# Southwest Hurricane Aid Technical Report - Phase II

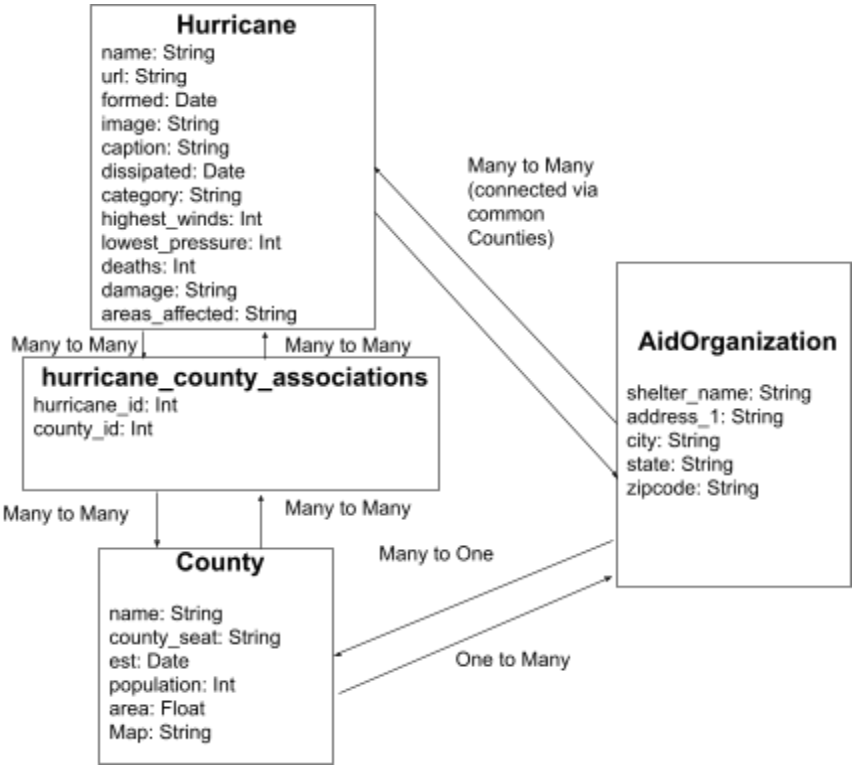
## Introduction and Purpose

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 do research on 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 Database Diagram:



## User Stories

- By the time we had started to look at our customer's input, we found that we had already solved most of the issues they gave us on our own. The issues that we hadn't already solved were relatively easy UI fixes.
- One tip we did take from our customer was to include images for all of the tools we used. We implemented this at the bottom of the about page and divided up al of the tools we used into Frontend, Backend, and Testing sections.

## 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

## 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
        - Tests: using jest and selenium
    - Dockerfile
        - Runs npm install and then npm start

## Challenges Faced

- One unexpected challenge was getting the Selenium tests to work, as Selenium relies a lot on identifying HTML by tag, ID, CSS selectors, or other attributes, which doesn't really agree with React, since React creates a lot of HTML elements with long and complex class names.
- Another challenge was figuring out how to scrape the hurricane links. Some of the links didn't actually lead to hurricanes, and some hurricanes weren't in the format that we wanted, so we had to filter out the good hurricanes.
- A challenge with the Frontend was to change components of the page to adapt to changes in the page size. For example, some of the components' positions on the page would change if we were to minimize or maximize the page

## Summary of Phase II

**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 setup the relationships between hurricanes and counties as well as aid organizations and counties.