Texas Homes Project: Phase IV

Jeronimo Del Valle Ocejo | Gabriel Casanova | Shahmir Masood | Ananth Kothuri

Purpose/Motivation:

We're on a mission to make it easier for people experiencing homelessness to receive the help they need. Texas Homes Project is envisioned to be an online resource for anyone looking to support the Texas homeless population. We aim to connect communities with nearby homeless shelters, support organizations that aid the homeless, and spread the word about upcoming volunteer opportunities.

User Stories:

Stories that we implemented:

- Fix image crop on some of the instance cards
    - We modified the images displayed on some instance cards so the image is cropped correctly
- Fix image scaling
    - We solved it by adjusting the aspect ratio in the images
- Add more statistics on counties page
    - We pulled and added more text, statistics, and details to the counties' page
- Clean up event instances
    - We updated the UI to a more clean, clear layout
- Add a slide deck of pictures of people in the homeless shelters on the landing page
    - We completed this issue with a slide show on our splash page

Stories that we gave:

- Update Splash Page image slideshow with clearer images
- Fix model instance cards image sizing
- Add actual ‘cards’ for the related models in instance pages
- Add a favicon that is more representative of your website
- Bring the site back up so that it is accessible to the public

Challenges:

Phase 4:

- This phase presented challenges different from the previous three. At this point in the process, we had virtually accomplished everything that we needed to do for this project. We implemented our site’s front-end functionality, integrated the back-end and API

servicing, and also allowed users to search/filter the data on our site. For Phase IV, the main hurdle for us was simply going through our entire codebase (the same one we'd been using for upwards of 4 months), and refactoring the heck out of it. This mainly posed issues when it came to refactoring code we'd written earlier in the semester, as the lack of descriptive comments often made it hard to understand what exactly we were doing in certain files. Nevertheless, we eventually got our codebase to a much tidier point (one that we'll all be proud of). Additionally, working on the presentation and PechaKucha slides presented a different challenge that we had yet to experience during the first 75% of the course.

Phase 3:

- This phase was challenging for reasons different than the previous two phases. The first phase was hard as we needed to start building our site from scratch, and the second phase was hard because we needed to implement our backend and API functionality, this third phase was hard since it required we refactor/rework previous code/components/functions and update them to be reusable across various parts of our site. Implementing searching was challenging at first, as we needed to figure out how exactly we wanted the logic to work. We wanted to reuse as much code from previous phases to make this phase as easy as possible. We decided we would implement our search functionality via filtering after we'd made the API call to receive all instances of a given model. Besides this, the other challenge was updating our user interface to incorporate new buttons/search fields, while also solidifying a more professional and aesthetic look to our website.

Phase 2:

- This phase of the project was difficult, especially considering how many moving parts there were. Our team was able to split up tasks between our members, but many tasks relied on other parts to be completed (populating the front end needed a function API, a functioning API needed a completed database, etc.). Additionally, since most of us were new to API development and using resources such as AWS, there was a large learning curve that required more time for debugging and development. However, by working together and using all of our resources, we were able to complete all of the required parts of our website.

Phase 1:

- For all of us, this project was our introduction to the world of web development. With the team's limited experience as a whole, it felt a bit daunting to find where to even start. On top of that, one of our team members ended up dropping the course, leaving only 4 people remaining to complete the project, rather than the typical 5. Despite these setbacks, we made sure to start early and utilized resources including YouTube videos and Ed Discussion to answer any questions we had along the way. By starting early, we were able to split the workload into more manageable chunks where each person had a

reasonable goal that wasn't too overwhelming to handle. Additionally, we made an effort to meet in person to collaboratively code and help each other debug.

Hosting:

We used AWS Amplify for website deployment and hosting.

