Phase 2 Technical Report

Website Link

Group 06 Members: Aryan Samal, Rohan Damani, Will Matherne, Jeremy Nguyen, Kenny Nguyen

1. Introduction

Purpose and Motivation

PalestineWatch aims to raise awareness regarding the state of Palestine as well as inform others on what they can do to help out. We include a number of support groups that are currently aiding Palestine, countries that serve as places of asylum for Palestinian Refugees, and news articles documenting current events in Palestine.

This report serves as a technical background for anyone wishing to understand the tools/technologies used to build our website.

Questions We Answer

- What is the latest news regarding this conflict?
- What are people doing to help?
- What can you do to help?

2. Models and Instances

Model Overview

For this phase, each model has three instances. Every instance connects to 2 instances of different models.

- **News**
    - Represents articles that are related to the Palestine-Israeli conflict. It may include articles about other countries involved in the conflict.
        - **id:** Unique id of this DB entry.

        - **author:** Creator of the article.
        - **title:** Article title.
        - **description:** Represents a description of the article.
        - **url:** Link to the actual article.
        - **url_image:** Link to the article’s image.
        - **publish_date:** Date of which the article was published.
        - **content:** Represents the full content of the article.
        - **source:** Source from which the article was published by.
        - **supportGroupId:** Id for linked support group page.
        - **countryId:** Id for linked country page.
- **Countries**
    - Represents countries that are supporting Palestine by allowing Palestinian refugees to seek asylum.
        - **id:** Unique id of this DB entry.
        - **common_name:** Name of the country displayed.
        - **official_name:** Official name of the country displayed.
        - **unMembership:** Yes for UN member country, no otherwise.
        - **maps:** Url to google map location of the country.
        - **coa_iso:** Three letter country code.
        - **flag_url:** Url to a picture of the flag of the country.
        - **capital:** Capital city of the country.
        - **population:** # of people living in this country.
        - **region:** The overall region in which the country is located.
        - **subregion:** Smaller, more specific region the country is located in.
        - **newsId:** Id for linked news page.
        - **supportGroupId:** Id for linked support group page.
- **Support Groups**
    - Represents support groups that help Palestine by providing donations, supplies, etc.
        - **id:** Unique id of this DB entry.

- **name:** Support group name.
- **email:** Email of the support group.
- **city:** City the group operates out of.
- **state:** State the group operated out of.
- **zipcode:** Zip Code of the support group.
- **link:** Support group's website link.
- **url_image:** Link to the support group's image.
- **newsId:** Id for linked news page.
- **countryId:** Id for linked country page.

3. API Documentation

https://documenter.getpostman.com/view/38731121/2sAXxLBDwn

4. Toolchain

The main tools we used were Bootstrap-React for styling components, Axios for fetching data from APIs, Next.js for creating our frontend interface, Docker for creating a consistent development environment, and Cheerio for parsing/manipulating HTML. For the backend, we use Flask as our backend framework, Selenium + BeatifulSoup4 for web scraping, and SQLAlchemy as our ORM. Additionally, we used Flask CORS to facilitate Cross-Origin Resource Sharing and Flask Migrate to update the PostgreSQL DB from the backend code.

5. Frontend, Backend, Database Hosting

The frontend portion of the website is hosted on AWS Amplify. We chose this provider because of its "easy to start, easy to scale" design and architecture.

The backend portion of the website is hosted on an AWS EC2 instance. We used Flask as the backend framework along with SQLAlchemy as our Object Relational Mapper. We also used Gunicorn which is a Python WSGI (Web

Server Gateway Interface) HTTP server that can handle concurrent requests. We also use Nginx as a reverse proxy to process Gunicorn requests.

The database we used is a PostgreSQL database that is hosted on AWS RDS. We use pgadmin4, the PostgreSQL workbench, to create tables and do other db tasks.

6. Architecture

The site is primarily divided into pages that are built with individual components. We have a pages folder where all the pages can be found as well as a components folder to access all of the components.

The first page the user is greeted with is the splash page, where they can then navigate to a multitude of other pages using the Navigation Bar at the top of the page. The splash page also contains some history about the Palestine-Israel conflict as well as our motivation for creating the site. The About page has information related to all of our group members, including bios, number of commits, pictures of us, as well as some general project information.

The next 3 nav bar items are the News, Support Groups, and Countries tabs. They each link to a page that houses the 3 instances of each particular model. Each model card displays a variety of information and is also clickable, allowing you to display more information. When you click on a card, it will navigate you to a page named [id].js in either the countries, News Pages, or support-groups folders. This will then allow you to view a full page of information on the given topic.

7. Challenges

A major challenge our group faced was related to acquiring data using APIs. Although we did vet the API in the RFP stage, we did not realize that some of the APIs would have hidden flaws. For example, many of the news APIs had a content paywall that would not allow you to access the full article content as a JSON object. This was quite frustrating at first, but we managed to make it work. Another major challenge we faced was deploying the website on AWS Amplify. We had a lot of issues where it said the website was deployed, but when we tried to click on the deployment link, it would return a 404 error saying the website was not found. This was due to an error within the pathing configuration that prevented the index.js file from being found.

8. Searching and Sorting

Our search algorithm is implemented primarily through the backend. We were able to simply modify the “getAll” endpoints to account for a search query word. The sorting algorithm is done mainly through the frontend, where, depending on the attribute, it sorts the array of data.

9. User Stories

Most of the user stories for this phase centered around fixing small issues like broken links and placeholder pictures, as well as fixing our search in the News model, which happened to be broken when they were testing the site.