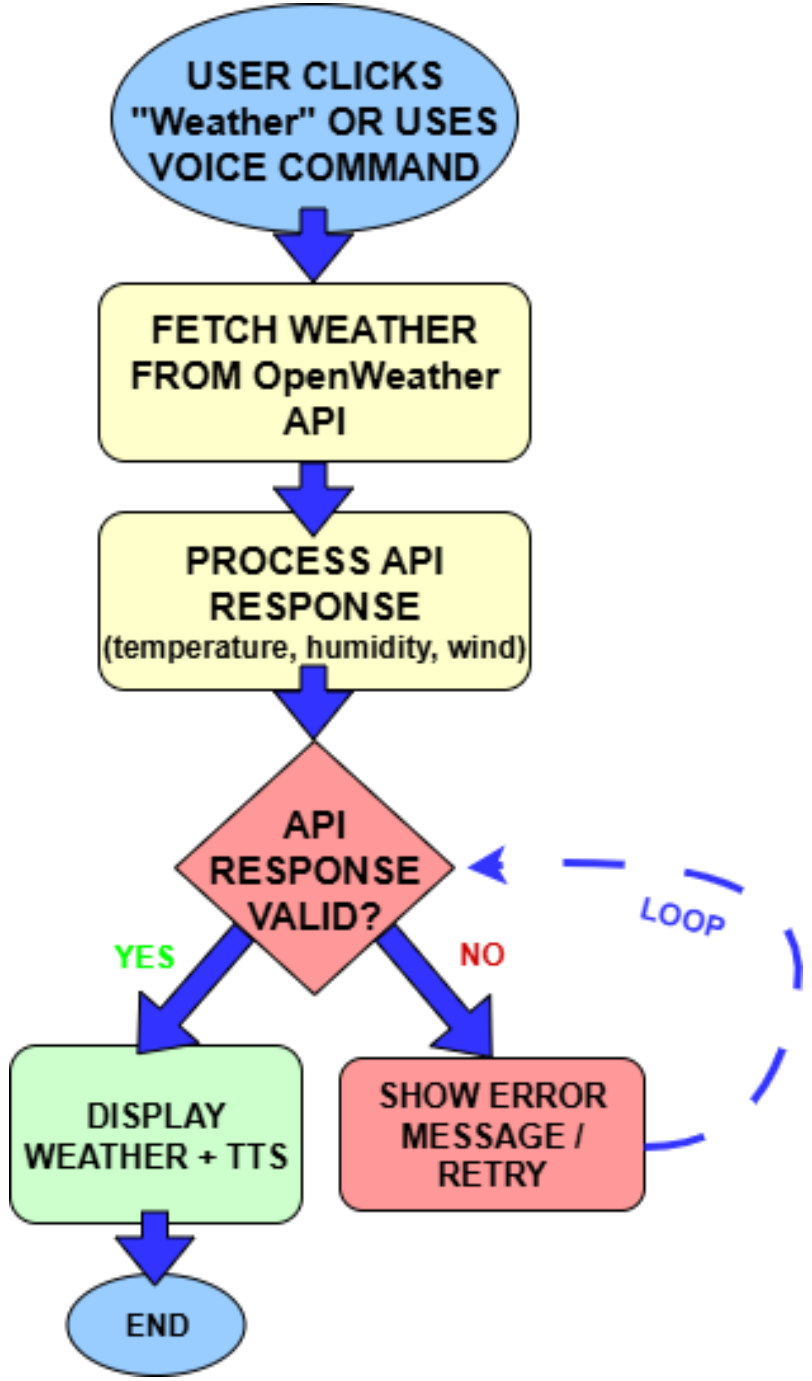
Error Handling Flow



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.