API Documentation:

Our API documentation involves **GET** requests to poll information about our different shelters, counties, and events. These requests include retrieving all model instances, or specific model instances by ID. For more details about these requests, please refer to our Postman documentation: [Postman Documentation]

Models & Instances: Currently, to add more model data or instances to the website, this data should be stored within one of the model **JSON** files in the **front-end/src/data** folder. The data must follow the following format for it to work properly:

- **Shelters:**
  - **id**: internal id
  - **name**: name of shelter
  - **address**: address of shelter
  - **city**: the city where it's located
  - **state**: state where it's located
  - **zip_code**: zip code where it's located
  - **location**: coordinates of latitude and longitude
  - **phone_number**: the phone number of the shelter
  - **email_address**: the contact email address
  - **fax_number**: the fax number of the shelter
  - **official_website**: the website of this shelter
  - **twitter**: the twitter link for this shelter
  - **facebook**: the facebook link for this shelter
  - **instagram**: the instagram link for this shelter
  - **description**: a textual description of this shelter and what its goals/purpose is
  - **photo_urls**: a list of photos associated with this shelter
  - **update_datetime**: the last time this information was updated
  - **video_url**: an optional video URL for this shelter
  - **related_models**: a list of related shelters, events, or counties and their IDs.
- **Counties:**
  - **id**: internal id
  - **name**: name of this county
  - **population**: the population of this county

    - **housing**: the number of housing units in this county
    - **website_url**: the wikipedia url that describes this county
    - **description**: a textual description and summary of this county
    - **image_url**: an image URL of this county
    - **map**: a possible map image of this county
    - **images**: an additional list of images for this county
    - **related_models**: a list of related shelters, events, or counties and their IDs.
- **Events:**
    - **id**: internal id
    - **title**: the title of this event
    - **organization**: the name of the organization running this event
    - **image**: image URL for this event
    - **description**: a textual description of this event and what it consists of
    - **date_posted**: the date this event was posted
    - **time**: the time this event will occur
    - **cause_areas**: what causes this event hope to help with
    - **location**: the location of this event
    - **skills**: skills that are sought after for this event
    - **requirements**: requirements for people participating in this event
    - **related_models**: a list of related shelters, events, or counties and their IDs.

Frontend Architecture:

The front end of our website is a React.js project hosted on AWS Amplify. We use numerous Bootstrap components to ensure our website dynamically scales to all screen sizes. We also have a dynamic splash page (Home) and About page, as well as pages for Shelters, Events, and Counties models and instances.

Installation:

1. Clone the repo and ensure npm is installed
2. Run **npm install** to install all dependencies
3. Run **./install_dependencies.sh** within the **/front-end** folder
4. Run **npm start** to start the website on your local server

File System:

Our front-end file system is located within the **front-end/src** folder. This folder contains multiple subfolders, each corresponding to a different part of the website:

- **Main Tabs**: Each tab on our website has its own folder for its content
    - **Home**: /front-end/src/Home/HomePage.jsx
    - **About**: /front-end/src/About/AboutPage.jsx
    - **Shelters**: /front-end/src/Shelters/SheltersPage.jsx

    - **Counties**: /front-end/src/Counties/CountiesPage.jsx
    - **Events**: /front-end/src/Events/EventsPage.jsx

- **Model Templates**: These are components that we reuse across all models. This allows the webpage to change dynamically based on what type of model is being used.
    - **PageLayout**: /front-end/src/ModelTemplates/PageLayout.jsx
        - Creates a grid of models to be searched and filtered through
        - *Pagination*: fetch all models at once from our API, and split them into separate pages, with the ability to go to the next and previous pages
    - **InstanceCard**: /front-end/src/ModelTemplates/InstanceCard.jsx
        - A single instance card with a few attributes and features
    - **InstancePage**: /front-end/src/ModelTemplates/InstancePage.jsx
        - A page for a model instance with information and related models

- **Assets**: All of our assets are stored in the **front-end/src/assets** folder and include local images, member photos, and tool icons.

- **Components**: Other components that were used, such as the NavBar, are stored in the **front-end/src/components** folder.

