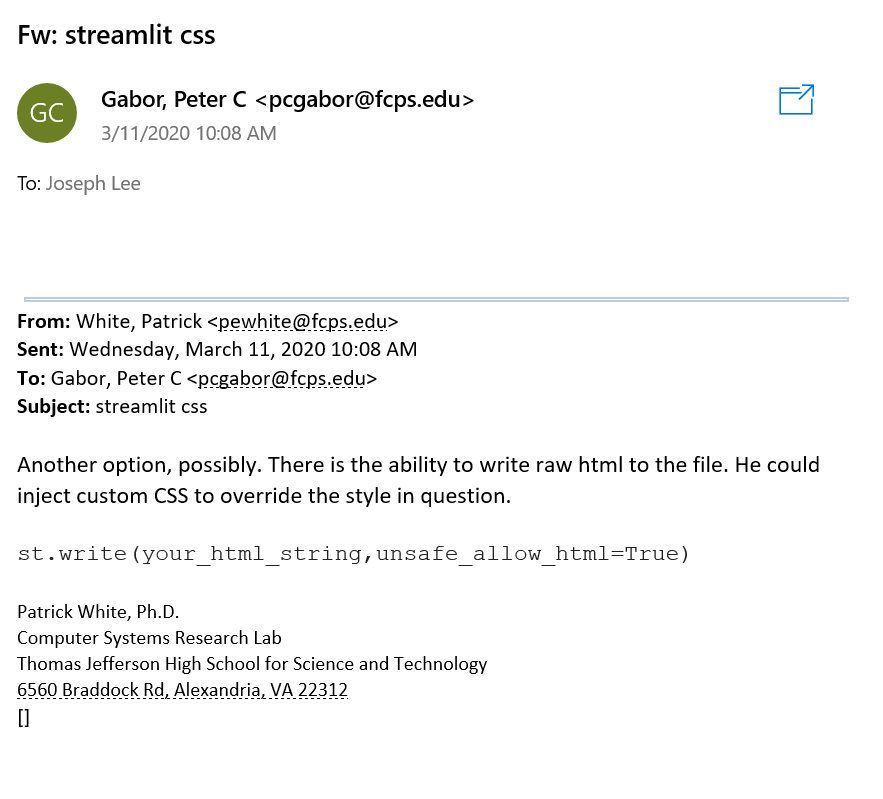