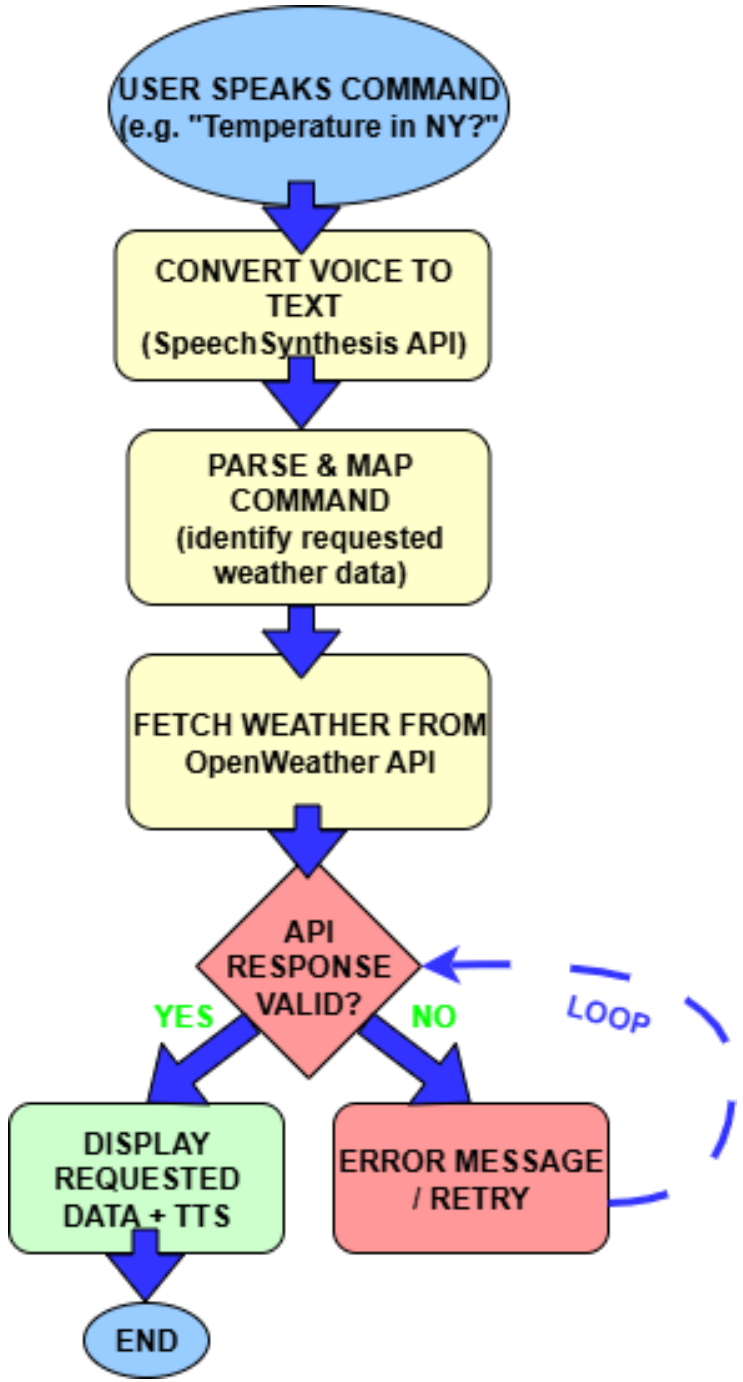
CASE 1: CHECK CURRENT WEATHER

- 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.



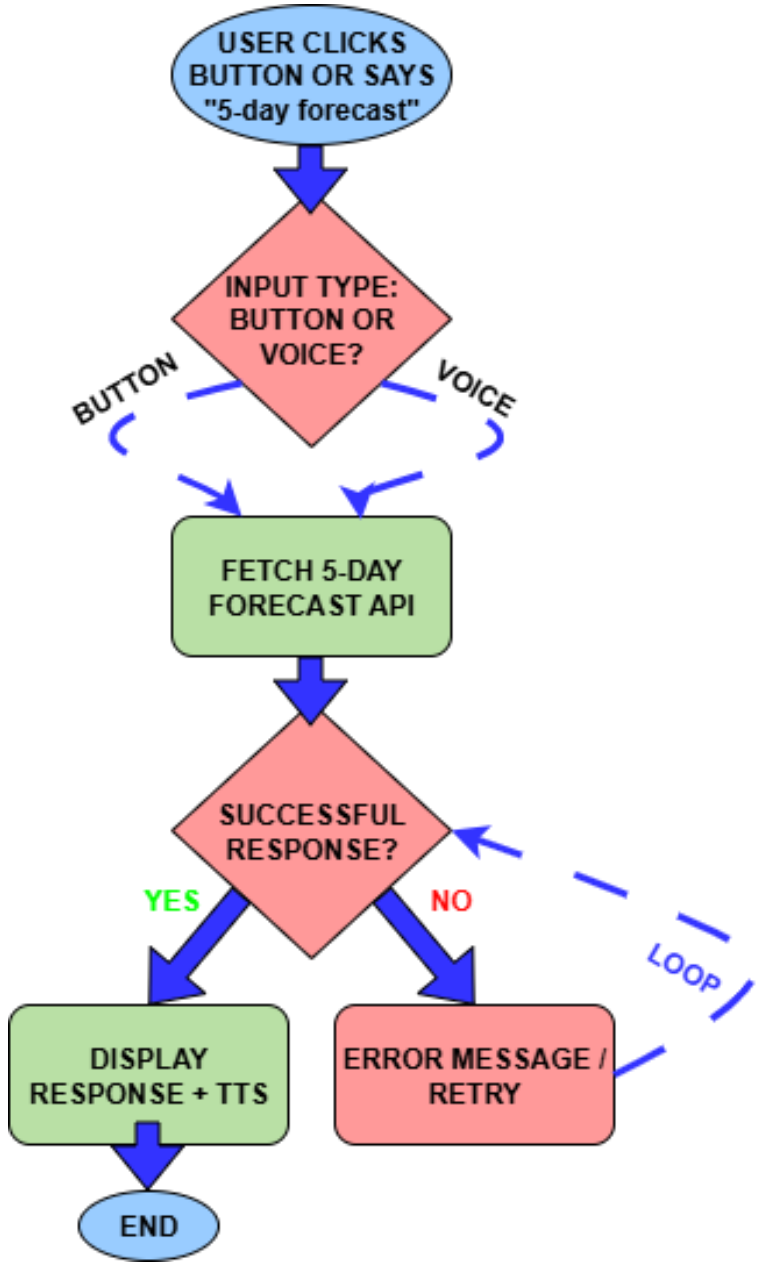
CASE II: VOICE QUERY FOR WEATHER DETAILS

