Project Description [2.5 pts]: The name of the term project and a short description of what it will be.

Comic Sans Artist: an application where you can build your own comic out of frames of yourself. There will be options to customize it using comic graphics and different image filters, as well as an option to auto-generate a comic from a recording of yourself.

Competitive Analysis [2.5 pts]: A 1-2 paragraph analysis of similar projects you've seen online, and how your project will be similar or different to those.

One similar project I found was Ayesha Gupta’s FilterLab(s20), which is an image editor. My project will be similar in the sense that it allows you to apply different image filters, largely inspired by opencv functionalities. Though I am not sure how she implemented them, I am planning on writing many opencv functionalities by hand for algorithmic complexity, which I believe will make my project different. Additionally, since my project is based around the idea of generating a comic, the filters will be comic-themed, and there will be customization/graphics drag-and-dropping options that her application does not have.

My project is similar to an online application(not 112-related) called comixify, which takes in a video or image input of a person and return an AI-generated comic with AI-generated text. My video feature is fairly similar to this, but my project will be different in its further customization opportunities of graphics options that will make my application more of a studio-artistic setting. Also, I don’t think I will focus too much on trying to generate non-random text.

Structural Plan [2.5 pts]: A structural plan for how the finalized project will be organized in different functions, files and/or objects.

I currently have objects for my different button types(which inherit a Button class), frames and different Region objects that extend from my Region class. This will give me functionality to contain different regions/objects and enable drag-and-drop customization functionalities. These are in my widgets.py class, and in my main.py, I have a StudioApp that runs on cmu\_112\_graphics that manages the entire program. As main gets larger, I will likely make filter objects in another file that can be applied to images in order to separate things out. Also, I may separate my opencv interface into another file that deals with taking photos/videos and applying real-time filters on its own.

Algorithmic Plan [2.5 pts]: A detailed algorithmic plan for how you will approach the trickiest part of the project. Be sure to clearly highlight which part(s) of your project are algorithmically most complex, and include details of the algorithm(s) you are using in those cases.

The trickiest part should be rewriting opencv filtering and editing capabilities by hand. Specifically, the most difficult algorithms will involve edge detection and adaptive thresholding.

Edge detection: inputs: sigma(size of kernel for gaussian blur), low and high thresholds(as a measure of gradient)

1. I will convolve a gaussian kernel across my image in order to blur it to remove noise for edge detection.
2. Apply the sobel kernel to the image to get the approximated gradients at each pixel.
3. Continue with Canny edge detection by setting the above-threshold gradients that are not surrounded by any other above-threshold pixels to below-low threshold.
4. Eliminate pixels below the low threshold and maintain pixels above the high threshold.
5. Eliminate pixels between low and high if they are not surrounded by high threshold values; otherwise, set to high.

Adaptive Thresholding:

1. Taking the input sigma corresponding to the size of the convolution kernel, we convolve the image to calculating the mean(or gaussian value) of each sigma-sized neighborhood to be our threshold.
2. Using our threshold-matrix, we classify each pixel of the original image as “in” or “out” depending on whether it is higher or lower than the associated threshold matrix value.

Timeline Plan [2.5 pts]: A timeline for when you intend to complete the major features of the project.

Sunday(tp1): implement user interface with drag/drop, adding graphics and clicking text/thought bubbles(with mouth-level automation), segregate the images so that filters can be applied independently of the user interface, opencv or pil form

Tuesday: write opencv algorithms: convolution, gaussian blur, edge detection, pyr up/down, thresholding, erosion

Thursday: implement algorithms into multiple filters: basic cartoon filter, dotted halftone print, dramatic exposure, pastel, background/vignette

Saturday: generate comic from video with random filtering or selection of a filter for each image based on facial recognition, etc with automatic text bubbles, random text for now probably