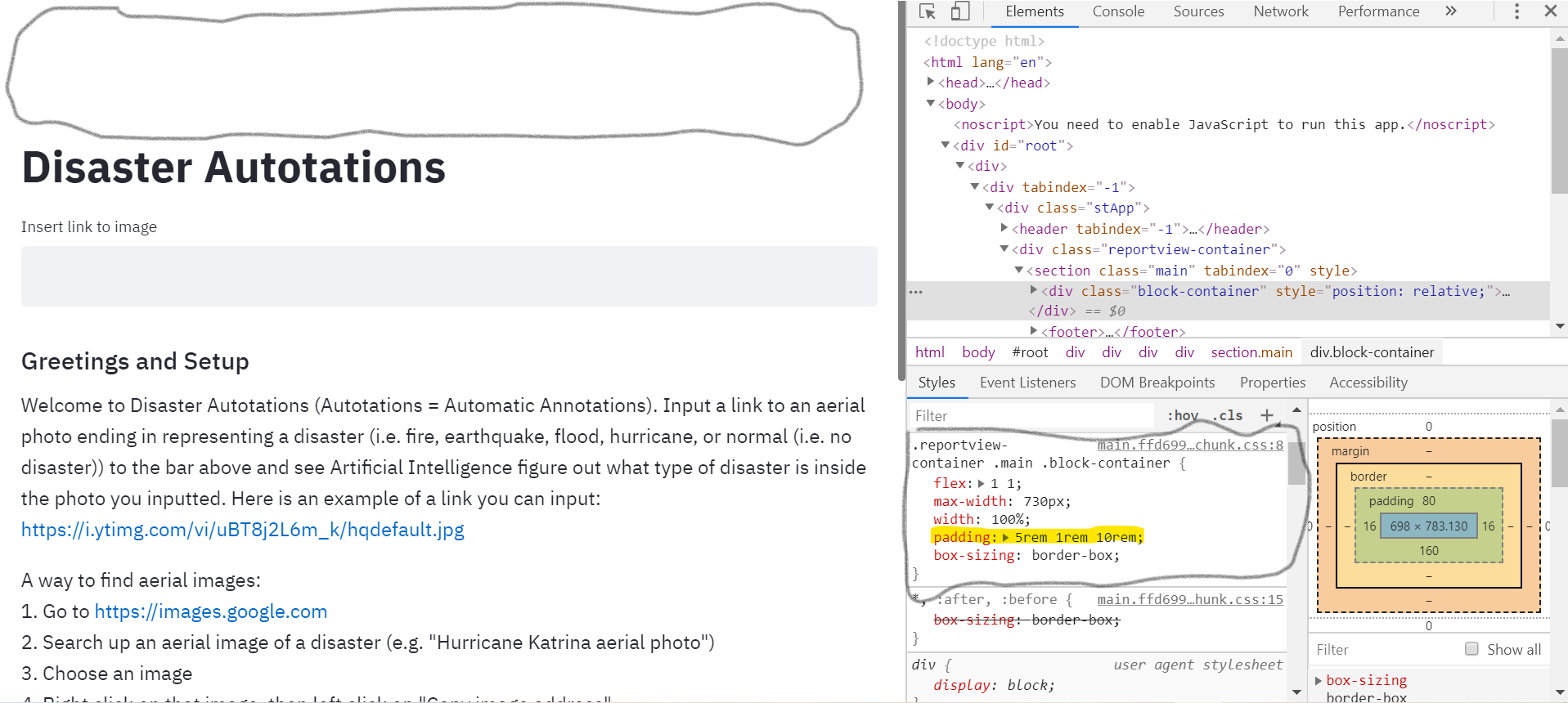
**Saturday 3/21/2020**

