



Problem



NDCTrack

Connect. Detect. Prevent

The Neighbourhood

Jeremy Tubongbanua

Jerry Shum

Nehmat Farooq

Wesley Kyle De Guzman

NDC Tracker
National Disaster Canada Tracker

[Home](#) [Find Disasters](#) [Report a Disaster](#)

National Disaster Canada Tracker Information Portal

Welcome to the National Disaster Canada Tracker Portal. Our mission is to provide accurate and timely information about natural disasters affecting Canada by utilizing accurate information provided by the public.

Our Mission

- Allows for reporting of disasters and sending of warnings to other users based on geographical location
- Allows for new disasters to be added and modified by the community
- Tracking of disasters and displaying data on disasters in an easy to read method

Key Features

- Real-time Disaster Tracking and Monitoring
- Geographic-based Alert System
- Community-driven Reporting, Removing and Editing tools
- Interactive Disaster Maps

[Report a Disaster](#) [Disasters Near You](#)

How NDCTrack Works

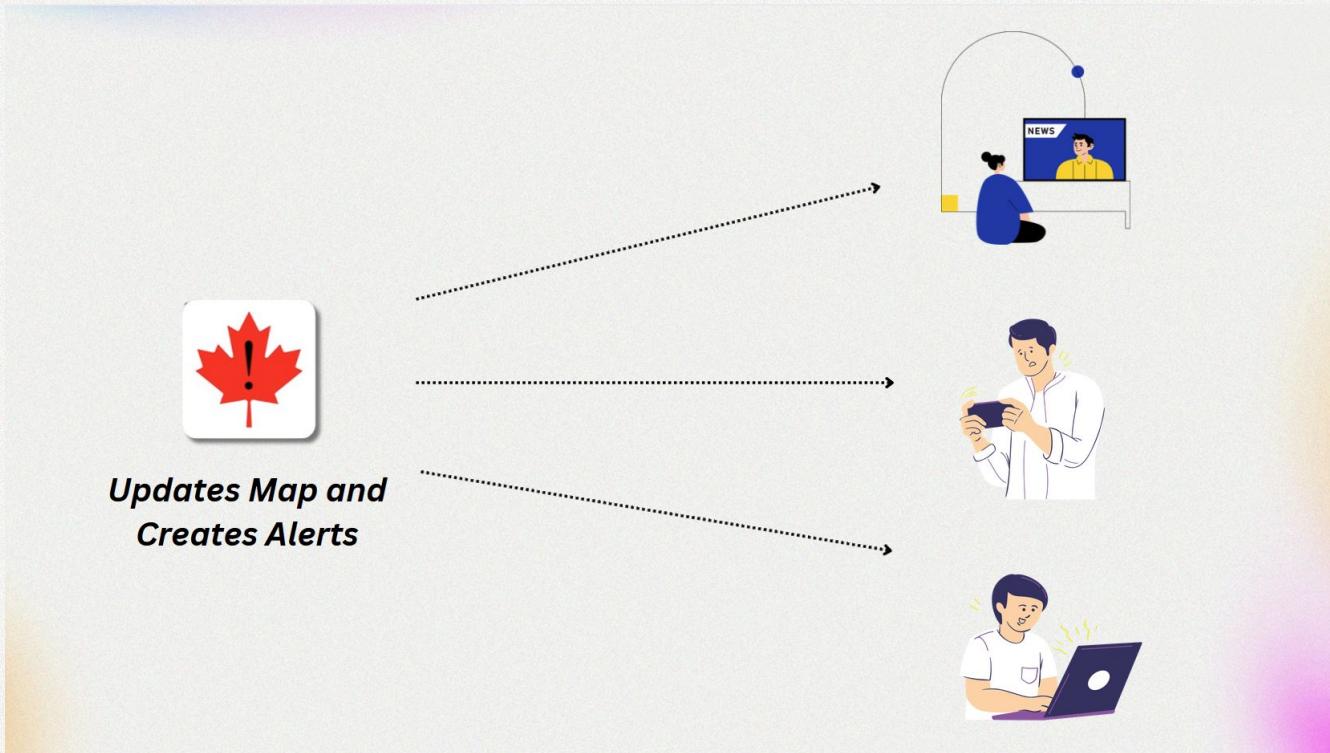


Bystander: I spot a disaster!



*Register Report on
NDCTrack*

How NDCtrack Works



Demo Time!

Front End Overview

4 Potential User Routes

Home Screen Page

 **NDC Tracker**
National Disaster Canada Tracker

Home Find Disasters Report a Disaster Admin

There are currently 4 disasters near you. Stay safe and take necessary precautions!
For more information, please click [here](#).

National Disaster Canada Tracker Information Portal

Welcome to the National Disaster Canada Tracker Portal. Our mission is to provide accurate and timely information about natural disasters affecting Canada by utilizing accurate information provided by the public.

Our Mission

- Allows for reporting of disasters and sending of warnings to other users based on geographical location
- Allows for new disasters to be added and modified by the community.
- Tracking of disasters and displaying data on disasters in an easy to read method.

Key Features

- Real-time Disaster Tracking and Monitoring
- Geographic-based Alert System
- Community-driven Reporting, Removing and Editing tools
- Interactive Disaster Maps

[Report a Disaster](#) [Disasters Near You](#)

7

Find Disasters Page



Home Find Disasters Report a Disaster Admin

There are currently 4 disasters near you. Stay safe and take necessary precautions!

For more information, please click [here](#).

Disaster Tracker

Enter an Address to Find Disasters Near You

Address

Radius

100

Submit

Found Disasters:

Earthquake

Major earthquake!!!

