```
import streamlit as st
import tensorflow
import pandas as pd
from PIL import Image
import pickle
import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import ResNet50,
preprocess_input
from tensorflow.keras.layers import GlobalMaxPooling2D
from tensorflow.keras.models import Sequential
from numpy.linalg import norm
from sklearn.neighbors import NearestNeighbors
import os

features_list = pickle.load(open("image_features_embedding.pkl", "rb"))
img_files_list = pickle.load(open("img_files.pkl", "rb"))

model = ResNet50(weights="imagenet", include_top=False, input_shape=(224,
224, 3))
model.trainable = False
model = Sequential([model, GlobalMaxPooling2D()])

st.title('Clothing recommender system')


def save_file(uploaded_file):
    try:
        with open(os.path.join("uploader", uploaded_file.name), 'wb') as
f:
            f.write(uploaded_file.getbuffer())
            return 1
    except:
        return 0


def extract_img_features(img_path, model):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    expand_img = np.expand_dims(img_array, axis=0)
    preprocessed_img = preprocess_input(expand_img)
    result_to_resnet = model.predict(preprocessed_img)
    flatten_result = result_to_resnet.flatten()
    # normalizing
    result_normlized = flatten_result / norm(flatten_result)

    return result_normlized


def recommendd(features, features_list):
    neighbors = NearestNeighbors(n_neighbors=6, algorithm='brute',
metric='euclidean')
    neighbors.fit(features_list)

    distence, indices = neighbors.kneighbors([features])

    return indices
```

```
uploaded_file = st.file_uploader("Choose your image")
if uploaded_file is not None:
    if save_file(uploaded_file):
        # display image
        show_images = Image.open(uploaded_file)
        size = (400, 400)
        resized_im = show_images.resize(size)
        st.image(resized_im)
        # extract features of uploaded image
        features = extract_img_features(os.path.join("uploader",
uploaded_file.name), model)
        #st.text(features)
        img_indicess = recommendd(features, features_list)
        col1,col2,col3,col4,col5 = st.columns(5)

        with col1:
            st.header("I")
            st.image(img_files_list[img_indicess[0][0]])

        with col2:
            st.header("II")
            st.image(img_files_list[img_indicess[0][1]])

        with col3:
            st.header("III")
            st.image(img_files_list[img_indicess[0][2]])

        with col4:
            st.header("IV")
            st.image(img_files_list[img_indicess[0][3]])

        with col5:
            st.header("V")
            st.image(img_files_list[img_indicess[0][4]])
    else:
        st.header("Some error occur")
```