I finally figured out how to get rid of the excessive white space at the top of my web application (streamlit2.sites.tjhsst.edu.) The following email was of great help:



Dr. Gabor investigated that the part of .css file that handled the excessive white space at the top of my web page was:



The highlighted part was the part I needed to modify. Apologies for the screenshot not being clear, but close-up, we have:

.reportview-

container .main .block-container {

flex: 1 1;

max-width: 730px;

width: 100%;

padding: 5rem 1rem 10rem;

box-sizing: border-box;

The way I overrode the padding was by the following line of code:

st.write("<style> .reportview-container .main .block-container {padding: 0} </style>", unsafe\_allow\_html = True)

A large part of what helped me figure out this line of the code was another section of our “Autotations.py” code where I had gotten rid of the hamburger menu at the top right corner of our webpage:

hide\_menu\_style = """

<style>

#MainMenu {visibility: hidden;}

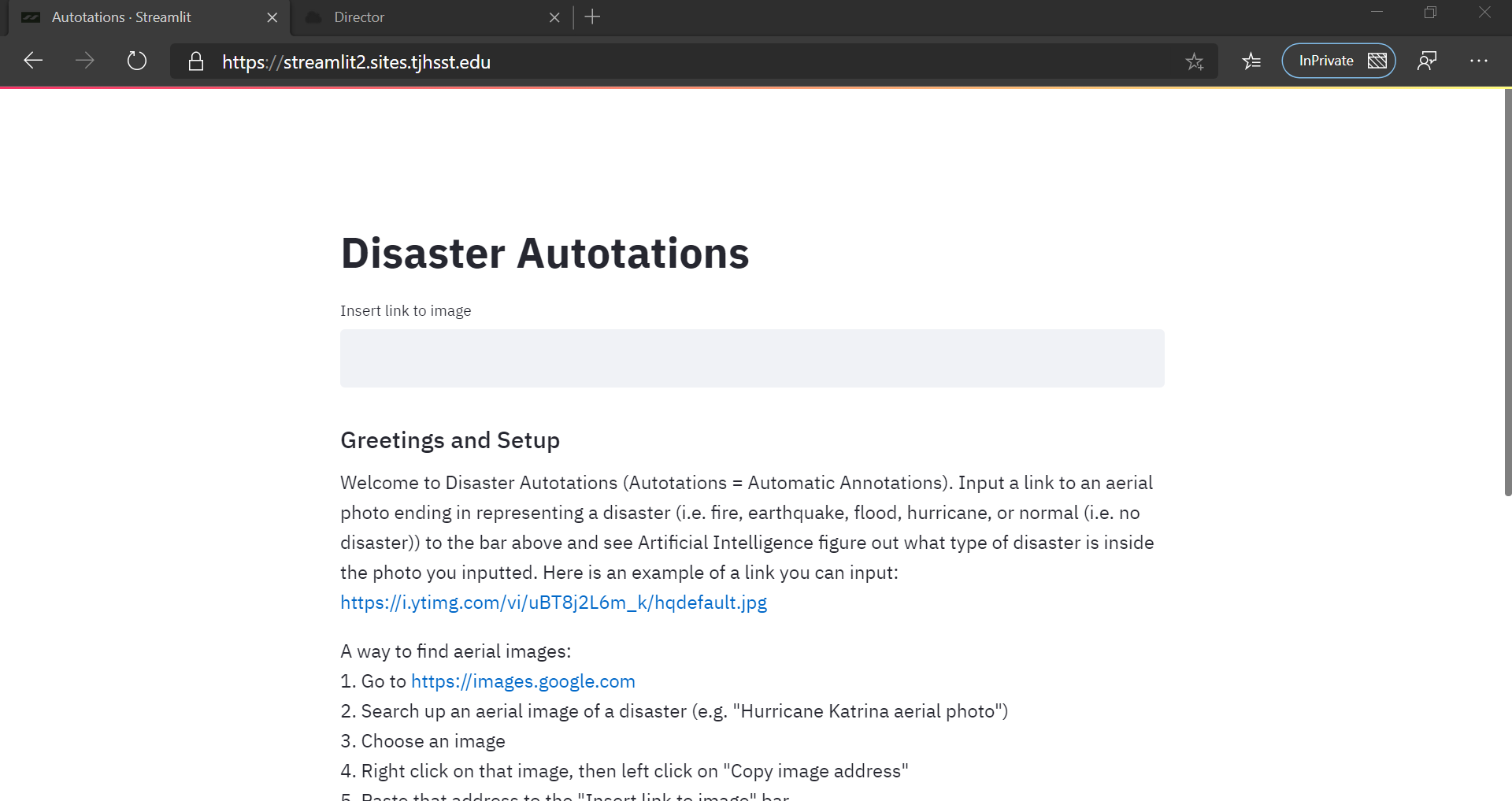
</style>

"""