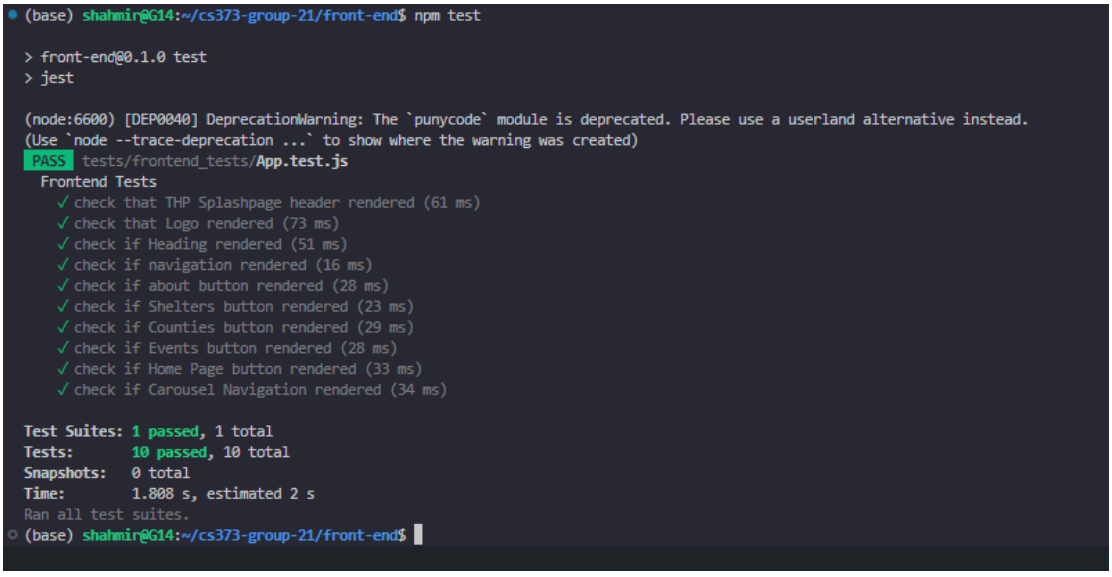
- **Static Metadata**: The **front-end/src/data** folder contains all our static metadata for the models, as well as information that is used to populate the About page.

Frontend Testing:

The frontend testing using jest can be run by **npm test** in the <root dir>/front-end/ directory. You will likely need to run **npm install** to ensure you have the proper dependencies and installations. The tests are located in /front-end/tests/App.test.js

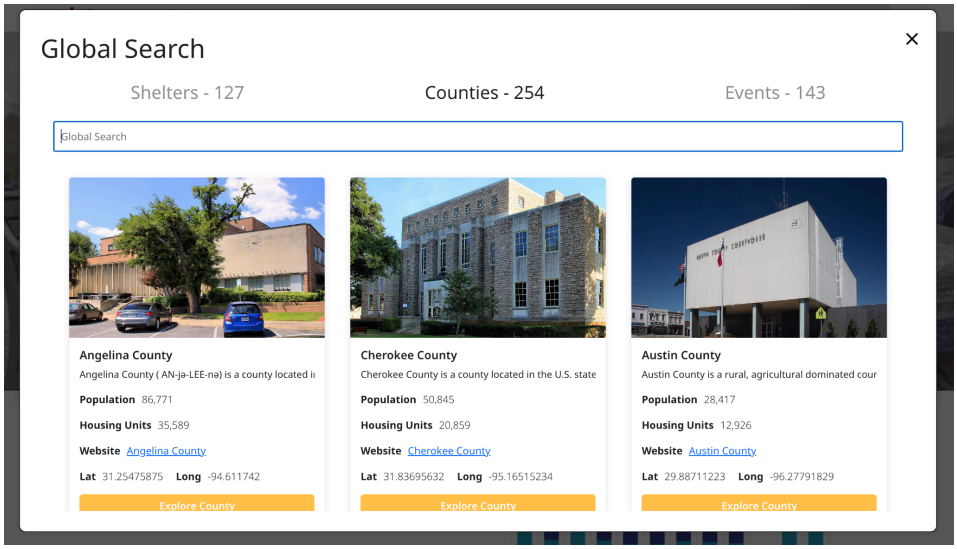
A description of what each test is doing is listed in the output.

