CS 307 | Software Engineering I
Group 12 | Project Backlog

ANDA: Application for Natural Disaster Avoidance

2/5/2025

Gleb Bereziuk, James Lamparski, Josh Rubow, Jinhoo Yoon

1. Problem Statement

Premise: As global temperatures rise and extreme weather becomes more frequent, the need to adapt to changing conditions grows. Our application leverages real-time IoT sensor feeds and meteorological data to monitor unsafe regions, blocked roads, and hazardous areas. This enables users to be rerouted to safer, more accessible paths. Unlike existing solutions–such as Indy Snow Force–that rely on limited, manually reported data from government agencies, our approach provides an affordable and reliable way to track weather conditions in remote areas in real time.

2. Background Information

Traveling is not always safe, and there's often a lack of reliable up-to-date information about road safety given that data comes exclusively from government agencies. With ANDA, our goal is to create a connected IoT network of devices that measure real-time data such as temperature and humidity in order to highlight high-risk and actively dangerous geographic regions and routes. Our total coverage would expand to wherever we could plant one of our IoT devices: from highly-populated areas that are prone to extreme weather conditions to more remote communities with insufficient weather reporting resources. We hope that anybody with access to the internet can freely access our application to plan out alternate routes that adapt to real-time weather conditions and allow users to safely plan their travels as well as contribute with manual reporting with the most up-to-date information.

As mentioned above, the Indy Snow Force website is one of hundreds of city-developed resources for "real-time" road conditions reporting. Similarly, there are tools for flood reporting, like GettingAroundIllinois by ILDOT. However, just like the Indy Snow Force website, it is only one of many independent resources. There are hundreds of small disconnected services developed by individual cities all over the country to serve the same purpose, and such fragmentation is very inconvenient to users, especially to those travelling between jurisdictions. By developing a single tool that all citizens could use in their daily routine, we are providing a resource that cities could integrate into. Additionally, the nature of exclusive government reporting on the current conditions is unreliable, as information on official sources is not always up-to-date. Letting users report current conditions on their own would make the tool more reliable. Lastly, given that not all remote municipalities could afford sophisticated systems, offering a tool that could monitor conditions at a relatively cheap cost would be appealing to remote communities. By creating a single tool that all users can use and agencies can report to, we will consolidate this fragmented sector.

3. Requirements (Backlog)

Functional Requirements

- As a User I would like to:
    - Register a new account so that I can access personal information
    - Save personal information & preferences so that I personalize my account
    - Log into web application so that I can access live weather and road conditions with my preferences
    - Browse securely as a guest so that I don't have to log in for quick access
    - Edit account settings so that I can save filters for rain, snow, sleet, update username, password, basic settings, etc.
    - Delete my account so that my information is no longer stored.
    - View current location so that I can access relevant weather and road information
    - Create a route to a new destination so that I know how to get to my destination.
    - Make condition filters so that I can access relevant weather data.
    - Create geographic filters to avoid areas and roads with bad weather conditions so that I could develop a safer route
    - Manually report geographic weather conditions on selected areas so that I can inform other people using the website of bad conditions.
    - Manually report road conditions in routes traveled with GPS so that I can inform other people of unsafe road conditions.
    - Report current snow plowing activity with GPS so up-do-date snow removal data is available (if time allows).
    - Register as a volunteer or a private entity so I could report my contributions to public services like snow removal (if time allows).
    - View current weather alerts from local officials so I am aware of current dangers.
    - Report bugs in the system so that the developers can fix them.
- As an Admin (e.g. local governing body) I would like to:
    - Register a new account so that I can save official information (devices, location, etc.);
    - Log into the web application so that I can view settings and official information.
    - View live IoT mesh network sensor data (temperature, humidity, precipitation…) so that I could manually analyze data.
    - Download sensor data in excel/csv format so that I can have direct access to info.
    - Filter live IoT mesh network by geographic region so that I can clearly view specific devices.
    - Manually adjust road and area safety, availability, and condition data so that the users have up-to-date information on closures/warnings by officials
    - Mark reports made by government vehicles (such as snow plows) so users could differentiate it from public reports (if time allows).

    - Mark certain users as public officials so their GPS data could be reflected on the website.
    - Push urgent updates so that the public has quick access to critical information
    - Remove inaccurate/malicious reports from my jurisdiction so that the website is not populated with inaccurate data.
    - Register new ESP32 devices so that they can report data .
    - Set up ESP32 devices to each other so that they can communicate sensor data.
    - Connect ESP32 *sentinel* device to an available network so that it can communicate all sensor data via the API.
    - View ESP32 mesh network online/offline availability so that I can analyze which devices might need maintenance.
    - View each ESP32 device's battery life so that I can plan ahead.
- As a System Admin I would like to:
    - Remove user accounts so that inactive users are no longer in the database
    - Remove admin accounts so that inactive officials are no longer in the database
    - Ban malicious users' IP addresses so that our website is not spammed with bad data
    - Push updates to the CI/CD pipeline so that keep the website up to date and minimize fix any bugs/errors (if time allows).

<u>Non-Functional Requirements</u>

Architecture & Performance

- We are going to use ESP32 devices with multiple sensors attached with a goal of 10ms communication time between individual devices.
    - Modularize the ESP32 container so that I can attach various sensors with ease (if time allows)
    - Setup signal boosters so that devices can communicate over greater distances (if time allows)
    - Setup the ESP32 device so that we can use cell network to report data (if time allows)
    - Implement a status light on each device so that we can tell if a device is too far from the current network (if time allows)
- Frontend and API will be built with React.js and Express.js.
    - Web page should have a load time of 7 seconds or less.
    - API should have a complete request time of 5 seconds or less.
- Perform efficient queries on SQL Server base within 5 seconds (may be updated later considering our table sizes)

Security

- Implement Google Sign-in API as an option for users.
- Implement user sign-in data encryption with at least a 256-bit encryption procedure
- Allow 3 password attempts for non-Google sign ins; force an email 2FA for such failed attempts.
- Encrypt location data in transit with TLS to secure sensitive user GPS information from potential misuse.
- Implement a secure connection protocol between ESP32 devices (if time allows).
- Encrypt all web traffic with HTTPS and API data with public/private key pairs.

Usability

- Ensure performance goals are met as specified above (e.g. load time, request time…).
- Implement accessibility features for users with special needs including but not limited to alt-text for images, text size adjustment, and audio bits. (if time allows)
- Implement an interface that's intuitive to everyday users.
    - Use Google Maps API for best compatibility and user familiarity.
    - Ensure the website is usable on mobile devices as well as laptops and tablets.
- Ensure available data is displayed accurately and updated in real time (<5 min delay).

Hosting & Development

- Ensure the system is available 24 hours a day, 7 days a week.
- Ensure the system is able to support at least 50 concurrent users (minimum viable product).

Scalability

- Ensure that the system does not exceed our target load and request times as more users register.
- Ensure that at least 5 ESP32 devices can be connected to one sentinel device.
- Ensure that the system works in any location by building a robust and modular device that can withstand a drop from 6 feet and temperatures in the range 0∘ to 100∘ F.