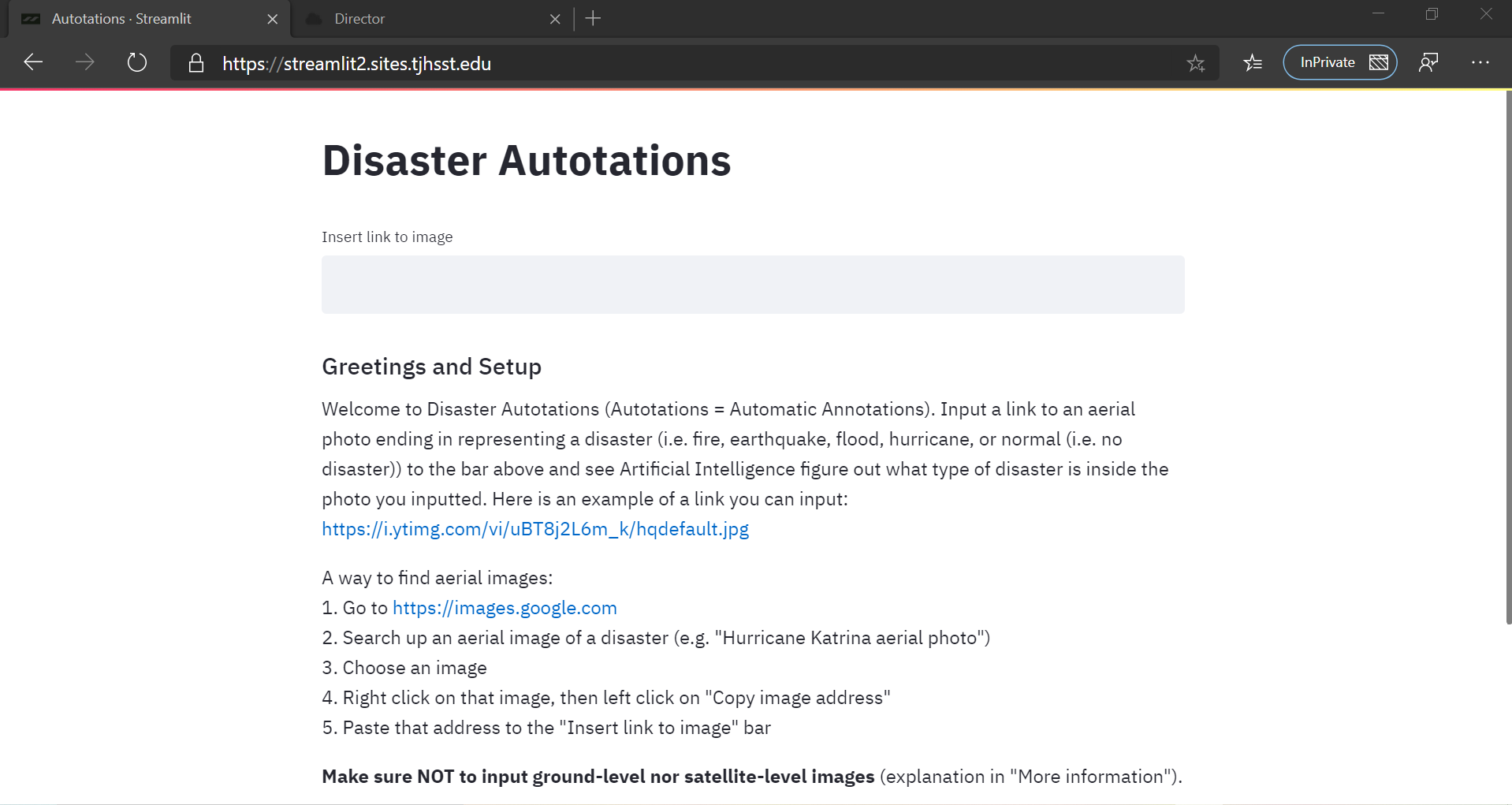
st.markdown(hide\_menu\_style, unsafe\_allow\_html = True)

I did not know one could override a CSS rule set by simply writing down the rule set again and inserting properties he or she wants to replace with. I had no idea that overruling a rule set could be simple. In addition, I did not know this could all be done within the <style> tags either.

Before:



