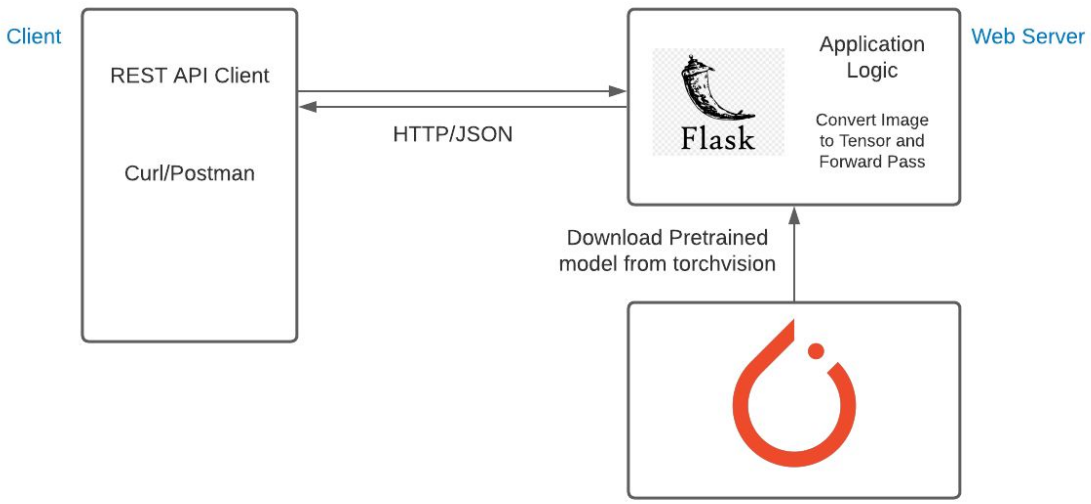
Nikita Agarwal, Varun Prasad, Ajinkya Ghadge
Design document

1. Project structure and instructions
    a. Instructions to test deployed container on Heroku
        i. Place your images in the test folder and execute the python file testscript.py. It will generate results for all the files in the folder. Note that it may take upto a minute for the heroku container to wake up from sleep, after which it will work faster.
    b. instructions to load and test (There is no difference between the image and the heroku deployment other than deployment)
        i. Download the ~3GB image from link https://drive.google.com/file/d/18I0kOLSW1rV3gUS8CtnYZmYweXTfAb6s/view?usp=sharing
        ii. Load the container using the command “docker load < flask-docker-demo-app.tar.gz”
        iii. Use command “sudo docker run --name flask-docker-demo-app -p 5000:5000 flask-docker-demo-app”
        iv. You should be able to run the test script testscript2.py by
    c. instructions to build docker container and image and deploy it to Heroku
        i. Inside the repository run the command “sudo docker build --tag flask-docker-demo-app .”
        ii. Run the command “sudo docker run --name flask-docker-demo-app -p 5000:5000 flask-docker-demo-app”
        iii. You should be able to use curl to post on the API
        iv. To save the image using docker save --output flask-docker-demo-app.tar flask-docker-demo-app
        v. To deploy the app on heroku
2. Architecture
    a. Flask is a web application framework which is being used to serve the image classification model as a service. We use REST architecture as shown below
    b. For containerization of the application we use Docker, which packages all dependencies together and can be exported into an image, which can be used as a plug and play system.
    c. For deploying the container we use Heroku, which is a cloud compute and storage solution.
    d. Execution flow of the program is as follows:
        i. An image is passed by the user, in the payload of a POST request.
        ii. The image is flattened out transformed before performing a forward pass.
        iii. We perform the forward pass and obtain the output from DenseNet
        iv. The output category is obtained by getting value for the output from a dictionary.
        v. The category is returned as an HTTP response



e.

3. Design considerations
    a. Using REST
        i. Other methods like SOAP protocol or RPC are slightly more complex and the given application can benefit from simple request and response based architecture used by REST architecture as it is a really simple system.
        ii. The Flask framework is primarily built for applications using the REST architecture.
    b. Excluding pretrained-model from the program to limit size of docker container and give flexibility
        i. The size of the container gets really large due to installation of all dependencies. We thought of letting the server download the pretrained model so as to keep the size of the docker image small.
    c. Deploying on Heroku vs Google cloud vs AWS
        i. All platforms are equally competitive with AWS offering great value. However, there are two areas where Heroku exceeds its competitors. Ease of working with the platform, and a very generous free tier. One can very easily push changes to a Heroku container.