Southwest Hurricane Aid Technical Report - Final

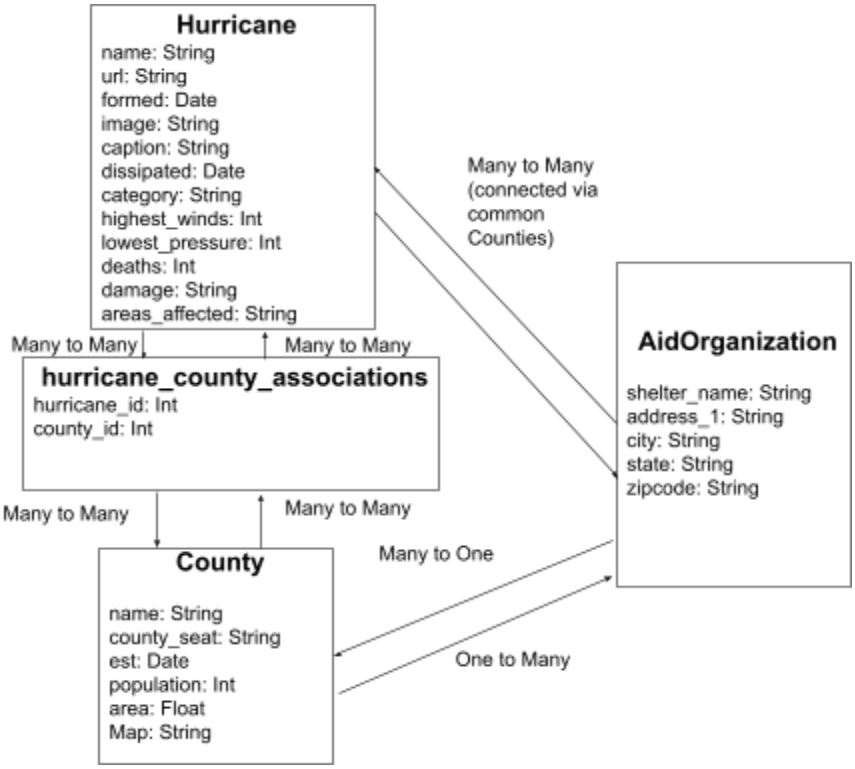
Introduction and Purpose

Hurricanes and tropical storms are a critical issue for many Texans living on the coast, and they will only become a greater issue in the future due to climate change. The goal of this site is to give resources to Texans who are in those areas, allowing them to easily research how hurricanes and tropical storms can affect them.