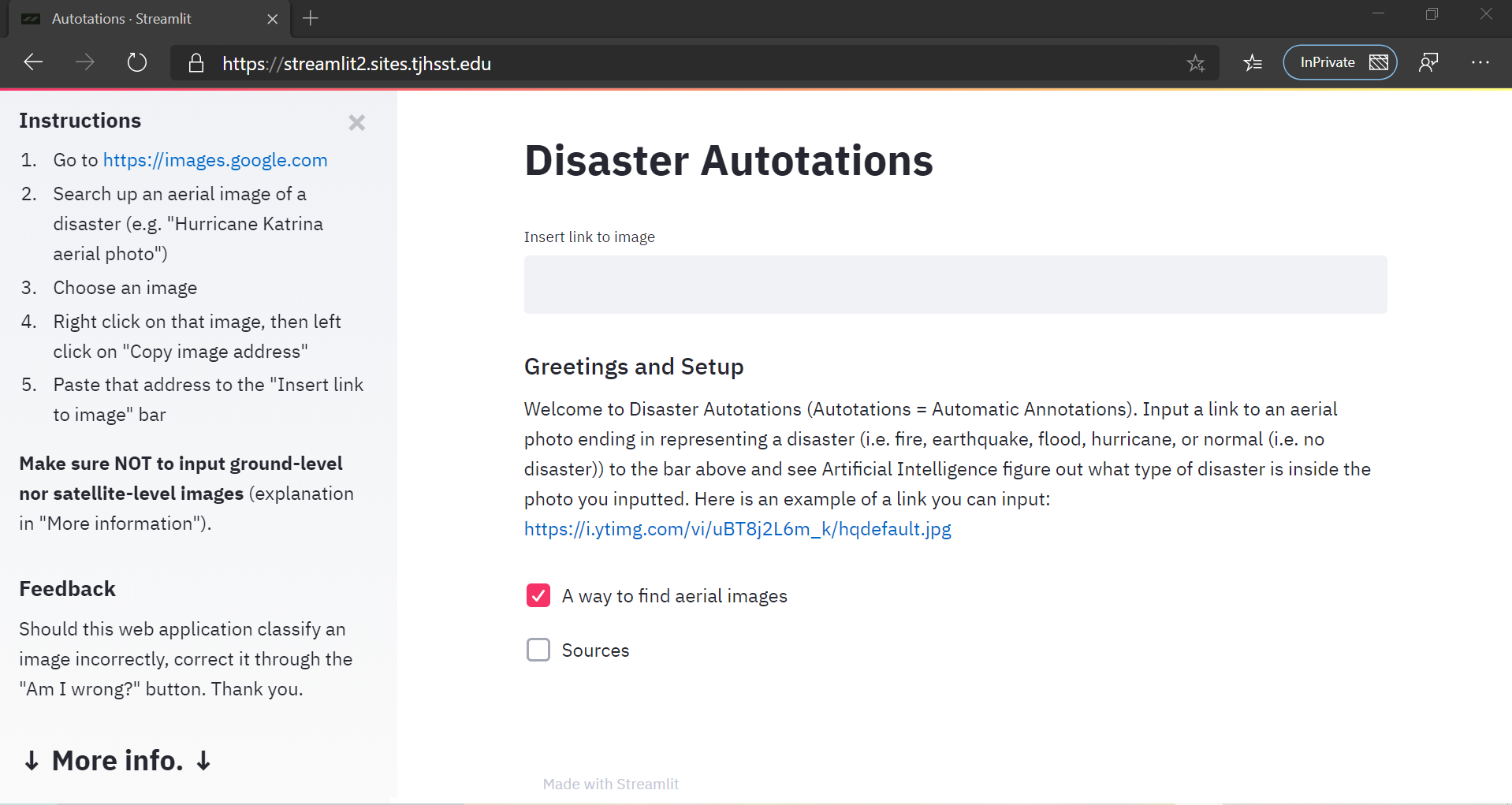
After:



The next task I wanted to accomplish to better my user experience was to address the comment of the navigation to another web page rather than keeping all the information in a checkbox.

After googling, I could not find a way to do it, unfortunately. However, I thought I could probably display a sidebar once a user clicks on the “More info.” checkbox rather than displaying more information all the way down to the web page. Of course, this was my partner’s original idea, and I decided I would go with it. Recall that I had not supported his idea earlier because I did not like how small the sidebar was. However, when I also considered the fact that I wanted my users to enjoy my web page without the need to scroll (thanks to Dr. Gabor who made that suggestion in the first place), I finally decided that the sidebar was a beneficial idea after all, so that users could read the instructions on copying/pasting links without having to scroll down and therefore use the main functionality of the web page.

I accomplished that idea, and for more compactness, I also migrated the “A way to find aerial images: [step-by-step instructions]” into the sidebar as well considering the fact that it is in my best interest to minimize scrolling for my users.



To get rid of the white space on top on the sidebar (as I got rid of in the above screenshot), I used:

st.write("<style> .sidebar .sidebar-content {padding: 0rem 1rem} </style>", unsafe\_allow\_html = True). I wanted to ensure that “↓More info.↓” was visible on the screen.

**Monday 3/23/2020**

Today, I worked on getting through my share of 250 annotations. It was incredibly challenging to stay motivated and keep up with the annotations after 30 or so photos, as I learned today that school was canceled for the rest of the school year. Many of the peers I knew and my friends made posts on Facebook sharing all the memories we had together, and I was heartwarmed and sad that I would not be able to see them again anytime soon. However, I knew we could reunite together as alumni.

I tried my best to not let the disappointing news affect my research. However, as I tried progressing along with the annotations, I just could not stop thinking about it. I took breaks time to time to message the people I appreciated studying with while at TJ.

