Create a pseudo database via Google Sheets.  
By publishing the sheet and setting its sharing permissions to “Anyone with the link can view,” we made the data readable by external tools.

Use a service opensheet.elk.sh to convert that Google Sheet into a JSON API, which can be fetched directly from a website.

Create html document that uses opensheet.elk.sh API to load data from google sheet and then display in whatever format is desired

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

URL of google sheet with information: <https://docs.google.com/spreadsheets/d/1kvs0UEPlUaPICdJ3uGFV1Jxm7O_Q8OjgY6R2WC5tpQ8/edit?gid=0#gid=0>

You extract the **Sheet ID** from the /d/.../ part:

Sheet ID: [1kvs0UEPlUaPICdJ3uGFV1Jxm7O\_Q8OjgY6R2WC5tpQ8](https://docs.google.com/spreadsheets/d/1kvs0UEPlUaPICdJ3uGFV1Jxm7O_Q8OjgY6R2WC5tpQ8/edit?gid=0#gid=0)

Sheet name: Organizations

Sheet must be shared publicly to allow it to be readable by opensheet.elk.sh.

Go to the Google Sheet → click **Share** →  
Set access to: **“Anyone with the link” → Viewer**

https://opensheet.elk.sh/[1kvs0UEPlUaPICdJ3uGFV1Jxm7O\_Q8OjgY6R2WC5tpQ8](https://docs.google.com/spreadsheets/d/1kvs0UEPlUaPICdJ3uGFV1Jxm7O_Q8OjgY6R2WC5tpQ8/edit?gid=0#gid=0)/Organizations

Ensure that sheet is shared with “anyone with the link”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Geocoder script is run on the sheet to get a latitude and longitude for the given address, This is needed to display the pins on the map. Can run the script like any normal sheets function. Note, OpenCage, the geocoding API has a **daily limit (2,500 requests/day).** The full script can be viewed via Extensions Apps Scripts.; Haven’t put in exception for if API fails, will do this later.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mapping: Leaflet is a Javascript library that is open source and uses openstreetmaps which is also open source (No tracking, no ads, and built by a community — aligns with mutual aid/grassroots values). It can integrate well with static websites which is important to us because we wish to use github pages which only allows for static sites. It is able to get data from a google sheet relatively easily also. It is fully customizable and lightweight.

Using leaflet (lightweight JS library for building interactive maps)

* Initialize map (defining where it centers on, extent of zoom), add map background via OpenStreetMap, add markers (can also be popups etc)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Calendar: Fullcalendar is a javascript library that allows for customization of appearance, filtering of events, is mobile friendly, and allows the use of google sheets as a source of data, It is opensource in that it is freely available under the MIT license but does have a paid version also. We would not be able to set recurring events but this seems to be the only drawback i’ve found thus far. I think overall it would be pragmatic for our purposes. If we are interested in getting the paid version its $89 per dev per year with options for non profits but i doubt we’ll ever feel the need to.

Wonky shit with the calendar - needs ISO 8601 date format ie year month day which means i’ve created another row to convert it to the correct format so the calendar reads it properly.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Google Cloud Platform (GCP)** is Google’s collection of tools and services for building software in the cloud. - allows us to interact with google sheets programmatically

| Thing | What It Does | Why You're Using It |
| --- | --- | --- |
| Google Cloud Project | A workspace to group your APIs, credentials, and access settings | To isolate and manage your bot/scraper logic |
| Google Sheets API | Lets programs (not people) read/write spreadsheet data | So your Python script can update your event sheet |
| Service Account | A secure, non-human "user" that can access APIs | Your script logs in as this bot to access your spreadsheet |
| JSON Key File | A secure credential your script uses to authenticate | So your script can act as the service account without human login |