Diminishing Disasters: Phase 2

Jonathan Debella, Dani Amir, Ruchi Bhalani, Anthony Modica, Martin Nguyen

March 28, 2022

1 Motivation

We created diminishing Disasters to elevate peoples' consciousness to the international sphere. Most of the population is prone to pay no mind to events that don't affect them personally, while individuals who do make an effort to step out of their filter bubble and seek information about global crises often feel overwhelmed by the overload of news streams. Our website aims to connect people with natural disasters clearly and intuitively, displaying natural disasters worldwide by country and directing users to charitable organizations that allow them to help. By providing this direct pathway to disaster relief, our goal is to encourage people to educate themselves about the crises other countries are currently facing and facilitate the process of providing donations and aid relief.

1. What current disasters are currently going on around the world?
2. What types of disasters most frequently occur in a given country?
3. Where can I donate or help out in these disasters? Which countries suffer from a lack of organized charities?

2 User Stories

2.1 Phase 1

2.1.1 Filter by if a disaster is being attended to by at least one organization

- **Story**: I'm from the American Red Cross, and I believe that we would greatly benefit from this website, especially if we could use it to determine which disasters need more help at a given time and which ones are already being helped by one or many organizations.
- **Solution**: We can add a filter for disasters based on whether or not at least one organization is currently helping out. With this feature, organizations can more easily determine which disasters to prioritize. This functionality will be implemented in Phase 2 when filters will become operational.

2.1.2 View severity of disasters

- **Story**: As a user of Diminishing Disasters, in addition to sorting by the time of occurrence, I would like to be able to sort disasters by severity. The sooner we can get help to those who need it most, the better!
- **Solution**: The customer wants "persons killed/injured" as one of the filters so that users can sort by which disasters currently have the highest mortality rate and which ones need aid quickest. However, this information is not included in our API. The descriptions of disasters on each disaster instance page will outline the severity of disasters, but there is no metric by which we can quantify disaster severity through our API, so we will not be implementing this functionality in our project.

2.1.3 Extra Visibility on Landing Page

- **Story**: As a user of Diminishing Disasters, I would like to gain more visibility on causes I can contribute to, specifically those that need it the most. There is potential for difficulty in finding the institutions that could do the most for a particular cause when users must go through the process of searching/filtering themselves.
- **Solution**: We will be including lists of charities that are helping out specific disasters on the instance pages for those disasters. We have implemented this aspect in Phase 1.

2.1.4 Data Visualization for the Models

- **Story**: As a user of Diminishing Disasters, I would love to be able to see data visualization on the given disasters. Maybe seeing some graphs or charts on how often these disasters occur in a given country, or if a given disaster is more likely to occur near tectonic plate boundaries. Having visible data visualizations would help visual learners interpret what is going on and help those looking at the data prepare for possible future disasters.
- **Solution**: We can include infographics of disaster types in each country on the country pages. Additionally, we can include information on which countries this type of disaster is most prevalent in for each disaster. These graphics allow users to see a visual representation of which disasters countries most frequently face. This functionality will be implemented in Phase 2.

2.1.5 Interactive Map Visuals of Disaster for a Specific Country

- **Story**: As a user of Diminishing Disasters, I would like to visualize all of the potential disasters for the country I am in. This could potentially help me and other users make proactive decisions about reacting to certain disasters in and around my area. An example of supporting this could be a map of a certain area/country that visualizes and shows all previous or current disasters, so they are all highlighted in a standard place.
- **Solution**: We can provide a map that is constantly updated with information about current and ongoing disasters so that users can visualize how their location is impacted. We will implement this graphic in future phases.

2.2 Phase 2

2.2.1 Summarize Disaster Descriptions/Updates

- **Story**: The descriptions/updates for each disaster are very informative. If possible, I'd love to see a summary of the description/updates on each disaster page alongside the full ones so that I can quickly understand the current situation. A possible way to accomplish this is to highlight key points in the description/updates, such as: number of people affected (injuries, deaths, displacements, etc.) and other information y'all think is important.
- **Solution**: We can add edit the styling so that numbers and statistics are bolded, so that users will be able to glean the key points of a report without having to thoroughly read the whole thing.

2.2.2 Make Pictures Clickable on Splash Page

- **Story**: Hey, as a user of Diminishing Disasters I would love to see the splash/home page be a little bit more user-friendly. Currently, I am only able to click on the "learn more" buttons on this page and not the pictures associated with these buttons. Would it be possible to implement this in such a way that clicking on the associated photo links to the same thing as the "learn more" buttons? I feel that this would make the website a bit more intuitive.

- **Solution**: We can make the images on the Splash page clickable, just as the buttons currently are.

2.2.3 Include Interactive Map on Country Instance Pages

- **Story**: As an active user of your website, I was wondering if it would be possible to add an interactive map to your country's instance pages rather than a static map. Perhaps having a google map that lets you move around the country would be beneficial to explore a region better than having a singular zoomed-out static map.
- **Solution**: We will use the Google Maps API to include an interactive map on the website, so that users will be able to better see the relation between geographical region and natural disasters.

2.2.4 Images next to Countries listed on /countries/

- **Story**: I've just come across your website and have found the website quite useful in expanding my knowledge on both disasters in a given country, and organizations doing their part in assisting those affected by the disasters. With that said, I feel like the countries page is a bit bland in that it just seems to be a list of the countries with associated information. Would it not be possible to implement this list in a way where the countries' flag is displayed next to their name, or changing the format so there is some visual element representing each country?
- **Solution**: On the page with all the countries listed, we can include the flag of each country next to it, to make the site more visually appealing.