Expected Output:



Searching:

We have a global search on the splash page of our website where users can search through all relevant instances (for each of the three models) for any given query. When clicking on the “Global Search” button on our NavBar, the user will be prompted with a search dialog screen where they may enter search terms and find relevant instance cards. We’ve categorized our results for each of the three models (Shelters, Counties, Events) to be reachable by clicking on tabs indicating the relevant search results for a given model. We decided this method of displaying the results would be most efficient for the user to quickly find relevant instance cards by model while also being able for the developers to rely on previous code written when displaying relevant instance cards on the model pages. An example of the user-interface dialog is pictured below:



Similar searching functionality exists and is accessible for each model type.

Sorting:

Each model page has options to sort by certain criteria (displayed via a dropdown menu). These options are unique per model and allow users to sort the instance cards in ascending order based on these criteria. The following is a breakdown of the currently offered sorting criteria:

- **Shelters:**
  - city
  - name
  - address
- **Counties:**
  - name
  - housing_units

    - population
- **Events:**
    - title
    - organization
    - date_posted
    - address

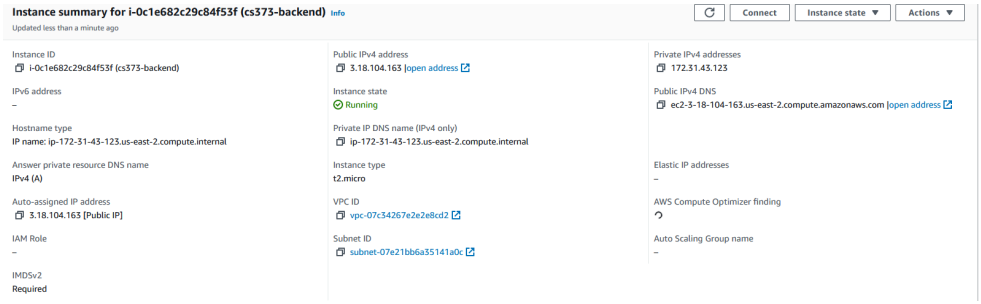
Backend Architecture:

To set up the backend hosting, we followed the tutorial here.

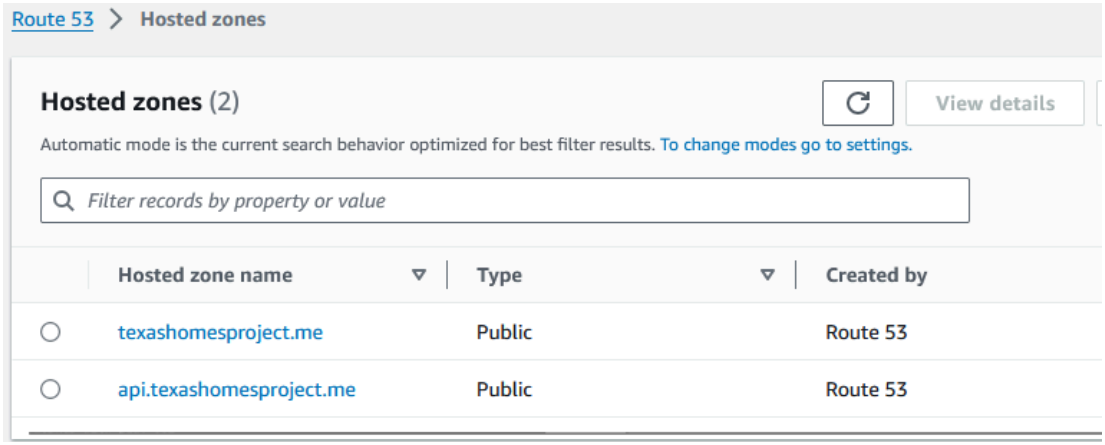
It is entirely set up on Amazon web services, and the API backend website can be found here: http://api.texashomesproject.me/

In particular, here is the structure of our server.

