Project Proposal

TP3 Update ---

**Project Description:**

Name: Photo Enhancer

Description: Photo Enhancer is a basic photo editing tool that serves to ‘embellish’ your images. It allows users to select an image from their computers or from an URL. Once the user has uploaded the image, he/she can edit it using the tools. The tools include: Rotate, Brightness, Contrast, Filter, Mirror, Crop, Blur, and Sharpen.

**Competitive Analysis:**

There have been many photo editing 112 term projects. Some of them that inspired my idea were Image Editor by Michelle Wan, Image Editor by Teng Zhang, and Photo Twist by Chris Lewis. All of these projects have the basic similar features such as adjusting brightness, contrasts, and adding filters; however, mine will have some additional features such as the ability to crop, sharpen, and blur. Furthermore, my project emphasizes on the usability. Because this is a photo editing tool that is aimed to make your image look aesthetic, I believe that the tool itself must be easy-to-use and navigate.

**Structural Plan:**

1. Main File that is similar to a start screen. This file is responsible for the user interface, such as drawing buttons, labels, and menu. There will not be any functions on this page.
   1. The basic structure was taken from https://sourceforge.net/projects/page/postdownload?source=dlp link (Website to download Page)
   2. I played around with color and formatting to make it look nice.
2. Menu Functions: openClicked(), importClicked, saveClicked, saveasClicked, closeClicked, helpClicked
   1. I have a menu, which is drawn in the main file. The menu functions are responsible for the action following the user’s click.
   2. In this menu, the user can upload an image from his/her computer, click on import using URL, which will allow them to import their URL at the bottom of the screen, save the image, save as image, close the image, and get help.
3. enableEdit(): when the user first opens the tool and there are no pictures to edit yet, I disabled all the buttons on the screen. Although he/she can move the slider, nothing will happen. However, as soon as the user inputs an image, I allowed the buttons to work.
4. Filter: The user is given 4 filter options: Black & White, Red, Green, and Blue. The user first selects the filter option, and then clicks filter for the image to change.
5. Mirror: This mirrors the image across the y-axis. In essence, the right side becomes the left and the left side becomes the right.
6. Crop: user selects a rectangular area, and this crops the image
7. Blur: this blurs the image
8. Sharpen: this sharpens the image
9. Rotate: the user can rotate the image clockwise or counterclockwise
10. Brightness: the user can adjust how bright or dark using the slide. The slider starts off in the middle, which is the original. Moving right will brighten, while moving left will darken.
11. Contrast: the user can adjust the contrast levels similar to that of brightness.