Disaster Type: natural

Radius: 50.0 km

Longitude: -79.118.2437

Latitude: 34.0522

Oil leak

Hazardous oil spill requiring containment

Disaster Type: manmade

Radius: 3.0 km

Longitude: -79.77373983912754

Latitude: 43.26691127633942

Tornado

Severe windstorm causing widespread damage

Disaster Type: natural

Radius: 50.0 km

Longitude: -79.87690025166192

Latitude: 43.251335599504124

Transportation accident

Major vehicle collision incident

Disaster Type: manmade

Radius: 10.0 km

Longitude: -79.91690987193221

Latitude: 43.15423471652471

Landslide

Ground movement causing terrain instability

Disaster Type: natural

Radius: 5.0 km

Longitude: -79.88004385453455

Latitude: 43.22583209403948

Food contamination crisis

Widespread food supply contamination

Disaster Type: biological

Radius: 40.0 km

Longitude: -80.08083395630251

Latitude: 43.23032686621058

Forest Fire

Rapidly spreading wildfire threatening residential areas

Disaster Type: natural

Radius: 25.0 km

Longitude: -80.00769185869119

Latitude: 43.258683067646025

Viral Outbreak

Local outbreak of infectious disease requiring containment

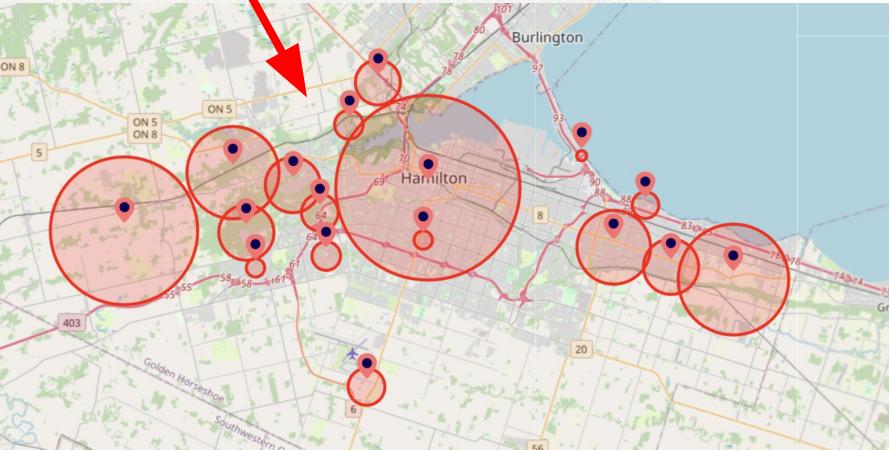
Disaster Type: biological

Radius: 15.0 km

Longitude: -79.96753215231814

Latitude: 43.252686069481346

Disasters have an area of effect



Report a Disaster Page



National Disaster Canada Tracker

[Home](#) [Find Disasters](#) [Report a Disaster](#) [Admin](#)

There are currently 4 disasters near you. Stay safe and take necessary precautions!
For more information, please click [here](#).

Report a disaster

Report a Disaster Near You

Address

1280 Main St W, Hamilton, ON L8S 4L8

[Get My Location](#)

Longitude

-79.9226783

Latitude

43.2609974

Disaster Type

Natural

Name

Earthquake

Description

123

Radius

100 km

Submit

Admin Page

 **NDC Tracker**
National Disaster Canada Tracker

Home Find Disasters Report a Disaster Admin

There are currently 5 disasters near you. Stay safe and take necessary precautions!
For more information, please click [here](#).

Admin Page

Use an Address to Retrieve disasters in That Area

Address: 1280 Main St W, Hamilton, ON L8S 4L8
Radius: 100

Submit

Disaster ID: 0

Disaster Name: Earthquake
Description: Major earthquake
Disaster Type: Natural
Latitude: 34.0522
Radius: 50.0
Longitude: -118.2437

Edit Delete

Disaster ID: 2

Disaster Name: Tornado
Description: Severe windstorm causing widespread damage
Disaster Type: Natural
Radius: 50.0
Latitude: 43.251335599504124
Longitude: -79.87690025166192

Edit Delete

Disaster ID: 3

Disaster Name: Transportation accident
Description: Major vehicle collision incident
Disaster Type: Manmade
Radius: 10.0
Latitude: 43.15423471652471
Longitude: -79.91809897193221

Edit Delete

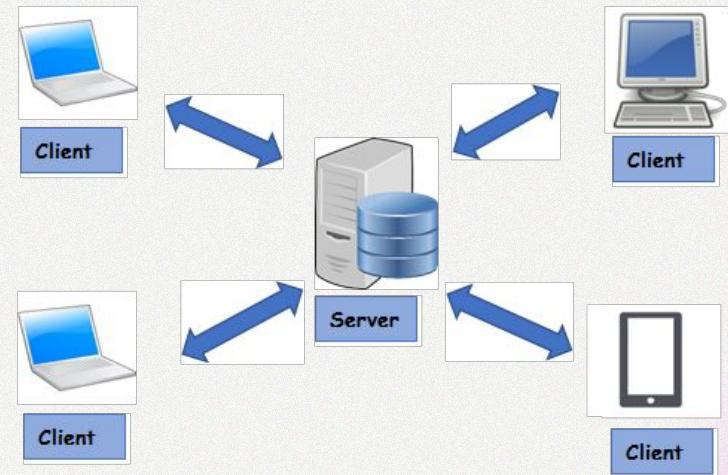
Backend Overview

How We Achieved a
Scalable Distributed Architecture

What We DIDN'T do

Centralized Client/Server Architecture

- One server cannot keep up with large load
- Not horizontally scalable
- Inefficient in handling large volumes
- Not extendable and not adaptable

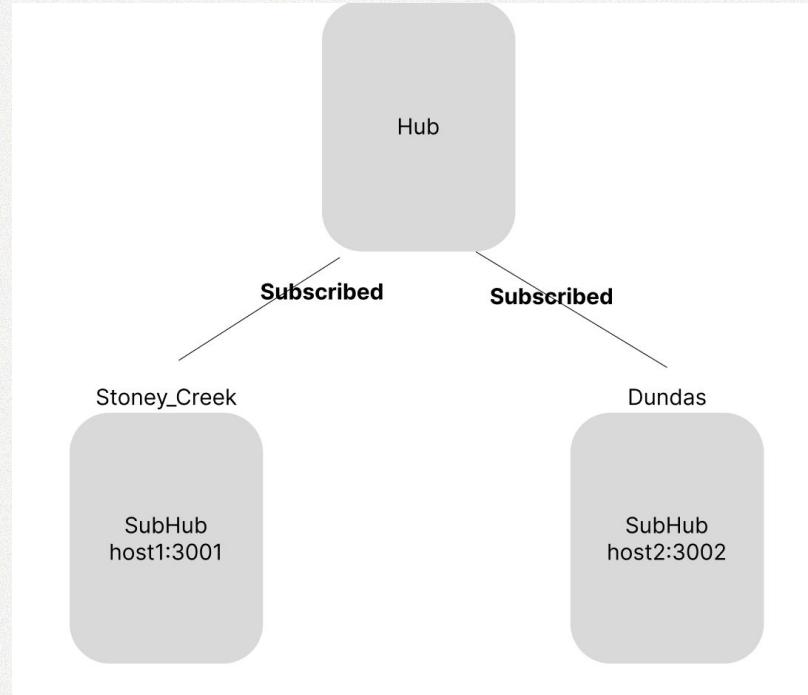


Hubs and SubHubs

Hub - Primary Hub, a server that SubHubs can subscribe to, like DNS

SubHub - A decentralized microservice that local users can view and report disasters to.