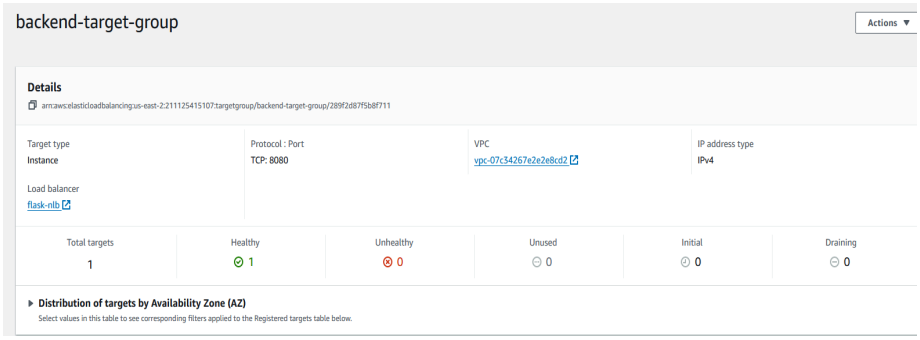
- **EC2 instance**: named cs373-backend
    - On the instance, we have a trivial flask app and our docker installation. We currently have it set up to use the Gunicorn Flask production server.



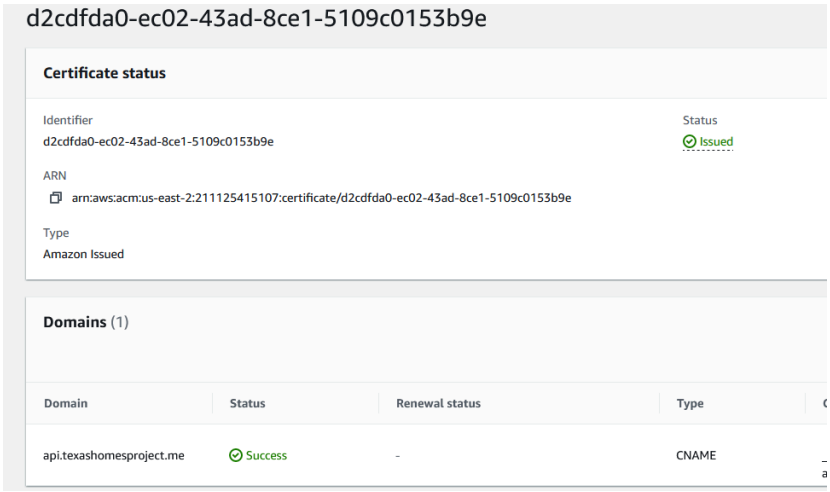
- **Route 53 Domain:** named api.texashomesproject.me
    - This hosted zone is in route 53 by copying the CNAME record from the Namecheap domain



- **Network Load balancer:** named flask-nlb
    - The NLB distributes incoming traffic from our domain to our specified target group
    - **Target group:** named backend-target-group



- **ACM certificate**: named d2cdfda0-ec02-43ad-8ce1-5109c0153b9e



Data:

Our data came from a variety of sources, including web-scraped websites and RESTful APIs. All of our code is located in the **back-end/scraping folder**.

County Data:

- CSV of counties, coordinates, and populations: https://data.texas.gov/widgets/ups3-9e8m?mobile_redirect=true
- **US Census data** - web scraped: https://data.census.gov/profile/
- **Wikipedia API** for county descriptions: https://www.mediawiki.org/wiki/API:Main_page

Shelter Data:

- **Homeless Shelter API:** https://rapidapi.com/topapis/api/homeless-shelter

Events Data:

- **VolunteerMatch** - web scraped for events: https://www.volunteermatch.org/search
- **VolunteerTX** - web scraped for events: https://www.volunteertx.org/search/i/

Additional APIs:

- **YouTube API** - used to find videos based on shelter/event names: https://developers.google.com/youtube/v3
- **Geocoding API** - used to find latitude and longitude coordinates for instances: https://developers.google.com/maps/documentation/geocoding/overview

Tools Used:

These are the tools we used for phase 1 of our website:

- **AWS Amplify** - used to deploy our website
- **Docker** - used when running the GitLab pipeline
- **GitLab** - used for version control and collaboration among members
- **Postman** - used to document our future expected API behavior
- **React** - chosen as the frontend library for developing our interactive UI
- **Bootstrap** - used for uniform and visually appealing UI components