Front-end/Backend report

1. The current state of the project:
    a. The following **front-end features** have been implemented:
        i. Title
        ii. Drag and drop file uploading area
        iii. Convert button
        iv. Project description area
    b. The following **backend features** have been completed:
        i. Drag and Drop zone - user has the ability to drag pictures into the area or alternatively click on the area to be able to upload a file
        ii. User also has the ability to remove uploaded pictures
        iii. Backend has made it so that only pictures and videos will be accepted
        iv. Two routes have been created, one for the home page and one for the results - as it is now the convert button automatically redirects to a dummy result page but this will be changed in further milestones
            1. We will need to check if a file has been uploaded to determine whether or not we move to the result page
2. Changes in milestone:
    a. Milestone1: Complete most of the front-end features
    b. Milestone2: The website should have most of the front-end features (such as the feature to upload the image) implemented as well be working on the back-end.
    c. Milestone3: The website should have a completed front-end and be ready to incorporate the model in the back-end.
    d. Milestone4: AWS server will be opened, and the project should have the model incorporated into the website and be hosted on the AWS servers.
3. Current challenge
    a. When an image file is inserted into the drag and drop area, the website design breaks; however, this will be fixed in the following milestone.
        i. To do this, we may need to add event listeners to when we add a file and then change the inner HTML of the page to hide previous elements
4. A description of each component and how it was implemented
    a. The drag and drop HTML and CSS code was referenced from this website: https://www.codingnepalweb.com/2021/02/drag-drop-or-browse-file-upload-feature.html. Important areas are divided such as title, drag and drop, convert, and description area. Then, features required in each area are inserted later. The style of each component and alignment was done with CSS under <style>.
    b. Routing:
        i. Flask was used to implement different routes/webpages and to configure the dropzone. With this, we now have the ability to have two pages - one for the uploading of the file and one for the model results
    c. Dropzone
        i. The backend of the drag and drop area was done using https://www.dropzonejs.com/ and has been configured for our specific uses. This

involved configuring the dropzone so it would only accept photos and video and that it would only accept one file at a time.

5. The overall contributions of each member to the project
    a. Front-end was done by Han
    b. Back-end was done by Adriana

Machine Learning Report:
Data scraping portion:

1. Current state: Have all needed photographic images and videos have half of the needed watercolor images.
2. Changes in milestone: N/A
3. Current challenge: Having trouble getting 5000 unique watercolor images because Flickr API only allows taking 3600 per hour, but I can't save my spot where I had taken images from. Also, the total amount of data I have is around 30gb, Drive and git aren't letting me upload things. My plan is to separate all of the data (around 25gb is the amount the images take) into 5 5gb image datasets, and we can train the model on each one.
4. How it was done: I followed this (https://towardsdatascience.com/how-to-use-flickr-api-to-collect-data-for-deep-learning-experiments-209b55a09628) guide. Basically, I got an API key from Flickr, then got a script to scrape image URLs from the API, and a script to download images from those URLs, then I did that for every category of images we needed. For the videos, I went to https://viratdata.org/, and downloaded 200 of the smaller videos from their ground videos dataset.
5. The overall contributions of each member to the project: Adrienne

ML code portion:

1. Current state: The code for the generator and critic models is completed as well as the training code for these models. However, the training code might have some bugs that will show up when the model is being trained. Also, the critic currently does not accept images of sizes other than 64x64.
2. Changes in milestone:
    a. Milestone1: Most of the training and ~~testing code~~ completed.
        i. Reasoning: There is no testing code to write since the results of the model cannot be tested in code. The model results will be visually and manually inspected to see if they are acceptable.
3. Current challenges: Based on some minimal testing, it will take a very long time to train the models. Currently, it can take up to an hour per a few hundred images. To help with this, the generator models can be simplified from 9 resnet layers to 4 or 5 resnet layers. This should improve the training time with the cost of potentially making the results worse. In the end, the resnet layers can be added back to the generators to improve the final results.
4. A description of each component and how it was implemented:
    a. The code for the dataset was modified from https://github.com/eriklindernoren/PyTorch-GAN to represent images of type A (images of the style to transfer) and type B (images to transfer style onto).

b. The training code was also modified from https://github.com/eriklindernoren/PyTorch-GAN and the class notes. The training code trains both of the critics and the generators to implement a cycleGAN.
c. The code for the critic and generator model were modified from the class notes on Gans to fit the model architecture outlined in the proposal.

5. The overall contributions of each member to the project: Harjot