**Algorithmic Plan:**

1. UI Design:
   1. Used PAGE to design, which is a tkinter module for UI design
2. Brightness
   1. If the user wants to change the brightness, he/she first adjust the scale under brightness, and then clicks the Brightness button
   2. brightnessClick function is called when the brightness button is clicked
   3. brightnessClick changes the brightness of the image by going through each pixel (nested For loop)
   4. After getting each pixel, it will look at each of the R,G,B values and increase it by the adjusted value, which is determined by the user on the scale.
   5. However, the R,G,B value only allow for the max to be 255 and min to be 0, so I made sure that the adjusted values are in the range.
3. Contrast:
   1. If the user wants to change the contrast, he/she first adjust the scale under brightness, and then clicks the Contrast button
   2. contrastClick function is called when the contast button is clicked
   3. contrastClick changes contrast of the image by going through each pixel (nested For loop)
   4. After getting each pixel, it will look at each of the R,G,B values and change it by the the formula, which has the adjusted value, which is determined by the user on the scale, embedded. The formula is taken from a source.
   5. However, the R,G,B value only allow for the max to be 255 and min to be 0, so I made sure that the adjusted values are in the range.
4. Rotate:
   1. The user has to first indicate the direction of the rotation (as indicated on the buttons). This will set the direction for the function
   2. Then when the user hits the rotate button, it will call the rotateClicked function. Depending on the direction, the image’s width and height pixels will get switched. (This is similar to the rotate function in Tetris)
5. Mirror:
   1. For loop through the width and create a flipped version
   2. Nest for loop through the height to get the height coordinates.
   3. Set the original pixels to the new flipped version of the pixels.
6. Filter:
   1. The user first indicates the filter that he/she wants by hitting the radio button, which will set the filterColor for the function.
   2. Then, the user hits the filter button, which will call the filterClicked function.
   3. Black & White filter takes the average R,G,B values of all the pixels
   4. The other filters, Red, Green, and Blue is playing with just one of the RGB values.
   5. Red looks at the red pixel – 1st value. Green looks at the green pixel – 2nd value. Blue looks at the blue pixel – 3rd value.
7. Blur:
   1. When the user clicks the Blur button, the blurClicked function will get called.
   2. There are many blur algorithms, and the one that I used is to look at each pixel and its surrounding 8 pixels, so a matrix of 9 pixels. I separately looked at each RGB value, and I averaged the pixel’s RGB values with the pixels of its surroundings.
8. Sharpen:
   1. When the user clicks the Sharpen button, the sharpenClicked function will get called.
   2. The algorithm I used to sharpen starts the same way as my blur algorithm. However, instead of taking the average the pixel and its 8 surrounding neighbors, I took the greatest difference between the pixel and its 4 surrounding neighbors (up, down, left, and right).
   3. I have a helper function that takes the greatest difference
   4. After getting the greatest difference amongst each of the RGB values, I set the new value to the original and adding the difference \*2 (\*2 is how much I want to sharpen by). However, since I am adding, I need to make sure that the new values are still in the RGB values.
9. Crop:
   1. Data.doCrop = 0 if not cropping and data.doCrop = 1 if crop
   2. When user clicks crop button, will call crop function
   3. Then, user can select a rectangle.
   4. The initial click as well as the final release is recorded. I have to make sure that however the user draws the rectangle, I will make data.x1 & data.y1 the top left of the rectangle and data.x2 & data.y2 the bottom right 🡪 do conversions with if statements
   5. After the rectangle dimensions and position is located, call the createNewEditImage function. This function takes the user’s initial mouse click and end mouse release to get the dimensions. This will also create a temporary new image that is used to convert. I found the starting position to loop, which is x0+x1 and y0+y1 (everything left of that will get cut off). I don’t need to get the ending dimensions because I have the height and width.
   6. After seeing the rectangle, the user clicks inside the rectangle and a new image is created.
10. Move the image around:
    1. If data.doCrop =0, that means the user hasn’t selected the crop function. Therefore, any mouse drag movements will move the image if data.inAction =1. Data.inAction will differentiate if the user is still holding down the mouse button. I have to make sure that the user is dragging the image to different locations.
    2. Data.x1 & data.y1 will be the initial positions of the click. Data.x2 & data.y2 will be the final positions. There difference will be where I am going to position the final image in terms of the initial position.

**Version Control:**

Still using Google Docs

**Modules List:**

Tkinter

Pillow

Below are from TP1 and TP2:

**Project Description:**

Name: Image Editor

Image Editor is a graphics image editor that has basic photo editing capabilities as well as more advanced features that are similar to Adobe’s Photoshop and Apple’s Photo Booth.

**Competitive Analysis:**

My project is inspired by my own interests in photo editing and the idea of being able to edit photos easily. The purpose that my project is to provide an easy-to-use photo editing tool that has not only the basic editing features, but also more advanced features such as merging and using your webcam to take photos.

In the past, there have been numerous of photo editing 112 term projects. I have identified three past projects that are similar to mine. Those are Image Editor by Michelle Wan, Image Editor by Teng Zhang, and Photo Twist by Chris Lewis. They all have the same basic features such as adjusting an image’s brightness or contrast. As of now, there are many features that my editor will perform such as adjusting the brightness, contrast, sharpness, blurriness, applying the filter, etc. I am also the idea of merging, in which 2 images can get merged together, which is similar to that of Michelle’s. However, I believe that Michelle’s project lacks on the usability, which I plan on doing in mine. Furthermore, to address my goal of creating an easy-to-use interphase, I will make it much more graphical and easier to navigate. I believe that Teng Zhang’s project has the best user interface because all the buttons are very clearly defined for the user. What differentiates my project from the two is that I allow users to import images from the web, from user’s computers, and from their webcam. If time permits after TP2, I will try to create a feature in the webcam where the user can select his/her face and it can add some effects similar to that of Photo Booth on Macs. In essence, the gap that my project will address, which will make it competitive in the photo editing area, is that it is easy-to-use and provides some advanced features.