- Disasters that happen locally will only cause traffic to the SubHub of that region
- Distributed architecture scales horizontally



hub.py & subhub.py

Dynamic Input and Flexible Command Line Tools

```
+ backend git:(main) ✘ python3 hub.py -h  
usage: Hub [-h] [--host HOST] [--port PORT]
```

Run a hub instance that users can report to

options:
-h, --help show this help message and exit
--host HOST The host to run the hub on
--port PORT The port to run the hub on

```
+ backend git:(main) ✘ python3 subhub.py -h  
usage: Sub_Hub [-h] [--hub-host HUB_HOST] [--hub-port HUB_PORT] [--ip IP] [--host HOST] [--por  
[--existing-disasters EXISTING_DISASTERS]
```

Run a sub hub instance that users can report to

options:
-h, --help show this help message and exit
--hub-host HUB_HOST The host of the primary hub
--hub-port HUB_PORT The port of the primary hub
--ip IP The ip of the sub hub that can be accessible and seen by the primary h
--host HOST The host to run the sub hub Flask API on
--port PORT The port to run the sub hub Flask on
--id ID The id of the sub hub
--name NAME The name of the sub hub
--longitude LONGITUDE The longitude of the sub hub
--latitude LATITUDE The latitude of the sub hub
--radius_km RADIUS_KM The radius of the sub hub in kilometers
--existing-disasters EXISTING_DISASTERS The path to a CSV file containing existing disasters

hub.py - primary hub

Run an API that:

1. Holds records of sub hubs
2. Registers new sub hubs that subscribe to this hub
3. Refers callers to sub hubs within their location & radius

subhub.py - secondary hub

Run a secondary API that lets users:

1. Add, edit, and remove disasters
2. Search for disasters in their area
3. Persist disasters in CSVs

Distributed Architecture

Walkthrough

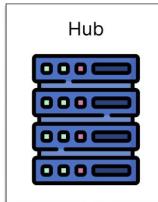
1) Start the hub using `bootcamp-hub.py`

```
python3 hub.py  
--host 0.0.0.0  
--port 3000
```

Distributed Architecture

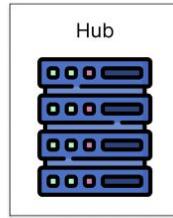
2) A Hub is spawned, now SubHubs can subscribe

```
python3 hub.py  
--host 0.0.0.0  
--port 3000
```



Distributed Architecture

3) SubHubs can spawn using **python3 subhub.py <args>**



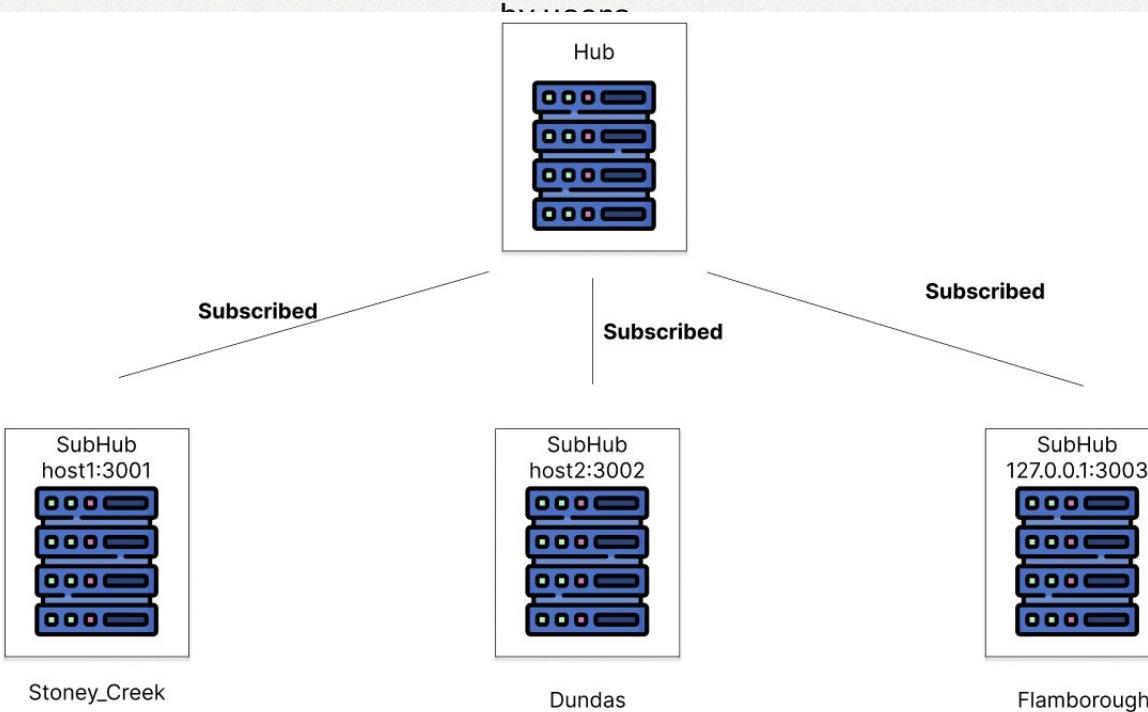
```
python3 subhub.py
--hub-host 127.0.0.1
--hub-port 3000
--ip 40.233.92.183
--host 0.0.0.0
--port 3001
--id 2
--name Dundas
--longitude 43.26691127633942
--latitude -79.77373983912754
--radius_km 3
--existing-disasters ./Dundas/Dundas.csv
```

```
python3 subhub.py
--hub-host 127.0.0.1
--hub-port 3000
--ip 40.233.92.183
--host 0.0.0.0
--port 3002
--id 3
--name Ancaster
--longitude 43.251335599504124
--latitude -79.87690025166192
--radius_km 50
--existing-disasters ./Ancaster/Ancaster.csv
```

```
python3 subhub.py
--hub-host 127.0.0.1
--hub-port 3000
--ip 40.233.92.183
--host 0.0.0.0
--port 3003
--id 4
--name Stoney_Creek
--longitude 43.15423471652471
--latitude -79.91809897193221
--radius_km 10
--existing-disasters ./Stoney_Creek/Stoney_Creek.csv'
```