I reflected back on my TJ experience, thinking of all the beautiful memories, but what bothered me often were the feelings of impostor syndrome I had and my lack of having many friends to talk to. I had many peers I was comfortable talking with, but I would say only a select few are truly what I would consider my friends. While I did wish for more friends, I was content with the few friends I did meet at TJ, and one of my closest friends was actually a guy I met in middle school. I also realized that even if I did not have many friends, it would still be nice to take the time and message people who I have had interactions with, no matter small or big, on Facebook messenger. So, that was exactly what I did.

It took me a while (approximately two hours) until I finished the first 100 annotations (which were all earthquake aerial photos). But, I was glad to celebrate that first benchmark and proceeded on with the hurricane annotations (another 100.) Then, I finished 40 (give or take a couple) more photo annotations, which were all hurricane aerial photos.

**Wednesday 3/25/2020**

On this day, I wrapped up the rest of the hurricane aerial photo annotations, took a break by walking outside, enjoying the cool weather despite the fact that I had to deal with wet ground since it had rained earlier today. Afterwards, I finished the remaining 50 annotations I had to, and they were all flooding aerial photos.

Then, I was stuck on one part of the tutorial. That part was:

np.random.seed(42) # set random seed so we always get the same validation set

src = (ImageItemList.from\_csv(path, 'train\_v2.csv', folder='train-jpg', suffix='.jpg')

# Load data from csv

.random\_split\_by\_pct(0.2)

# split data into training and validation set (20% validation)

.label\_from\_df(label\_delim=' ')

# label data using the tags column (second column is default)

)

When I implemented my own version of this, I figured out that “ImageItemList” was no longer the class name to use, and instead it had transferred to “ImageList” instead. However, I realized I also forgot to tune “train\_v2.csv” to my own “.csv” file.

However, I was like “Hmm? What happened to the dataframe with the tags column?” Then I realized “Ohh, right, the dataframe already inherently had the tags column, and he was just using the .csv file instead.” I looked up the fastai documentation to see if there was a method something along the lines of “ImageList.from\_df.” It turned out there was. However, I had to spend around an hour to understand the general gist of what the method did, since I also had to look around the source code for variable names and NumPy methods I did not understand (i.e. “np.char.add.”) I then decided it was probably best for me to conform to the tutorial style so I would not have to overcomplicate things for myself.

I had to understand what “path” represented. So, I looked into the fastai documentation on “ImageList.from\_csv,” and I figured out that “path” represented the common directory where both the annotations file and the folder to the images were located. I had to reformat my annotations file since the one on the tutorial just had a “image\_name” and “tags” columns, while mine had both of those columns plus 1s and 0s scattered across. Instead of getting rid of the 1s and 0s on my google spreadsheet, I decided to read into that file by pandas.read\_csv(‘multi\_annotations.csv’) and concatted just the “photo\_ID” (the equivalent of “image\_name”) and the “tags” columns to make a new dataframe that I could convert to a .csv file format as shown in the tutorial. I used the .to\_csv method to accomplish that task. The .csv file was then visible on my Google Drive, and I then migrated that file to the to the “dataset” folder (renamed from “images” previously since I thought that name suited better to the what the folder actually contained (which was now both the annotations file and the folder of images.) Then, I proceeded along with the tutorial just fine, until I got confused with: “The only difference is that this time we will perform this process twice. Once for the 128x128px images and then with the 256x256px images.” “What 128x128px and 256x256px images?” I asked myself. I checked earlier in the tutorial to see if I had missed anything, and I did. The tutorial mentioned: “Now that we have an ImageItemList we just need to apply our transforms, convert our data in a databunch object and normalize it. We didn’t do the whole process in a single step because we will first of train on a smaller image size — 128x128px — and then switch to the original size, which is 256x256px.”

“Hmm, that is very interesting” I thought. In addition to all this, I realized resizing my images to 500x500px had been pointless all along since the tutorial did the resizing automatically. “Oh well, but I suppose it was nice seeing the standardized size while I was simultaneously annotating and inspecting photos closely. Standardization made sure that no images popped too big or too small--just right.” So, even though standardization on our part (to 500x500px) did not serve the purpose I thought it would, it had a nice indirect purpose after all.

It was getting very late, and I headed off to bed.