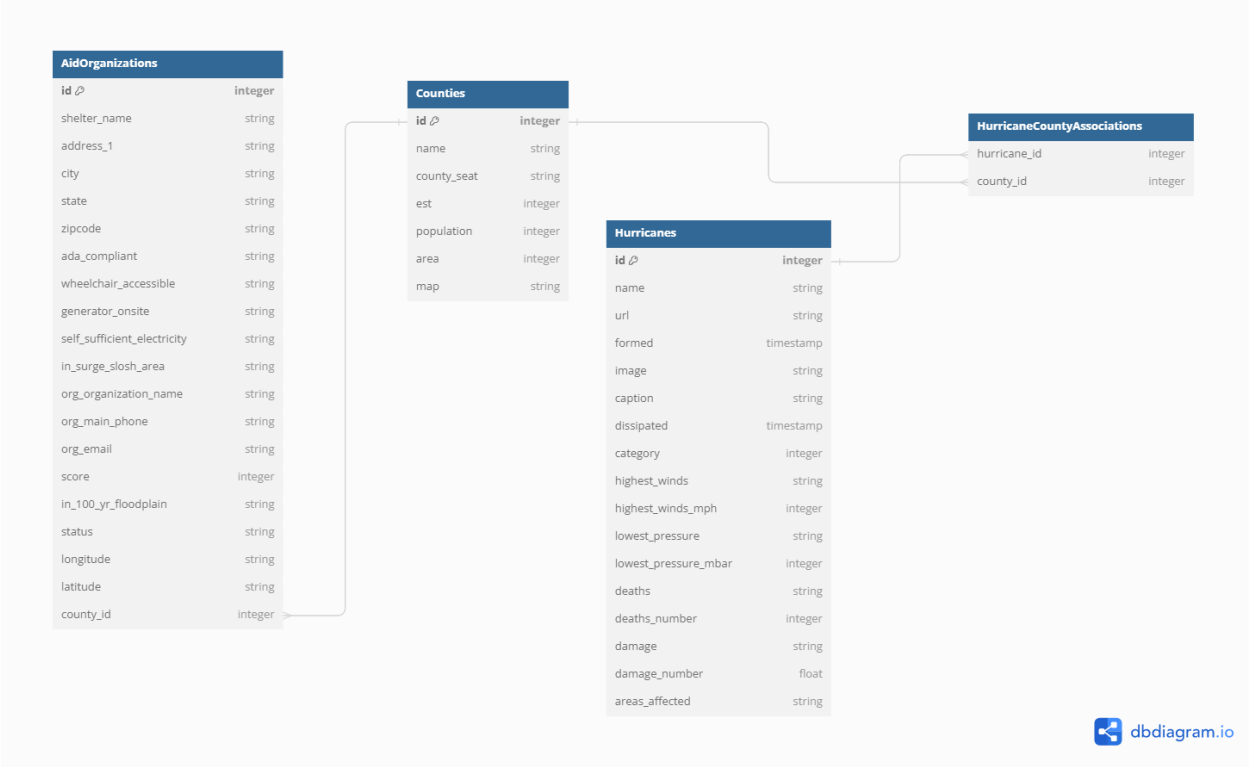
Database

The database is a PostgreSQL database that is run in Docker as its own container using the Postgres image. It uses SQLAlchemy to communicate with Flask, and all that really needs to be done to start it up is running docker-compose up with the build flag to build the container, and the database will be good to go from there.

Below is our UML Database Diagram:



Below is our UML Database Diagram:



User Stories

- Most of our user stories were relatively simple UI fixes, and this was the case for the user stories given to us and for user stories that we sent out.
- For a lot of our user stories, we had already done them on our own before we started working on the user stories, so we didn't have to do a lot of work for them.

Data Scraping

- For the hurricanes, we ended up scraping data from Wikipedia. We scraped data from two places: one site with all the links to recent hurricanes in Texas (https://en.wikipedia.org/wiki/Category:Hurricanes_in_Texas), and then the sites that those links led to. For each hurricane, there was a table near the top right of the article, and we got most of our statistics from there. We also looked at which towns were mentioned in the article, and then translated them into counties, which became the "relevant counties" section of a hurricane instance.
- For the counties, we also scraped from Wikipedia, but we only scraped data from one source: https://en.wikipedia.org/wiki/List_of_counties_in_Texas. This source contains a large table of information related to every county in Texas, and we took information from that table.
- For aid organizations, we used the National Shelter System Facilities API: https://hifld-geoplatform.opendata.arcgis.com/datasets/geoplatform::national-shelter-system-facilities/api

API Documentation

- REST API | GitLab
    - Used for collecting data for the About page
- Aid Organizations API
    - Using for the Aid Organizations page
- Wikipedia API
    - Used for both counties and hurricanes
- Postman API Docs
    - Full documentation for API on Postman

Models and Instances

Historic Hurricane data

- Hurricane Name
- Category
- Date
- Wind speeds
- Fatalities
- Hurricane Photo
- Counties affected

Texas Counties

- County Name
- Population
- Land Area
- Est. date
- County Seat
- County Map
- Historic Hurricane Hits (names)
- Aid Organizations in that county

Aid Organizations

- Facility Name
- Address
- Organization Name
- County
- Score
- Facility Map
- Historic Hurricane Hits (names)

Tools and Hosting

Frontend:

- React - main Typescript framework
    - React Router is used for routing
- Material UI - main CSS framework. We use many Material UI components in this project, including Box, Link, and Card.
- Namecheap - domain registration
- npm - package manager

- Docker - containerization application we used to run the React front end

Backend

- AWS - used EC2 instance (VM) for hosting database and backend
- Flask - used to create the backend and create a RESTful API
- SQLAlchemy - used by Flask to communicate with the PostgreSQL database in the docker container
- Adminer - used as a tool to allow us to easily view the database, i.e. the tables and data within them, without having to write raw SQL and giving us an easy GUI to interact with the database
- PostgreSQL - the database that we chose to store our tables and data
- Docker - containerization application we used to run our database, adminer, and Flask backend in their own respective containers called “db”, “adminer”, and “web” respectively

Testing

- Selenium - for frontend acceptance tests
- Jest - for frontend unit tests
- Python UnitTest - backend unit tests
- Postman - API unit tests

Architecture

- Backend
    - Models - includes __init.py__, aid_organization.py, associations.py, county.py, hurricane.py
    - Scripts - includes code to scrape data
    - Dockerfile
- Frontend
    - Public
        - img - includes all of the images used in every page
    - src
        - About - directory for the component(s) of the About page
        - Aid Organizations - directory for the component(s) of the Aid Organizations model page
        - Aid OrganizationInstances - directory for the component(s) of an Aid Organization instance page
        - App - main component
        - Counties - directory for the component(s) of the Counties model page
        - CountyInstances - directory for the component(s) of a County instance page
        - Home - directory for the component(s) for the home page
        - HurricaneCard - card that is displayed on the /Hurricanes model page
        - HurricaneInstances - directory for the component(s) of a Hurricane instance page
        - Hurricanes - directory for the component(s) of the Hurricanes model page.
        - Navbar - directory for the components of the navigation bar
        - Our Visualizations - directory for components for our visualizations
        - Provider Visualizations - directory for components for provider visualizations
        - Search - contains component for the full text full website search