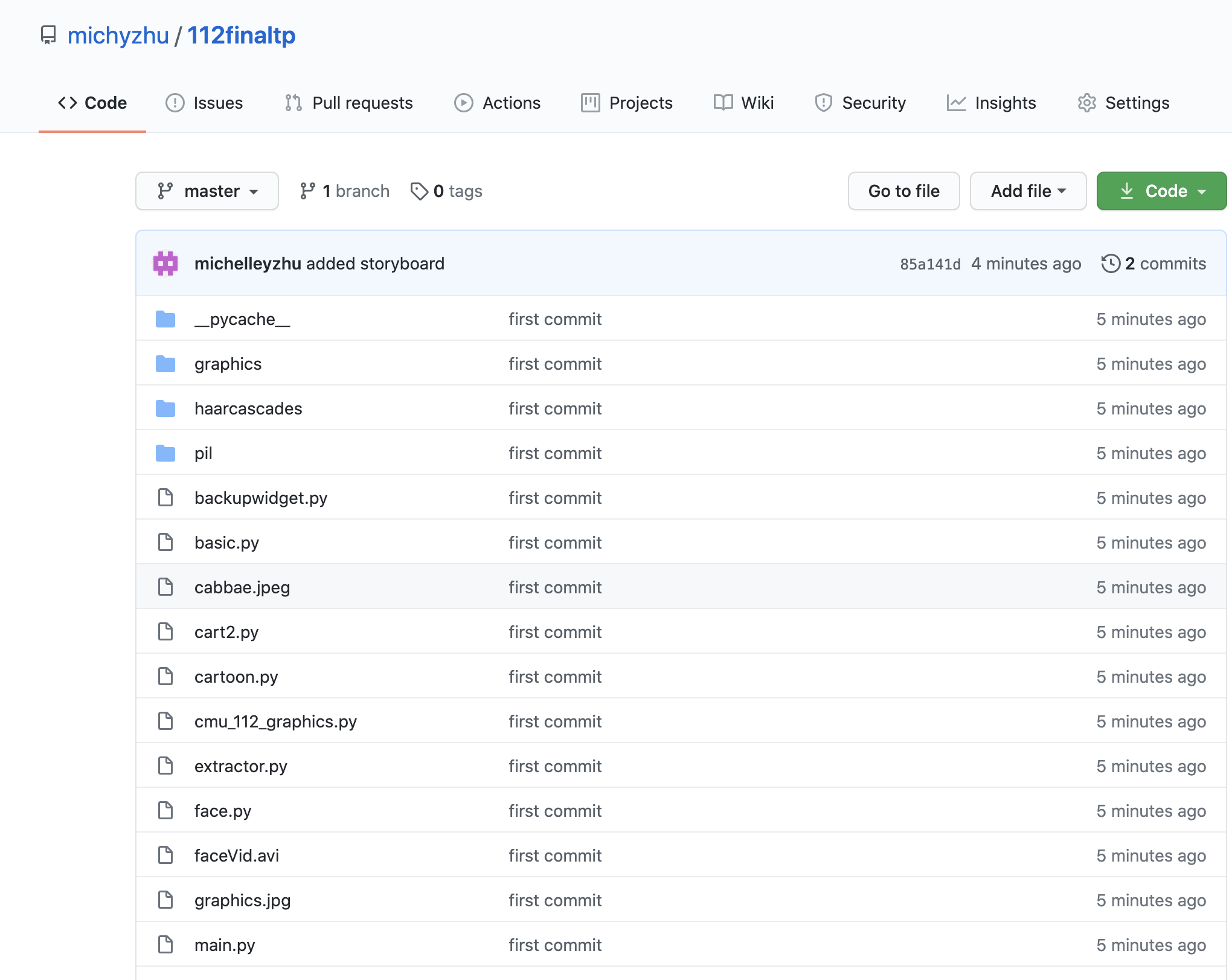
Sunday: hard deadline for this past week’s listed items: meets MVP

Tuesday: continue to improve/expand/complete any filtering that wasn’t good enough/complex enough

rest of the time: improve user interface and either attempt the harder opencv algorithms(bilateral and multi-level thresholding with otsu’s method) or look into emotion detection(past mvp with tensorflow) and automatic placement of emotion-labeled graphics

Version Control Plan [1.5 pts]: A short description and image demonstrating how you are using version control to back up your code. Notes: You must back up your code somehow!!! Your backups must not be on your computer (ideally, store them in the cloud)

**I will be using Github.**



Module List [1 pts]: A list of all external modules/hardware/technologies you are planning to use in your project. Note that any such modules must be approved by a tech demo. If you are not planning to use any additional modules, that's okay, just say so!

opencv, pil(not past what was covered in optional lecture)