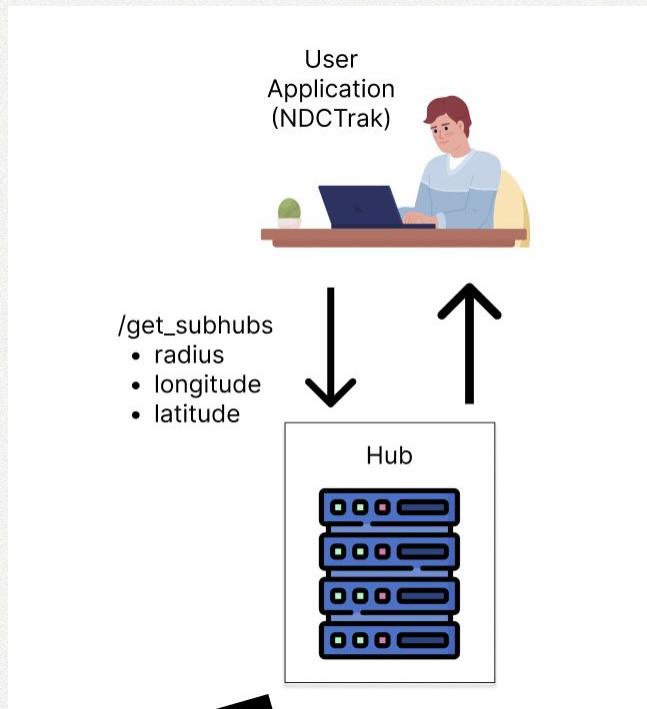
Distributed Architecture

4) Now SubHubs are subscribed to the Hub and discoverable



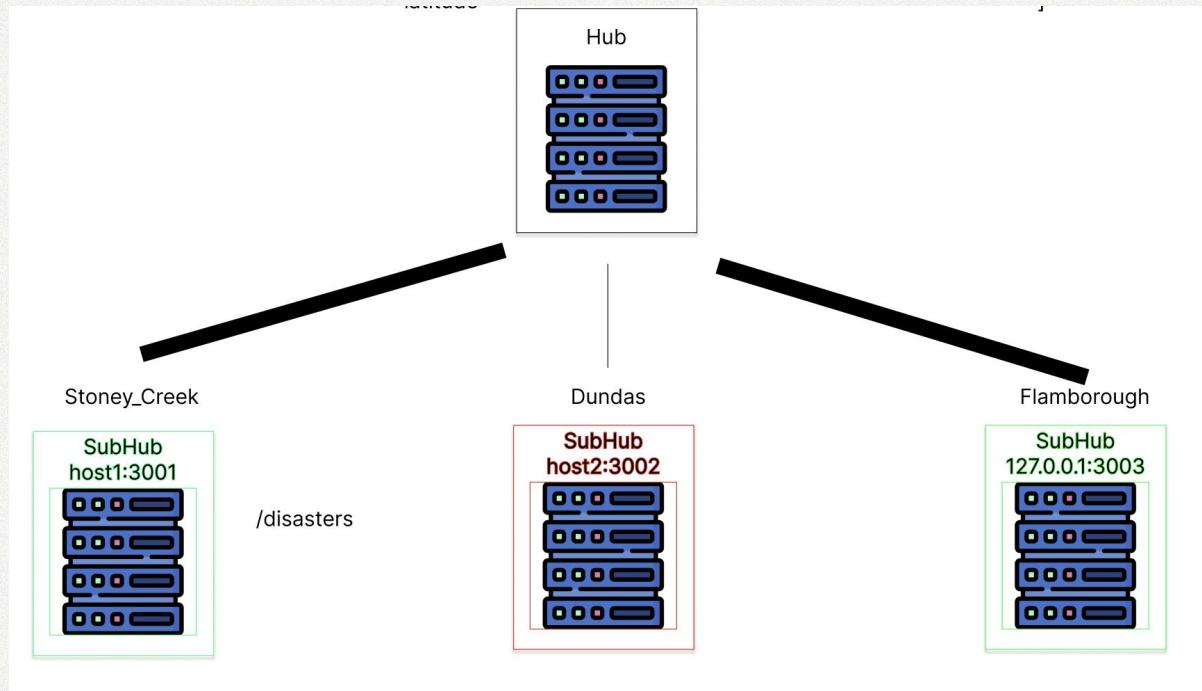
Distributed Architecture

5) A User wants to find disasters near them



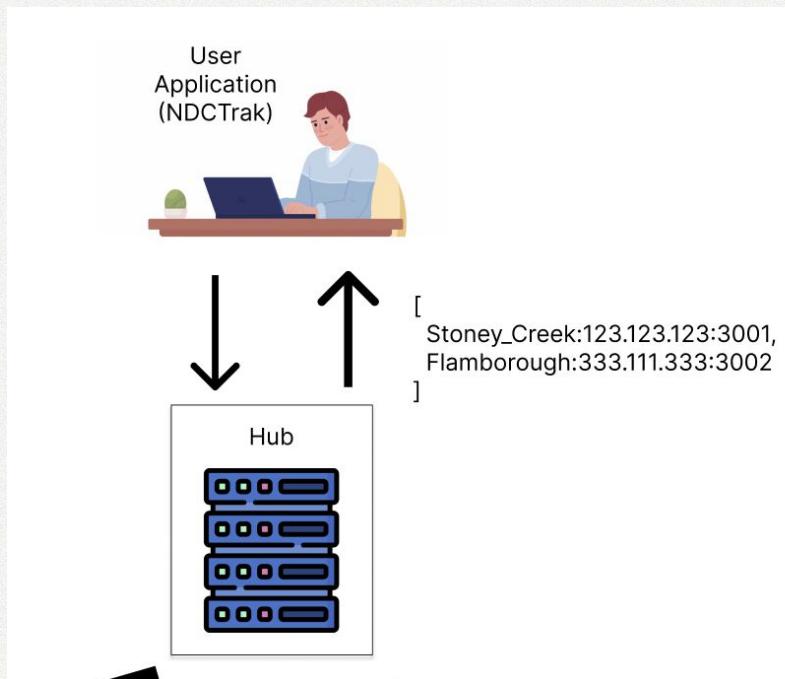
Distributed Architecture

6) Hub finds the SubHubs that are within their radius



Distributed Architecture

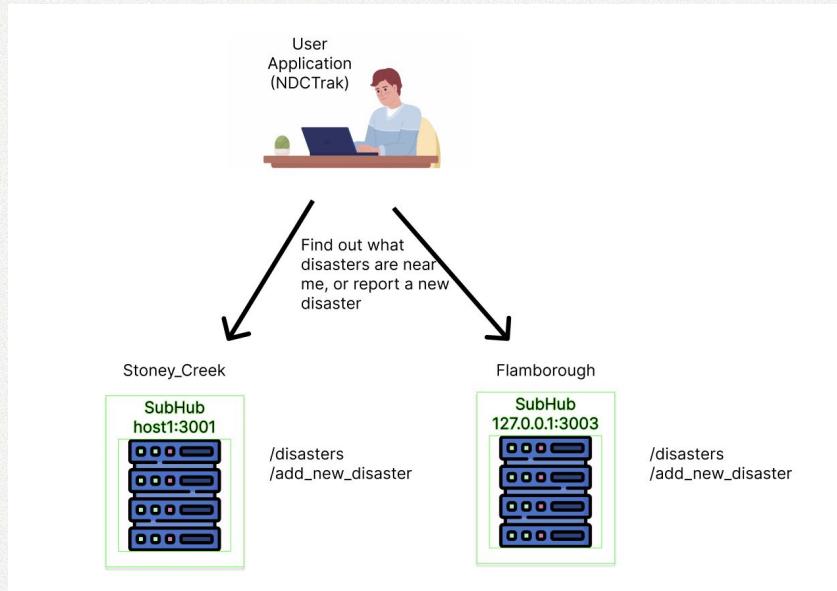
- 7) User Application finds the Host and Ports of how to reach the SubHubs across the network (host:port)

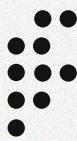


Distributed Architecture

8) User Application can now DIRECTLY talk to SubHub microservices

1. Report Disasters
2. View Disasters in the Area





Backend Extensibility and Adaptability

```

class DisasterNatural(Disaster):
    def __init__(self, disaster_id, disaster_type, name, description, longitude, latitude, radius_km):
        super().__init__(disaster_id, disaster_type, name, description, longitude, latitude, radius_km)
        self.severity = None
        self.affected_species = None

    def set_severity(self, severity):
        self.severity = severity

    def set_affected_species(self, affected_species):
        self.affected_species = affected_species


class DisasterBiological(Disaster):
    def __init__(self, disaster_id, disaster_type, name, description, longitude, latitude, radius_km):
        super().__init__(disaster_id, disaster_type, name, description, longitude, latitude, radius_km)
        self.pathogen = None
        self.transmission_rate = None

    def set_pathogen(self, pathogen):
        self.pathogen = pathogen

    def set_transmission_rate(self, transmission_rate):
        self.transmission_rate = transmission_rate


class DisasterManMade(Disaster):
    def __init__(self, disaster_id, disaster_type, name, description, longitude, latitude, radius_km):