        - SortFilterBar - contains the shared component that enables sorting searching and filtering on model pages
        - Utils- contains component for allowing highlighted text.
        - Tests: using jest and selenium
    - Dockerfile
        - Runs npm install and then npm start

Challenges Faced

- One challenge we faced while doing the front end for the full-text search was being able to navigate to the page when the button was clicked in aligning it to the right on the NavBar so that it was separate from all other tabs.
- Another challenge that we ran into for sorting and filtering was trying to create a React component to handle the sorting and filtering for all 3 models. There ended up being a very long list of props for that component, and most of the props were the event handlers that would do API calls whenever the search/sort/filter criteria changed.
- Some challenges we faced with trying to implement sorting and filtering is that a lot of the columns were previously being stored as strings even though it was mostly numerical data in the Hurricanes table. This meant that we had to re-write the data scraping script and add more columns to store the numerical data as ints and floats so that we could actually filter and sort by these attributes, which took a lot of work in writing Regular Expressions to re-scrape the data as it was not always in the same format across all the Hurricane pages.
- Full text search was a large issue, as we had to try a lot of different things to get a full text search that we thought worked. Overall, we ran into some issues with trying to get multi-word searches to work but were able to fix that by encoding the search string to change spaces to a different sign so that postgres could take those in.
- One big challenge we faced towards the ending of phase 4 was trying to figure out how to create the visualizations. We struggled for a while to try to get D3.js to work with Typescript, as it was just throwing lots of errors. We overcame this by installing the types as another npm package or just using a .jsx file for some of the visualizations, but then we were unable to get D3.js to work, and even simple examples online seemed like a lot of work to get it to work easily with React. However, by understanding how other groups did it, we were able to create them ourselves using different libraries from D3.js
- We made use of two different libraries for visualizations based on preference of different members, ReactViz and Rechart.

Implementation of Phase II Features

Frontend Changes:

- Home
    - Added slideshow of hurricane images
    - Added a centered transparent box that contains the title
    - Description of the website under the title
    - Automatic changing of the slides every few seconds
- About
    - Member cards are now sizable and can adapt to screen size change
    - Raw HTML links converted to buttons

    - Tools added at the bottom of the page
        - Divided into the frontend, backend, and testing tools
        - Gray box sections off these properties
        - Images of tools are centered and labeled
    - Information at the top of the about page centered
    - Fixed the total commits count issue
    - Changed from 3 slides per row to 2
- Hurricanes, Counties, Aid Organizations Pages
    - Added the rest of the cards after backend was done
    - Several pages to handle all of the cards
    - Images added for each card
    - Google Maps integrated for Aid Organizations
    - Fixed the issue with cards clipping into each other; cards should now behave with varying screen sizes

Backend Changes:

- We created tables in our database for all 3 models using Flask and SQLAlchemy
- We also started to create the API for our website, and included 1 POST and 2 GET endpoints for all 3 tables, for a total of 9 endpoints. We had one GET endpoint to get a specific instance by id, and one "get all" endpoint that returns a paginated response for the instance. We had one extra endpoint for the County table, a GET with a "search" parameter to get the county ID of a county that matches the search parameter, and this was used to set up the relationships between hurricanes and counties as well as aid organizations and counties.

Implementation of Phase III Features

- We implemented the searching, sorting, and filtering algorithms on all of our attributes
- Created a UI React component to implement searching, sorting and filtering on model pages
- Hurricanes - expand more on all the models
    - Added parameters to the get API endpoint for hurricanes for a search term, a filter criteria (one of the attributes), a filter direction (<, >, =), a sort criteria (one of the attributes), and a sort direction (<, >, =)
    - Added numeric columns for wind speed, air pressure, and fatalities in order to get the sorting and filtering working
- Counties
    - Added parameters to the get API endpoint for hurricanes for a search term, a filter criteria (one of the attributes), a filter direction (<, >, =), a sort criteria (one of the attributes), and a sort direction (<, >, =)
- Aid Organizations
    - Added parameters to the get API endpoint for hurricanes for a search term, a filter criteria (one of the attributes), a filter direction (<, >, =), a sort criteria (one of the attributes), and a sort direction (<, >, =)
- We also included a full-text search to scan the entire website for a given input - expand on this

Implementation of Phase IV Features

- **Visuals**

- Created visuals to represent the data in our database using react-vis and recharts
- Accessed our developer's API to create visuals using react-vis
- Added pages to our site to display the visuals.