**Structural Plan:**

Updated Design for TP2:

Class Sample App:

* This is the main class that is responsible for switching frames
* Has a show\_frame function that shows the particular frame

Class StartPage:

* This is the frame of the start page/main screen
* When the user enters the application, they will see this
* There will be two buttons: Start Editing and Help on this page

Class Help:

* This is the frame that displays what the application is and tells user to select an image
* There will be a back button that redirects back to home page

Class EditMode:

* This is the frame that is responsible for displaying the editing image as well as the different buttons for editing functions
* From the buttons, the user can select different functions
* Each function may or may not have additional buttons that appear for the user to select

filterOptions():

* This function is called when the filter button is clicked
* When called, four buttons appear: bw, yellow, blue, and pink
* When the user clicks on those buttons, the particular function is called a different filter will be applied

Mirror():

* This function is called when Mirror is clicked
* When called, the image is mirrored, meaning that the right side becomes the left and the left becomes the right

Rotate():

* This function is called when rotate is clicked
* Two buttons appear: Clockwise and Counterclockwise
* Depending on the selection, the image will rotate in a certain way
* Right now, I am using the built-in rotate function as my current algorithm has a lot of bugs and is difficult to debug without a proper user interface
* I am working on the UI first

Brightness():

* This function adjusts the brightness of the image
* When called, there is a ‘+’ or ‘-‘ button
* + means that brightness will be increased
* - means that brightness will be decreased
* I might change it so that it becomes a slider, instead of the two buttons

Contrast():

* Similar to Brightness, except changes contrast

Crop():

* Crops the image

Resize():

* User inputs the dimensions and the dimensions get resized

User can import and image by clicking File.

Right now, I focused on just getting an image from the computer, but there is an option to import from URL. The URL option works, except, I just hardcoded a URL instead of working on the UI for a pop up display.

Save and Save As in menu

Possible Additonal Features:

Merge

Seam Carving

**UI Design**:

* Similar to my storyboard from TP1. My goal is to create that design.

~~\_\_init\_\_.py:~~

* ~~This is the main file that runs the entire Image Editor application~~
* ~~In this directory, the files are: Start\_Screen, Edit\_Screen~~
* ~~Data.mode is initialized to ‘startScreen’ and will change depending on the user’s actions~~
* ~~Data.images = [] – This is a list of the images that the user wants to edit. I need to put it in a list, because there’s an editing option called merge, that combines multiple images together. However, only in this mode will the user be allowed to select multiple images.~~
* ~~Data.moves = [] – Every version/move since the last ‘Save’, will be stored here~~
* ~~Data.redo = [] – When the user hits undo, the undo move goes into this list~~

~~Start\_Screen.py:~~

* ~~This is the screen that users will see when they first enter the application~~
* ~~There’s a ‘Help’ button at the bottom left.~~
* ~~When the user hits ‘Help’, data.mode = ‘startScreenHelp’, and a pop-up will instruct the user~~
* ~~There’re three buttons ‘Computer’, ‘ Web’, and ‘Camera’ that allow users to choose how the want their images to be uploaded.~~
* ~~When the user clicks ‘Computer’, a pop up will ask the user to select a file from his/her computer.~~
* ~~When the user clicks ‘Web’, a pop up will ask the user to type in the URL of the image.~~
* ~~When the user clicks ‘Camera’, a pop up turn on his/her webcam and take a picture.~~ 
  + ~~Use Pygame for the camera~~
  + ~~If time permits, after TP2, I will add in a feature where the user selects his/her face, and the camera can add in special effects (similar to that of Photo Booth on the Mac)~~
* ~~After selecting the image, data.mode = ‘edit’~~

~~Edit.py: #This is just editing for 1 image~~

* ~~This is the screen for when an image has been selected, and the user is presented with different actions for editing.~~
* ~~I will most likely use a Class. Class takes in data.images~~
* ~~There are ‘Select Area’, ‘Crop’, ‘Brightness’, ‘Contrast’, ‘Reflect’, ‘Rotate’, ‘Sharpen’, ‘Blur’, ‘Filter’, and ‘Merge’~~
* ~~Initializes: self.brightness = 50, self.contrast = 50, self.rotate = 0, self.sharpen = 0, self.blur = 0~~
* ~~When an action is selected, data.mode changes, and the subclasses are called.~~
* ~~Each of these actions will have its own file~~
* ~~Ribbon contains ‘Save’, ‘Undo’, ‘Redo’, ‘Help’, and ‘Restart’ buttons~~
  + ~~Hitting Save will clear data.moves list~~
  + ~~When a user clicks ‘Help’, data.mode = editHelp, and a popup describing all the features is displayed~~
  + ~~Validate that data.moves != [] for redo moves.~~
  + ~~Restart Button reverts back to original image, so create data.original that stores the original~~
    - [~~https://www.tutorialspoint.com/python3/tk\_menu.htm~~](https://www.tutorialspoint.com/python3/tk_menu.htm)
  + ~~Back Button’ at the bottom goes back to Start\_Screen (data.mode = startScreen)~~
  + ~~If data.moves != []: pop up that warns user that they haven’t saved their work yet~~
  + ~~Ask them if they still want to continue~~
* ~~Functions: saveImage(), getPixel(), createImage()~~