        super().__init__(disaster_id, disaster_type, name, description, longitude, latitude, radius_km)
        self.cause = None
        self.economic_impact = None

    def set_cause(self, cause):
        self.cause = cause

class DisasterFactory():
    def create_disaster(self, disaster_id, disaster_type, name, description, longitude, latitude, radius_km):
        if disaster_type == 'natural':
            return DisasterNatural(disaster_id, disaster_type, name, description, longitude, latitude, radius_km)
        elif disaster_type == 'biological':
            return DisasterBiological(disaster_id, disaster_type, name, description, longitude, latitude, radius_km)
        elif disaster_type == 'manmade':
            return DisasterManMade(disaster_id, disaster_type, name, description, longitude, latitude, radius_km)
        else:
            raise ValueError(f"Invalid disaster type: {disaster_type}. Valid types are: {valid_disaster_types}")

```

Data Model

To enhance extensibility, OOP and Factory Design principles were used:

1 Abstract Class - Disaster

3 Subclasses - DisasterNatural, DisasterBiological, DisasterManMade

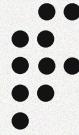
1 Factory Class - DisasterFactory

```

class Disaster():
    def __init__(self, disaster_id, disaster_type, name, description, longitude, latitude, radius_km):
        self.disaster_id = disaster_id
        self.disaster_type = disaster_type
        self.name = name
        self.description = description
        self.longitude = longitude
        self.latitude = latitude
        self.radius_km = radius_km

    def to_dict(self):
        return {
            'disaster_id': self.disaster_id,
            'disaster_type': self.disaster_type,
            'name': self.name,
            'description': self.description,
            'longitude': self.longitude,
            'latitude': self.latitude,
            'radius_km': self.radius_km
        }

