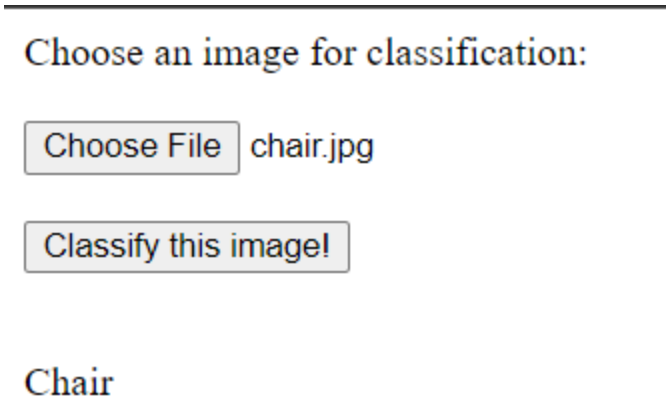
HOW TO RUN THE CODE:

- First go to the **"prateek_classification_main"** folder
- To run it on **command prompt**, write the following line
  "python -m classification_api".
- This will run the **Flask API** and it will run on a Localhost server, just click the link and the website will open.
- To run it on **Docker**:
  "docker build -t flaskapp --no-cache . -f DockerFile".
  This will build a new docker image and read the docker file.
  NOTE: The port used for the website is **5000**.
- You can then choose a file to see what type of furniture it is and then click Classify, the classify model will run in the background and display the predicted output on the website.



REQUIREMENTS:

You have to install the following:

- pip install tensorflow
- pip install opencv-python
- pip install matplotlib
- pip install numpy

And import the following libraries:

- import pip
- import numpy as np
- import tensorflow as tf
- import os
- import cv2
- from tensorflow import keras
- from tensorflow.keras import layers
- import matplotlib.pyplot as plt
- from flask_cors import CORS
- from werkzeug.utils import secure_filename
- import os
- import cv2
- import numpy

- from PIL import Image

Classification Model:

- First, all the images need to be preprocessed so that there is uniformity in the dataset and is easier for the model to train. That is done by rescaling the images to 128*128, converting them to grayscale.
- After that, all the images provided for Test in each category(Bed, Chair and Sofa) are read and are passed to the preprocessing function. A label is given to each of the categories because that is consistent and better for the CNN model.
- When working with image data, **Sequential convolutional neural networks (CNNs)** are often the most effective type of neural network to use. I used a combination of convolutional layers and pooling layers, followed by a few fully connected (dense) layers.
- To address overfitting in neural networks, another approach is to incorporate dropout regularization
- In order to train the weights and handle multiple classes (3 classes), I employed the widely used Adam optimizer and utilized the categorical cross-entropy loss function.