~~Select\_Area.py:~~

* ~~Allows users to select and deselect area~~
* ~~Inherits from Edit~~

~~Crop.py:~~

* ~~Inherits from Select\_Area~~

~~Brightness.py:~~

* ~~Inherits from Edit~~
* ~~Screen on right that allows users to scroll and adjust brightness or input number~~
* [~~https://www.tutorialspoint.com/python/tk\_scale.htm~~](https://www.tutorialspoint.com/python/tk_scale.htm) ~~<-- Website that teaches how to use scroll in tkinter~~

~~Contrast.py:~~

* ~~Inherits from Edit~~
* ~~Screen on right that allows users to scroll and adjust contrast or input number~~

~~Mirror.py:~~

* ~~Inherits from Edit~~
* ~~Flips it so that the left of the image becomes the right and the right becomes the left~~

~~Rotate.py:~~

* ~~Inherits from Edit~~
* ~~Screen on right has two buttons: One rotates clockwise and one rotates counterclockwise~~
* ~~Rotates 90 degrees~~

~~Sharpen.py:~~

* ~~Inherits from Edit~~
* ~~Screen on right that allows users to scroll and adjust sharpness or input number~~

~~Blur.py:~~

* ~~Inherits from Edit~~
* ~~Screen on right that allows users to scroll and adjust blur or input number~~

~~Filter.py:~~

* ~~Inherits from Edit~~
* ~~Black and White, Red, Blue,~~

~~(If time permits, create Merge):~~

~~Merge.py:~~

* ~~Inherits from Edit~~
* ~~Adds to dictionary (imageDescription = dict()) adds to all of the features of the image to a dictionary, with the name of the image as the key~~

~~After TP2 and if time permits:~~

* ~~Add the special effects to Webcam~~
* ~~Add more editing features~~

**Algorithmic Plan:**

Algorithm is still the same

For Filtering:

1. Take image and make a copy
   1. This will be the image that will be edited
   2. Get width and height of the image
2. Create an array of pixels and change each pixel of the image (Each filter will have its own function)
   1. Double for loop the height and width to get each pixel and its R,G,B values, then convert set the R,G,B values into desired colors of the filter
   2. Image.load() gets the pixel map in PIL
3. After user leaves the Filtering tab, will set the image to the copied image (the one with the changed filter)

For changing the contrast:

1. Makes copy of image and gets its height and width
2. Double for loop the row and col to get each pixel and its R,G,B values, then apply the amount changed

For changing the brightness:

1. Makes copy of image and gets its height and width
2. Double for loop the row and col to get each pixel and its R,G,B values, then apply the amount changed

For creating the mirror image:

1. Makes copy of image and gets its height and width
2. Mirror = []
3. Double for loop the row, reverse the row and add to Mirror (since only swapping the left and right

For selecting area and cropping

1. Already written out the selecting area code for my Tech Demo
2. Use PIL
3. Takes the top left of rectangular selection and bottom right of rectangular selection
4. Crop the image with image.crop in PIL (If want to do it easy)
5. Harder way: get upper left and bottom right coordinates, then get boundaries of those coordinates and create a new pixel array of those coordinates

For rotating images:

1. Use the rotate in PyGame

For sharpening:

1. Easy method: Use the built-in one in PIL
2. Hard method: <https://ieeexplore.ieee.org/document/6030937/> this describes an algorithm for sharpness
3. Idea is to for loop each pixel, then use the algorithm to sharpen

For blurring:

1. Double for loop to get each pixel, then for each pixel, add the blurAmout

**Timeline Plan**:

April 12 – April 16:

* Design Proposal- Project Proposal, Storyboard, Preliminary Code
* Tech Demo (April 14)
* Finalize plan/idea
  + Ask questions
* Start coding the Main Menu (April 16)
* Fill out Time Sheets

April 17 – TP1 due!!:

* Meet with Mentor
* Start TP2:
  + Working Demo: April 18 – April 19:
* Work on TP2:
  + Working Demo:

April 22 – April 24:

* Work on TP2:
  + Working Demo:
* Fill out Time Sheet
* Update Design docs

April 25 – TP2 Due!

