For the news apps government website upgrade project, I wrote a tracker for University of Maryland Police Department activity and arrest logs. Their current site lists activity and charges separately from arrests, even though both are catalogued by a case number. Furthermore, their site has flat tables with no searching or sorting capabilities, and activity is broken up month by month, year by year.

To make this site more accessible and data-dive-ready, I first scraped data from the original using Beautiful Soup in Python. The two page scrapers accessed the table element on each page. They both had to alternate rows for starting a new case object because description was in a cell beneath the rest of the info on that case. I used datetime to check today’s date and ping the webpage connected to the current month and year in order to get up-to-date data. I looped through the entire range of available months and years once to collect the site’s data backlog for both arrests and activity, then I connected the scrapers to GitHub Actions to run every day at 8 a.m.

I wrote a simple Flask app to display the new activity log, and I created links to detail pages where there was arrest information. I reformatted the date so that the datatables package would sort dates properly, albeit in a less user-friendly way. I also froze the app to run on GitHub Pages.

The problem with this set-up is I couldn’t figure out how to deploy Flask through GitHub Actions. If I freeze the page, old data is baked in. Especially once I linked in legacy data, my little laptop had a hard time freezing the entire app with its thousands of detail pages. I could put the freeze task into a GitHub Actions itself and let the git virtual machine deal with it at 8 a.m. every morning. This is a task I didn’t get to.

I also wanted to include a time-since dashboard at the top of the page, but I never converted my dates to Date objects, so finding the difference between today and every day present for each of many crime categories would be (even more) hacky. I’m not super well-versed in pandas.

I ran into a data problem where an activity line has an Arrest disposition, but there’s no arrest case to match the activity case number. I saw this in very recent data, something from this past week, so my guess is that that data is added in later when the officer remembers. In any case, it broke my app, so now there’s a catch for if no cases come up in arrest\_list despite the disposition.

It would have been best if I could display one human-legible date in my table, but sort by a more algorithm-legible value in datatables. I don’t think the package discriminates that closely, though, and likely sorts based on the current surface-held text values, not by object property values passed in at creation.

Hana noted that CSA logs would be good to include, but the data for those is different and would need its own page – which wouldn’t be too hard to build out a template for. It would largely reuse the code I’ve already written, and I could add a navbar in the top corner to switch between the two pages.

I’m not very good at seeking out pre-made packages to help me with projects. I’m sure there’s a beautiful timeline-crafter-package available, but it seems a hassle to learn new syntax for every module included in a project. I guess the point is that you learn it, then reuse it in projects ad infinitum.

I may put in more work on this over spring break, just to flush out some of these easier-than-thought tasks. I’ve already worked through the bot portion of the website, so it would be cheating to use this as my bot project. I didn’t make any wireframes or sketches to go with this.