```



Bash Script + TMUX

```
backend > up.sh
1  #!/bin/bash
2
3  # start up tmux sessions with these commands
4
5  # python3 hub.py --host 0.0.0.0 --port 3000
6
7  # python3 subhub.py --hub-host 127.0.0.1 --hub-port 3000 --ip 40.233.92.183 --host 0.0.0.0 --port 3001 --id 0 --name Hamiton --longitude 43.26691127633942
--latitude -79.77373983912754 --radius_km 1183.31
8
9  # python3 subhub.py --hub-host 127.0.0.1 --hub-port 3000 --ip 40.233.92.183 --host 0.0.0.0 --port 3002 --id 1 --name Burlington --longitude 43.3255 --latitude
79.7990 --radius_km 1183.31
10
11 tmux kill-ses -t hub
12 tmux kill-ses -t subhub0
13 tmux kill-ses -t subhub1
14 tmux kill-ses -t subhub2
15 tmux kill-ses -t subhub3
16 tmux kill-ses -t subhub4
17
18 tmux new-session -d -s hub 'python3 hub.py --host 0.0.0.0 --port 3000'
19
20 sleep 1
21
22 tmux new-session -d -s subhub0 'python3 subhub.py --hub-host 127.0.0.1 --hub-port 3000 --ip 40.233.92.183 --host 0.0.0.0 --port 3001 --id 2 --name Dundas
--longitude 43.26691127633942 --latitude -79.77373983912754 --radius_km 3 --existing-disasters ./Dundas/Dundas.csv'
23
24 tmux new-session -d -s subhub1 'python3 subhub.py --hub-host 127.0.0.1 --hub-port 3000 --ip 40.233.92.183 --host 0.0.0.0 --port 3002 --id 3 --name Ancaster
--longitude 43.251335599504124 --latitude -79.87690025166192 --radius_km 50 --existing-disasters ./Ancaster/Ancaster.csv'
25
26 tmux new-session -d -s subhub2 'python3 subhub.py --hub-host 127.0.0.1 --hub-port 3000 --ip 40.233.92.183 --host 0.0.0.0 --port 3003 --id 4 --name Stoney_Creek
--longitude 43.15423471652471 --latitude -79.91809897193221 --radius_km 10 --existing-disasters ./Stoney_Creek/Stoney_Creek.csv'
27
28 tmux new-session -d -s subhub3 'python3 subhub.py --hub-host 127.0.0.1 --hub-port 3000 --ip 40.233.92.183 --host 0.0.0.0 --port 3004 --id 5 --name
Hamilton_Mountain --longitude 43.22583209403948 --latitude -79.88004385453455 --radius_km 5 --existing-disasters ./Hamilton_Mountain/Hamilton_Mountain.csv'
29
30 tmux new-session -d -s subhub4 'python3 subhub.py --hub-host 127.0.0.1 --hub-port 3000 --ip 40.233.92.183 --host 0.0.0.0 --port 3005 --id 6 --name Flamborough
--longitude 43.23032686621058 --latitude -80.08083395630251 --radius_km 40 --existing-disasters ./Flamborough/Flamborough.csv'
```

Dockerfiles

```
backend > docker > hub > 🐫 Dockerfile
1  # This Dockerfile sets up a Python 3.8 environment for running a Python application.
2  # It installs the required dependencies from requirements.txt and sets up the working directory.
3  # The application is configured to run with customizable host and port arguments.
4  # Example of how to build and run this container:
5  # 1. Build the Docker image:
6  #     docker build -t hub .
7  # 2. Run the Docker container:
8  #     docker run -p 3000:3000 hub
9  FROM python:3.8
10
11 WORKDIR /app
12
13 COPY requirements.txt .
14
15 RUN pip install -r requirements.txt
16
17 COPY . .
18
19 ARG HOST=0.0.0.0
20 ARG PORT=3000
21
22 ENV HOST=${HOST}
23 ENV PORT=${PORT}
24
25 CMD ["python3", "hub.py", "--host", "${HOST}", "--port", "${PORT}"]
```

```
backend > docker > subhub > 🐫 Dockerfile
13
14 COPY requirements.txt .
15
16 RUN pip install -r requirements.txt
17
18 COPY . .
19
20 # Set default values for the arguments
21 ARG HUB_HOST=127.0.0.1
22 ARG HUB_PORT=3000
23 ARG IP=40.233.92.183
24 ARG HOST=0.0.0.0
25 ARG PORT=3000
26 ARG ID=2
27 ARG NAME=Dundas
28 ARG LONGITUDE=-43.26691127633942
29 ARG LATITUDE=-79.77373983912754
30 ARG RADIUS_KM=3
31 ARG EXISTING_DISASTERS=./Dundas/Dundas.csv
32
33 # Set environment variables based on the arguments
34 ENV HUB_HOST=${HUB_HOST}
35 ENV HUB_PORT=${HUB_PORT}
36 ENV IP=${IP}
37 ENV HOST=${HOST}
38 ENV PORT=${PORT}
39 ENV ID=${ID}
40 ENV NAME=${NAME}
41 ENV LONGITUDE=${LONGITUDE}
42 ENV LATITUDE=${LATITUDE}
43 ENV RADIUS_KM=${RADIUS_KM}
44 ENV EXISTING_DISASTERS=${EXISTING_DISASTERS}
45
46 CMD ["python3", "subhub.py", "--hub-host", "${HUB_HOST}", "--hub-port", "${HUB_PORT}", "--ip", "${IP}", "--host", "${HOST}", "--port", "${PORT}", "--id", "${ID}",
", "--name", "${NAME}", "--longitude", "${LONGITUDE}", "--latitude", "${LATITUDE}", "--radius_km", "${RADIUS_KM}", "--existing-disasters", "${EXISTING_DISASTERS}"]
```

Dynamic Input

```
→ backend git:(main) ✘ python3 hub.py -h  
usage: Hub [-h] [--host HOST] [--port PORT]
```



Run a hub instance that users can report to

options:

- h, --help show this [help](#) message and [exit](#)
- host HOST The host to run the hub on
- port PORT The port to run the hub on

```
→ backend git:(main) ✘ python3 subhub.py -h  
usage: Sub Hub [-h] [--hub-host HUB_HOST] [--hub-port HUB_PORT] [--ip IP] [--host HOST] [--por  
[--existing-disasters EXISTING_DISASTERS]
```

Run a sub hub instance that users can report to



```
options:  
-h, --help show this help message and exit  
--hub-host HUB_HOST The host of the primary hub  
--hub-port HUB_PORT The port of the primary hub  
--ip IP The ip of the sub hub that can be accessible and seen by the primary h  
--host HOST The host to run the sub hub Flask API on  
--port PORT The port to run the sub hub Flask on  
--id ID The id of the sub hub  
--name NAME The name of the sub hub  
--longitude LONGITUDE The longitude of the sub hub  
--latitude LATITUDE The latitude of the sub hub  
--radius_km RADIUS_KM The radius of the sub hub in kilometers  
--existing-disasters EXISTING_DISASTERS The path to a CSV file containing existing disasters
```