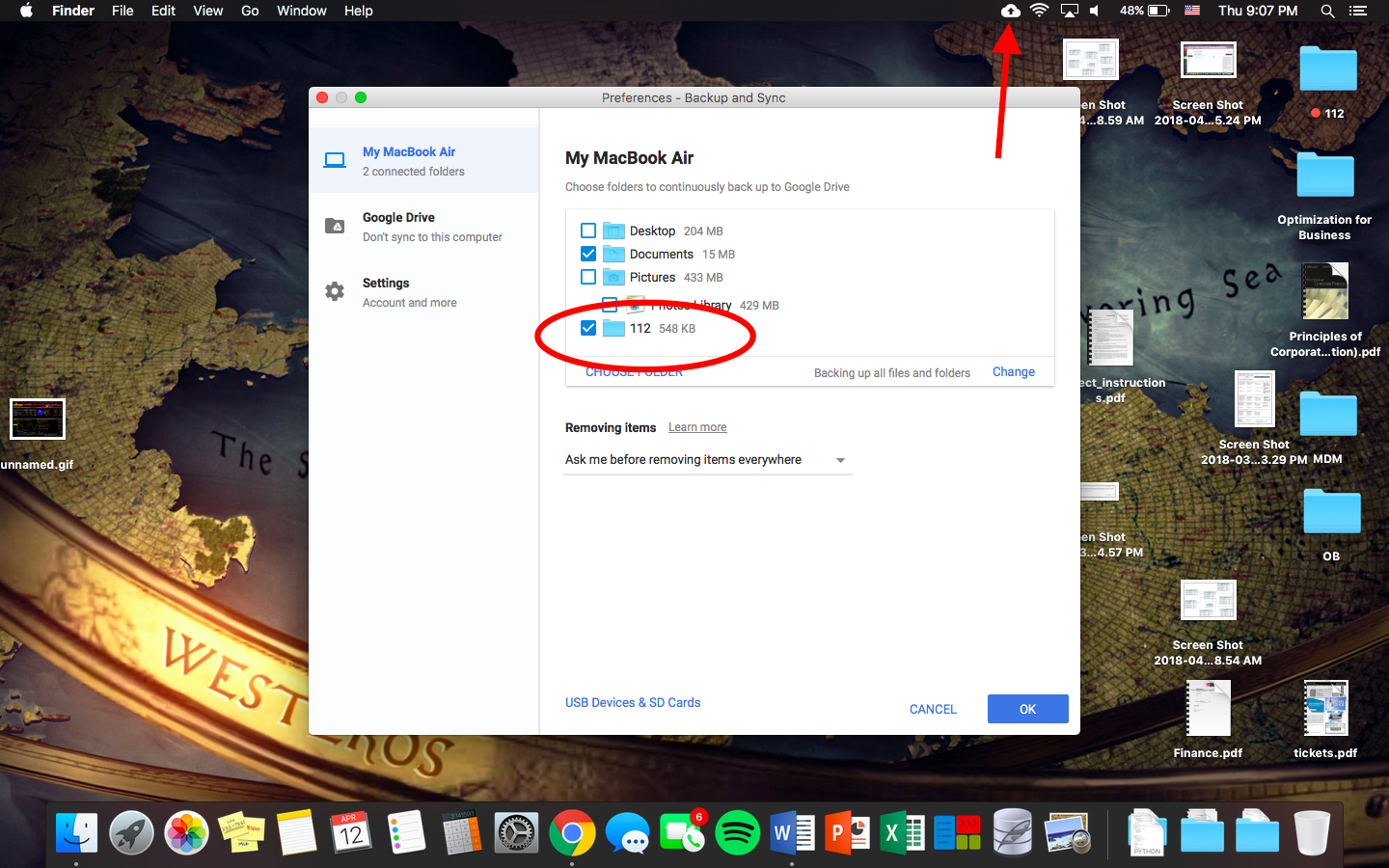
April 26 – May 3:

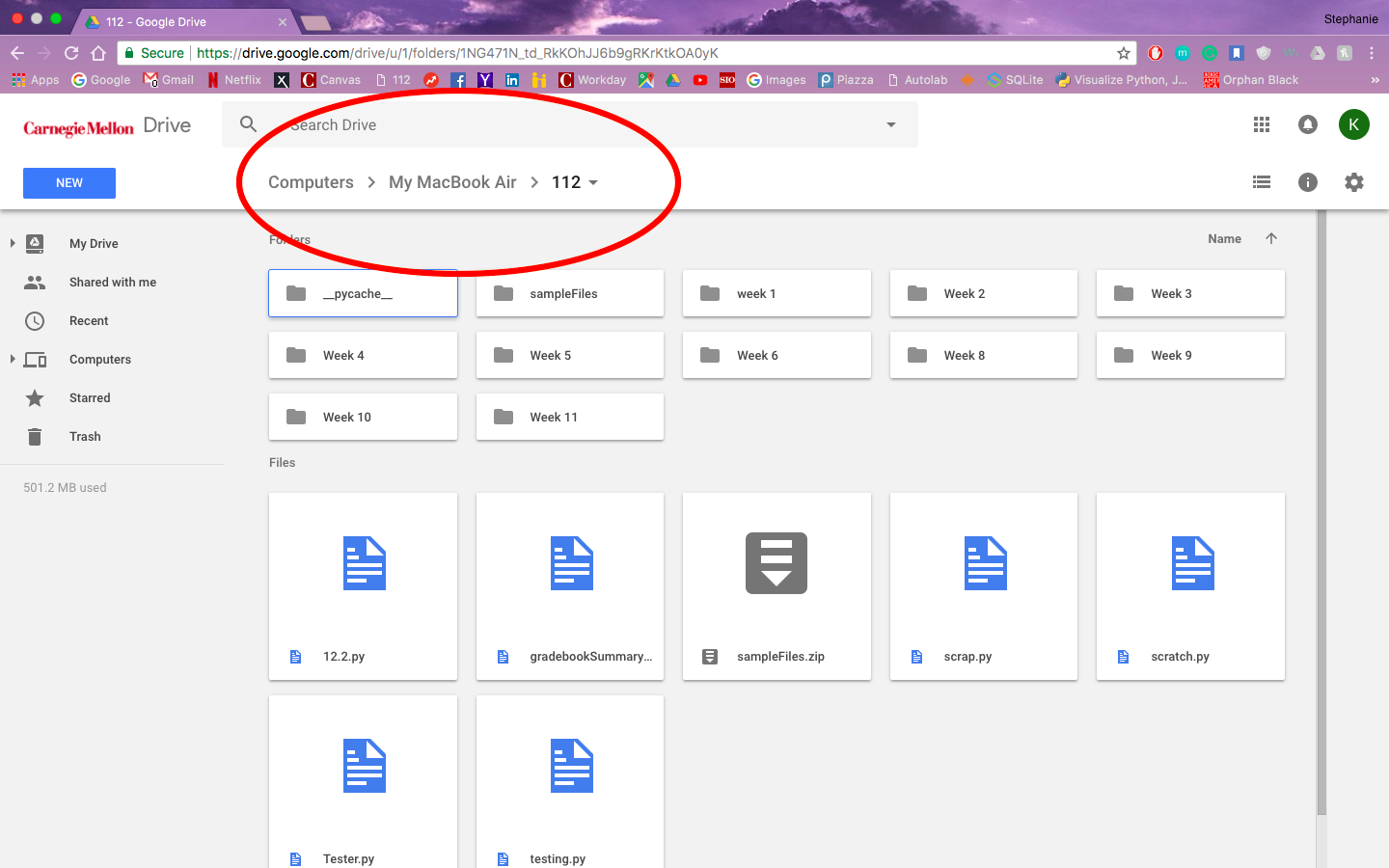
* Work on TP3

**Version Control Plan:**

Still use Google Drive

I will use Google Drive. I have Google Drive to my laptop, and it automatically syncs my 112 folder, which is the folder I store all my 112 work (including this project), in. The two images below show how Google Drive syncs my files from my computer to the Google Drive that is connected to my CMU email.





**Module List:**

None

Only using tkinter, Pillow

I am researching and learning about PAGE, which is better for designing UIs. Tkinter buttons are really difficult to format and make it user friendly.

After TP2, my aim is to learn Open CV so that I can add the function of using my webcam to upload an image.