2.2.5 How do organizations provide relief to an area affected by a disaster?

- **Story**: I came across the Diminishing Disasters website after looking for different organizations that were helping provide relief to areas that experience natural disasters. While your page certainly goes into detail about the organization and the disasters in which the organization has provided relief, there doesn't seem to be any information on "how" they provided relief. What quantifies providing relief? Does simply sending money count as relief, or does the organization need to provide additional resources such as manpower to quantify as providing relief. Or, is that information not provided by the organization leading to the lack of it being on your webpage?
- **Solution**: We determine which charities provide relief to which disasters based on the "causes" listed for each charity that the Charity Navigator API provides us. Most of these organizations provide development and "relief services", which includes manpower. If users wish to learn more about specifically what each charity does and how the help out, the link to the official charity website is on each instance page.

3 RESTful API

We made a RESTful API for the back-end of our website using Postman. We defined paths and schemas for our models: countries, disasters, and charitable organizations. The three RESTful API's we used for this project was ReliefWeb, Charity Navigator, and Rest Countries. The link to documentation for our API is here.

4 Models

The three models for this website are countries, natural disasters, and international charitable organizations. Natural disasters, when they occur, often impact multiple countries. Charitable organizations hosted in certain countries provide relief for these disasters by collecting donations.

From countries to disasters there is a many-to-many relationship. From disasters to charities there is a many-to-many relationship. From countries to charities there is a one-to-many relationship.

4.1 Countries

- Name
- Time Zone
- Capital City
- FIFA Code
- Area
- Filters
  - Population
  - Language
  - Currency
  - Region
  - Subregion
- Type of media to display
  - Country flag
  - Country map
  - Charts and graphs displaying disaster types

4.2 Disasters

- Name
- Description
- GLIDE Number
- Primary country
- Latest updates
- Filters
  - Status
  - Disaster type
  - Year
  - Month
  - Affected country
- Type of media to display
  - Infographics
  - Natural disaster maps
  - Charts and graphs displaying countries most frequently affected

4.3 Charitable Organization

- Name
- Website
- Address
- Phone number
- EIN
- Filters
  - Rating
  - Total expenses
  - Category
  - Cause
  - Receives government support
- Type of media to display
  - Logo
  - Embedded video media

5 Tools

5.1 Development

We utilized React for the front-end development of the website and set up Postman for the back-end design, defining GET endpoints for each model and a schema for the returned JSON object. We used Git to ensure continuous integration and minimize merge conflicts. Our main challenges for this phase involved familiarizing ourselves with the tools we would be using to build our website. Becoming familiar with React Bootstrap usage was challenging, as was learning how to consume the API. It was tricky figuring out how to organize the many different JavaScript files we had, and piecing together the different React components, but after we planned out our project and began working on it, using the tools came more naturally.

For deployment of our website on target branches, and setting up development environments, we used Gitlab Pipelines and Docker, with separate Docker images for the front-end and back-end. For the Python back-end, we used Flask and SQL Alchemy. For the database in which we store information about each instance of a country, disaster, and organization, we used PostgreSQL, hosted on AWS RDS and pgAdmin to monitor the state of the database as we added instances to it.

For data collection for our models, we used the Rest Countries API, the Charity Navigator API, and the Relief Web API. One of our biggest challenges was querying the Relief Web API for the rich media for natural disasters, which included maps and infographics, because of the way the API was set up.

5.2 Testing

For front-end JavaScript testing, we used Jest as the testing framework, as well as enzyme to create assertions for different React components, such as the About and Splash pages, as well as each model page.

For front-end GUI testing, we used Selenium as the testing framework. We tested site navigation by testing the validity of the links on the page.

For front-end unit tests, we used Jest, a JavaScript testing framework to check if our JavaScript is working properly. We made sure that our pages rendered correctly and not crashing.

For back-end unit tests, we used the Python unittests library, and tested the JSON responses from various API calls to make sure that they were returning valid and correct information.

For API tests, we used Postman, and we tested the JSON schema by using GET requests and checking the length and content of the responses.

6 Hosting

We are using AWS Amplify for our static website hosting. We used NameCheap to register our domain name and verify ownership with Amazon Web Services to register a TLS/SSL certificate and ensure our website was HTTPS secure.

For our back-end, we are using AWS Elastic Beanstalk for deployment and AWS RDS for database hosting. Our deployments for Elastic Beanstalk was done through GitLab's CI/CD. We also utilized environment variables in order to be able to deploy easily.

7 Database

When we were designing the database, we had to decide how we were going to present the information into the tables as well as displaying them on our website.

Deciding the column names was easily done through the attributes from the APIs we scraped as well through our constructed Postman API.

Regarding the relationships, we decided that we were to have a many-to-many relationship between all three models due to how they are connected. As a result, we had to build two association tables in order to link the three models together properly.

We trimmed and scraped our data by utilizing JavaScript in order to match the models together and be able to construct relationships.

We then implemented a Python script using SQLAlchemy to be able to convert all of our data from raw JSON into SQLAlchemy models and then push them into our database.

8 Pagination

For this phase, we implemented pagination on the front-end with the help of the Table Pagination component from the React library, which took in several props such as the current page, a function to handle page changes, and the total count of instances from the model.

To implement pagination, we first made a call to our API endpoint which returned all of the model's instances. Then we converted the returned JSON object into an array, and sliced it in order to display the amount of rows that we wanted per page.