Unit Tests and Continuous Integration

```
github > workflows > %_ ci.yaml
1  name: Unit Tests
2
3  on:
4    workflow_dispatch:
5    push:
6      branches: [main]
7    pull_request:
8      branches: [main]
9
10 permissions:
11   contents: read
12
13 jobs:
14   unit-tests:
15     runs-on: "ubuntu-latest"
16     steps:
17       - uses: actions/checkout@11bd71901bbe5b1630ceea73d27597364c9af683 # v4.2.2
18       - name: Install
19         working-directory: backend
20         run:
21           pip3 install -r requirements.txt
22           pip3 install pytest
23
24       - name: Run tests
25         run: pytest backend/tests
26
27
```

The screenshot shows the GitHub Actions interface for a pull request. At the top, a green checkmark indicates a successful merge of pull request #3 from JeremyTubongbanua/backend-1 into the main branch. Below this, the 'unit-tests' job is selected in the sidebar. The 'Annotations' section shows one warning. The 'unit-tests' job log details the execution of the workflow, including setting up the job, running actions/checkout, installing dependencies, running tests (which passed), and post-run actions/checkout. The final log output shows the pytest command being run, the test session starting, platform information, collected items, and the final result where 1 test passed in 0.01s.

```
<- Unit Tests
✓ Merge pull request #3 from JeremyTubongbanua/backend-1 #17
Re-run all jobs ...
Summary
Jobs
unit-tests
Annotations
1 warning
Search logs
unit-tests
succeeded 45 minutes ago in 8s
Set up job
Run actions/checkout@11bd71901bbe5b1630ceea73d27597364c9af683
Install
Run tests
Post Run actions/checkout@11bd71901bbe5b1630ceea73d27597364c9af683
Complete job

Run tests
Run pytest backend/tests
=====
platform linux -- Python 3.12.3, pytest-8.3.4, pluggy-1.5.0
rootdir: /home/runner/work/OEC_2025/OEC_2025
collected 1 item
backend/tests/test_is_point_in_circle.py .
[100%]
=====
1 passed in 0.01s =====
```

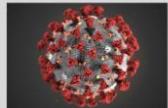
Initial Figma Prototyping

Initial Design: iteration 1

Desktop - 9

Find my Region (Button)

IMPORTANT !! CURRENT DISASTERS



COVID-19 Variant A
Reports: 389
Longitude: XYZ
Latitude: XYZ

/dashboard

Subscribe to Region:

Enter Address

Find My Region

/dashboard

Currently Subscribed Region: Toronto, ON M1T3N1

Reported at (73.1231, 121.311) COVID-19 Variant A

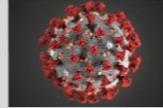
Desktop - 3

Enter Address (Text Box)

Find my Region (Button)

Your Address: ABCDEF

IMPORTANT !! CURRENT DISASTERS



COVID-19 Variant A
Reports: 389
Longitude: XYZ
Latitude: XYZ

Report A Disaster

Desktop - 4

Desktop - 7

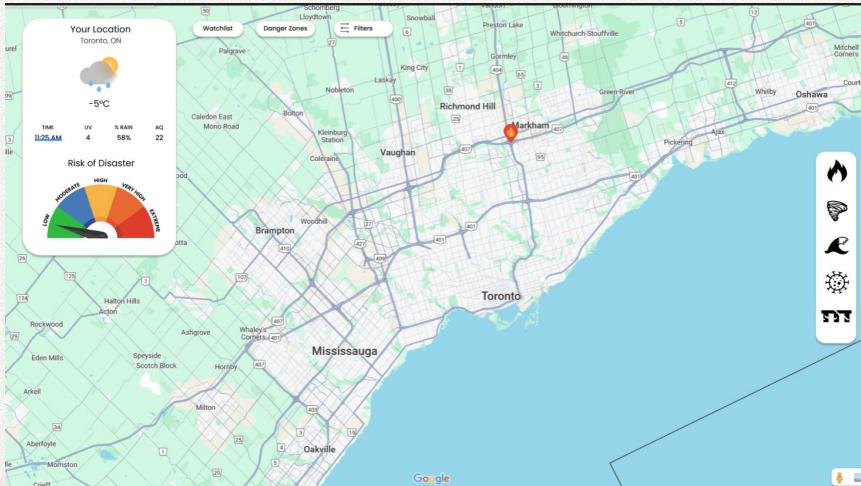
Report A Disaster

Name: Autocomplete To Existing Disasters

Location: Automatic from address fetching

Desktop - 8

Initial Design: iteration 2



The figure shows a screenshot of the NDctrak web application. The header is red with the text "NDctrak". Below it is a white card titled "Report New Disaster". The card contains the following fields:

- Disaster Classification:** An input field with a placeholder icon of a fire.
- Type of Disaster:** An input field with a placeholder icon of a lightning bolt.

At the bottom of the card is a red "SUBMIT" button.

Initial Design: iteration 3



Home Find Disasters Report a Disaster Admin

There are currently 4 disasters near you. Stay safe and take necessary precautions!
For more information, please click [here](#).

Disaster Tracker

Enter an Address to Find Disasters Near You

Address

Radius

100

Submit



Disaster Tracker

Enter an Address to Find Disasters Near You

Address

1280 Main St W, Hamilton, ON L8S 4L8

Radius

100

Submit

Found Disasters:

Earthquake

Major earthquake!!

Disaster Type:

natural

Radius:

50.0 km

Longitude:

-119.2437

Latitude:

34.0522

Oil leak

Hazardous oil spill requiring

containment

Disaster Type:

manmade

Radius:

3.0 km

Longitude:

-79.7737983912754

Latitude:

43.20691127633942

Tornado

Severe windstorm causing widespread

damage

Disaster Type:

natural

Radius:

50.0 km

Longitude:

-79.87690025166192

Latitude:

43.25133559904124

Transportation accident

Major vehicle collision incident

Disaster Type:

manmade

Radius:

10.0 km

Longitude:

-79.9169897193221

Latitude:

43.15423471652471

Home Find Disasters Report a Disaster Admin

There are currently 4 disasters near you. Stay safe and take necessary precautions!
For more information, please click [here](#).