- 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

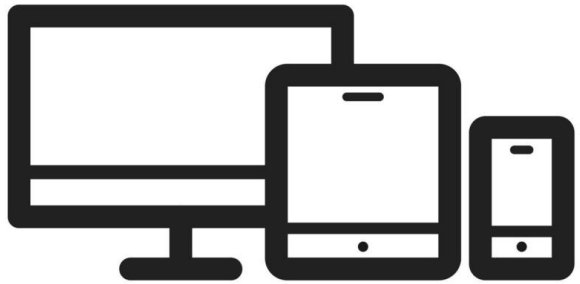
API
- OpenWeatherMap
- Web Speech API
- SpeechSynthesis API

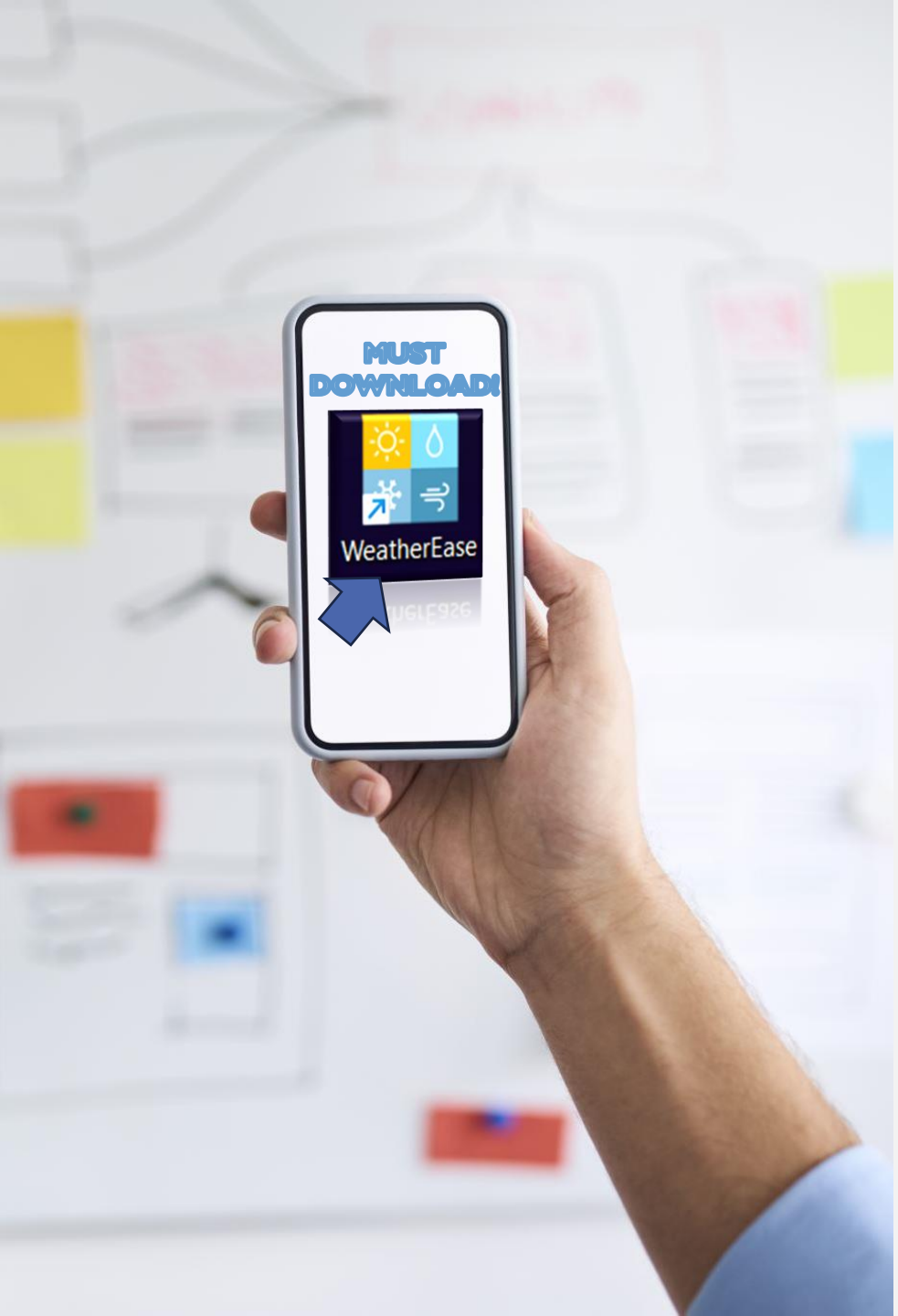
PWA COMPONENTS
- manifest.json (app metadata & install behavior)
- Service Worker (offline caching)

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/1IGvmm2wwOGerVoRkfISTbxjR2IwU7BI5_7Lgd7Z1JMg/edit?usp=sharing
- **WEBSITE:** WeatherEase