Syrian Refugee Crisis: Phase 1

IDB Group: 18
Shadow Sincross, Matthew Albert, Pranitha Kolli, Anish Madgula, Matthew Castilleja

Motivation / Overview

Increasing outreach for Syrian refugees is a pressing humanitarian imperative. It entails providing essential resources like food, shelter, healthcare, and education to those who have been displaced by conflict. Outreach efforts also extend to helping refugees integrate into host communities, offering mental health support, facilitating employment opportunities, and providing legal assistance.

Our project serves to assist Syrian refugees with resources as well as educate the public on the Syrian refugee crisis. We will be providing charities and organizations across many countries which can assist Syrian refugees with housing, medical care, education, etc. This site can be used as a great resource to help increase Syrian refugee outreach across the globe. To build this website so far, we have utilized JavaScript (React framework), CSS (Bootstrap), HTML, and AWS. We have also utilized public APIs from various sites, including UNHCR, Wikipedia, and Relief Web. In the future, we will have a Python (Flask framework) backend with a MySQL database.

Models

1. Charities
    a. Filterable Attributes:
        i. Name of organization
        ii. Year established
        iii. Parent organization
        iv. Awards received
        v. Location of headquarters
    b. Media:
        i. Logo of charity
        ii. Picture of headquarters
2. Countries
    a. Filterable Attributes:
        i. Country name
        ii. Number of Syrian refugees in country
        iii. Number of Asylum decision in country
        iv. Year of Asylum decisions

        v. Number of individuals granted asylum
    b. Media:
        i. Image of Country’s flag
        ii. Map of country
3. News/Events
    a. Filterable Attributes:
        i. Date of event
        ii. Location of Event
        iii. Disaster type
        iv. Media source
        v. Themes
    b. Media:
        i. Image of event
        ii. Logo for news/events source

User Stories

We created issues for all of the general requirements that were detailed for this phase. We put them on an issue board and divided them into various labels/categories, including Customer, Frontend, Backend, General, and Project Prep.

Customer Issues:

We received 4 issues from our customer team. They involve collecting a lot of data that is not accessible from our current sources and complex features that require more time than was given for Phase 1. We will continue to work on them across the next few phases while maintaining a dialogue with the Customer team.

Restful API

We created our postman api documentation by importing from the example api documentation. All of the models can be accessed through different requests. Each of our models has one request with one example request, as well as their individual instances.
Documentation: https://documenter.getpostman.com/view/30070229/2s9YJZ3jaX

Tools

Frontend:

- React: Main UI framework
- React-Bootstrap: CSS Framework
- React Router: Library for implementing website routing/navigation
- NameCheap: Domain name registrar
- AWS Amplify: Cloud-based hosting service

- NPM: Package manager for Node.js
- Unsplash: Provides free images for commercial and personal use

Backend:

- Python
- Requests: Python library for HTTP requests
- IPython Notebook: Used for developing Wikidata scraping script

Hosting

Our project is currently hosted using AWS Amplify. We connected our GitLab repository to our Amplify project to set up continuous deployment of our app. This means that with each commit, the changes were immediately available on our project's domain name. We also obtained our domain name using NameCheap. In order to connect our domain name to our Amplify project, we created a hosted zone through Amazon Route 53. We then configured our name servers on Namecheap and finally added the domain to our Amplify project. Route 53 was able to handle all of the verification of our domain. Amplify also allows us to modify our project's build settings to fit our specific requirements. These settings include commands to be executed at the start of our build, and the directories that they should be executed in.