Report a disaster

Report a Disaster Near You

Address

1280 Main St W, Hamilton, ON L8S 4L8

Longitude

-79.92267683

Latitude

43.2609974

Disaster Type

Natural

Name

Earthquake

Description

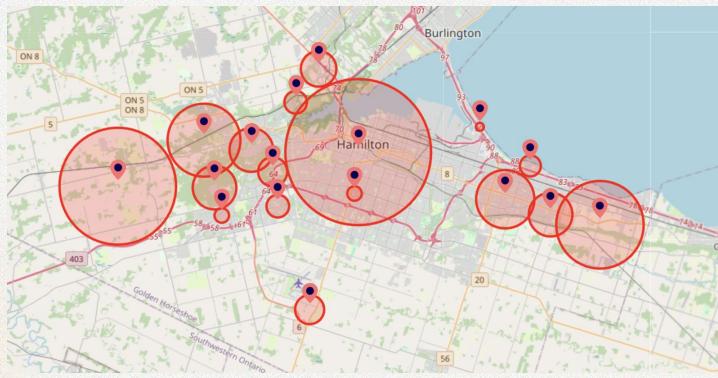
123

Radius

100 km

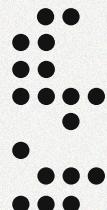
Get My Location

Submit





Development Stack & Toolkit



Frontend Technologies

Developer Tools:

- Git
- Github
- NPM
- VScode

- **Frontend Technology Stack:**

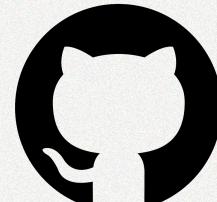
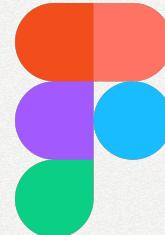
- React.js
- Next.js
- shadcn/ui
- TailwindCSS

- **Libraries**

- React-hook-form
- Leaflet.js
- React-Leaflet
- Zod

- **Prototyping & Design:**

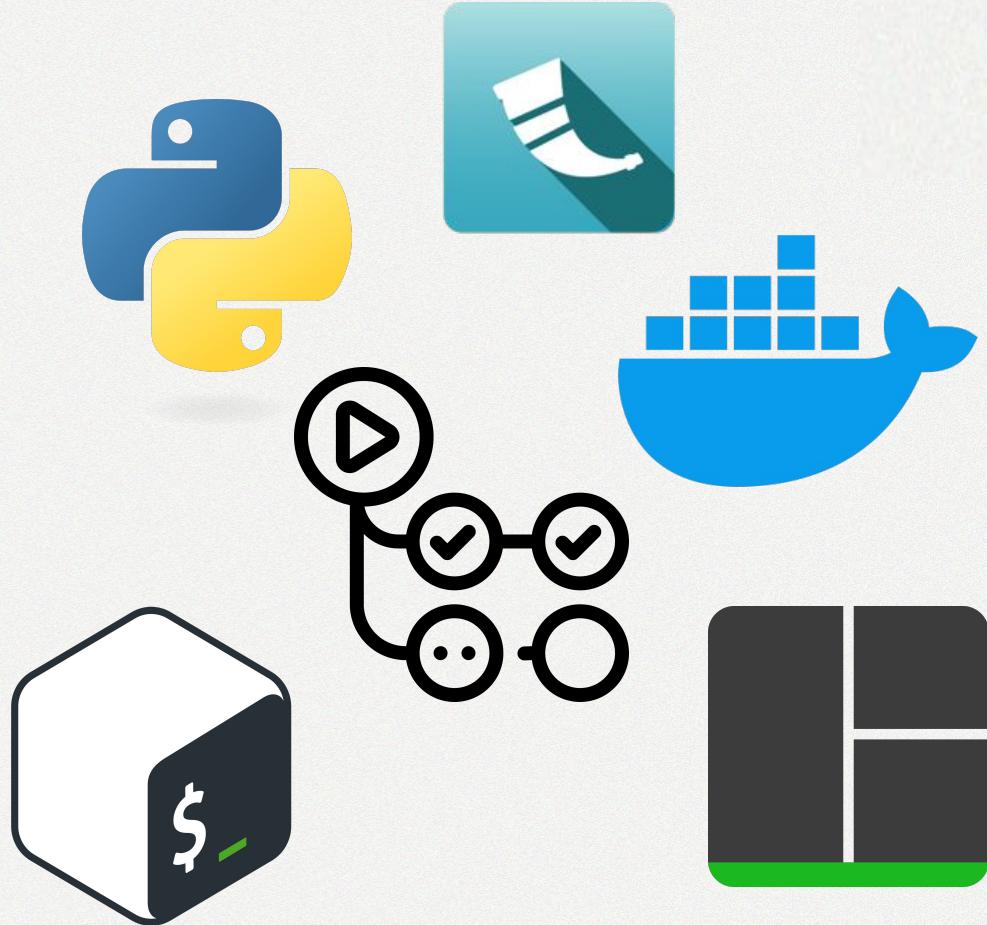
- Figma



Backend Technologies

Developer Roots.

- Git
- Github
- NPM
- VScode
- **Backend Technology Stack:**
 - Flask
 - Python
 - Docker
 - Github Actions
 - Tmux
 - Shell Scripts



Technical Challenges

- Implementing the interoperability and subscription between Hubs and Sub Hubs
- Map interactivity was difficult to properly implement with radius markers and location markers
- Interaction between backend and frontend was difficult to implement
- Implementing scalability techniques within the server architecture was to brainstorm and also implement



Future Improvements

- Improve **responsiveness** for the frontend to allow for a better user experienced
- Allow users to add **more detail** for the disasters when reporting
 - Severity levels which could be indicated on the map
 - Incorporate more complex data models to enhance complexity of disaster tracking
- Improve **UX** by adding filters and more options for deleting/editing disasters
- Add a footer.
- Add a form of **immediate communication** with emergency services and **notifications**
- Support for **multiple languages** (most notably french)



Thank You!