Interactive Photo Archive Website

Judson A. Hartley

Fort Hays State University

Course Number: INF651

Author Note

Email: j_hartley@mail.fhsu.edu or j@jahartley.com

Abstract

I run a computer at home that acts as my personal server. It is my data backup, family share drive, and my 24/7 host for a small range of software that I enjoy having at my fingertips. Over time, I have amassed a photo collection of all of my wife's and my photos, some of her surviving family photos from her youth, my families' photos and now a steadily increasing collection of photos of my kids as they grow up. Right now, the total sits at 21,943 photos over 60 folders, taking 228 GB of space. I have a project to digitally archive all my families' photos, and I bet that as time goes on this number will steadily increase. Although I understand my folder layout, there is almost no chance that my wife's or my family will be able to browse through them in a meaningful way. The goal then, is an interactive photo website, that has a few different ways of browsing the photos. As this is meant to be an archive, the main way to browse is via the date the photos are taken. For this project, I would like to implement searching by keyword and by person, along with a date range, as the base implementation.

Interactive Photo Archive Website

The following will outline will outline my work and challenges faced trying to create this website.

Website Functionality

At this time, the website is not fully functional. I have been implementing the details in vanilla JavaScript, HTML and CSS. There was a surprising number of pitfalls trying to make the site function as I intended. I have some backend in JavaScript for NodeJs that catalogs the EXIF data into a CouchDb database for use with this project. CouchDb provides native JSON results, which I have simulated with the photos.js and photos2.js JSON data. The structure for allowing a user to refine the results is in place, but among the details that are not finished include the search pages, and the shortcuts via clicking the people and tags that can be displayed as you scroll through the photos.

Website Structure, Features, Pages & Content Status.

List of pages:

1. Login page. Basic username and password login. This page is not yet implemented, but would load like the information and message modals, before the page begins loading any photos or data. I have implemented a JSON Web Token authorization backend in the past, and that is the plan for this feature. These features will be running on my personal server at my house, and I didn't want to take a chance on self-hosting this specific project.
2. Home page. Landing page after login, will have a menu to the search and about pages, and a random photo. This page has been merged with the view photos page, as the whole point was to see the photos. A specific homepage before clicking to the photos seems redundant.
3. Search page. This page will let users search for sets of photos, initially by date, and with by people and keywords is planned. The search pages menu is implemented, and the search modals for the different types are yet to be implemented. The ground work for selecting a subset of the total index, which I would normally run in the backend leveraging CouchDb's B-tree and map/reduce query system, has

been ruffly approximated over the sample set of pictures at the beginning of the main.js file. Searching by date would simply load the photo whose date is closest to the chosen date, as the photo viewing section can infinitely scroll forwards or back in time. This same date functionality would also be implemented for selecting the first photo shown from any search, with photos matching the search criteria streaming forward and backwards in time.

4. View photos page. Main scrolling content page, lets users scroll through the photos and click for full screen. This section is functioning as expected. I have used snap scrolling, lazy loading, and reactive element css to try and make the experience as streamlined as possible. Getting new image div's prepended to the top before you can scroll them into view was a nightmare of convoluted JavaScript fighting CSS. There was hundreds of iterations try to make this part work right. Currently the JavaScript removes snap scrolling, adds new div elements, while scrolling the view back to the photo you were looking at, for each going back in time load of five new elements, then reenabling snap scrolling, all while trying to keep the view from flopping back and forth.

5. About page. Basic page information, and photo database statistics. This page is low on the priority list as information here would be mostly for my own use.

6. New photos page. This page will have all newly added photos so that the user can verify the photo's information before it is sorted in the database. This page would just use the same page as the viewer, with the in use section set to new photos without a proper database id.

Conclusion & Future Plans

This proposal is for a family photo archive so that family can view and search through the photos that I have archived. The piecemeal way that the HTML/CSS/JavaScript model has grown has seriously dampened the speed at which a project like this can be created without using generators or large frameworks.