Details

Frontend Architecture

src Directory:

index.js

- Contains the entry point of the application and logic for rendering all the necessary components that make up the website.
- Defines the overall route structure of the website

src/Components

- Contains the React components that make up the website
- Each page of the website (about, home, charity model, country model, news/events model, instances) has its own component that is called from index.js to render
- Key Components:
    - GenericModelPage.js
        - This component defines the general structure of each model page (i.e. title, search bar, instance grid)
        - This component is called from each specific model page component (CountryModelPage, CharityModelPage, NewsEventsModelPage) with props that include the title of the

        model, the card component that should be displayed for each instance of the model, and the json formatted data associated with the instances of the model
    - We designed the components associated with model pages in this way in order to allow for more code/component reuse
- About.js
    - To use the GItlab API, we used several asynchronous functions to get the user commits, user issues, and total commits/issues
    - Utilized several public Gitlab endpoints (documentation on their website) to gather this information - called these endpoints within the asynchronous functions
    - Utilized useState and useEffect to update the commits/issues info and render the component when a change is made
    - Organized all this information into various Cards using a generic TeamCard.js component and a TeamInfo.js file

src/model_data
- Directory contains the json files that stores the scraped data for instances of each model
- This was necessary for Phase 1 as a database has not yet been setup

src/media
- Contains the downloaded images that are used for the carousel (slideshow) on the home/splash page

Backend Architecture

backend/data_scraping Directory:

data_scraping/models_data:
- Contains the json files where the scraped data was stored
- This was required for Phase 1 since a database has not been created
- Most likely will be removed upon the integration of a SQL database

country_scrape.py:
- Python script used to scrape data for the countries model
- Data scraped using the UNHCR refugee data API
- Packages used:
    - requests: Used to make HTTP requests
    - json: Used to convert python dictionary into json file

- Since this phase required 3 instances be collected, we hardcoded the country codes to scrape (this will change when a database in installed and we are attempting to collect more instances)

charity_scrape.ipynb:

- Python script and functions used to scrape data for the charities model
- Data scraped using the Wikidata API
- Packages used:
    - requests: Used to make HTTP requests
    - json: Used to convert python dictionary into json file
    - copy: Used to make deep copy of each python dictionary of instance data
- Various functions were written in order to transform Wikidata API responses into more useable data
    - For example, for some attributes that the Wikidata API returns (such as the headquarters location of a charity), the response is not plain text, but the Wikidata page ID
    - Therefore, the get_page_title() function accepts as page ID as an argument and returns the title of the page that this ID refers to
- Since this phase required 3 instances be collected, we hardcoded the charity names we wanted to scrape (this will also change in subsequent phases when a database is implemented)

news_scrape.py:

- Python script used to scrape data for the new/events model
- Data scraped using the ReliefWeb APi
- Packages used:
    - requests: Used to make HTTP requests
    - json: Used to convert python dictionary into json file

Challenges

1. Scraping data from Wikipedia pages:

Since Wikipedia is a collaborative encyclopedia and allows edits to pages and content from its user, this makes the pages inherently un-uniform. For example, while the Wikipedia page for the United Nations Children's Fund (UNICEF) contains a section on "Creation" which describes the founding of UNICEF and the year it was founded, the Wikipedia page for the World Food Programme (another United Nations organization) includes information on its founding under the section titled "History". These small discrepancies between section titles makes any effort to scrape specific data using the Wikipedia API from multiple pages very difficult and time consuming. We resolved this issue by instead using the RESTful API provided by Wikidata. Wikidata is an effort started by the parent organization of Wikipedia and as the site explains, it was created to “[act] as central storage for the **structured data** of its Wikimedia sister projects including Wikipedia, Wikivoyage, Wiktionary, Wikisource, and others” (Wikidata). Therefore, we found that Wikidata provided a much easier interface to gather the structured data that we needed for various charities and organizations. For this phase, we only used Wikidata to gather data for our charities model, however, we feel that this source could be very useful to gather supplementary information for our other models as well.

2. Learning and setting up a React application

We were all pretty inexperienced with using React as a frontend framework, so there was a lot of learning involved in this phase. Also, starting a web application from scratch and getting everything set up was also a tedious process which required a lot of time and effort. However, through various online resources, we were all able to learn the required tools and contribute to multiple parts of the project.