ALGORITHMS FOR WEB AND INFORMATION RETRIEVAL

PROJECT TITLE :

FASHION RECOMMENDER SYSTEM

TEAM DETAILS :

ID	NAME	SRN	SECTION
1	PRIYA MOHATA	PES2UG19CS301	E
2	PRIYANSH GAURAV	PES2UG19CS302	E
3	R SHARMILA	PES2UG19CS309	E
4	RITIK	PES2UG19CS332	E

CODE :

MAIN.PY

```python
import streamlit as st # to host system on web.
import os
from PIL import Image
import numpy as np
import pickle
import tensorflow
from tensorflow.keras.preprocessing import image
from tensorflow.keras.layers import GlobalMaxPooling2D
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
from sklearn.neighbors import NearestNeighbors
from numpy.linalg import norm

feature_list = np.array(pickle.load(open('embeddings.pkl', 'rb')))
filenames = pickle.load(open('filenames.pkl', 'rb'))

model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
model.trainable = False

model = tensorflow.keras.Sequential([
    model,
    GlobalMaxPooling2D()
])

st.title('Fashion Recommender System')
```

```python
# to upload the file - this code is required
def save_uploaded_file(uploaded_file):
    try:
        with open(os.path.join('uploads', uploaded_file.name), 'wb') as f:
            f.write(uploaded_file.getbuffer())
        return 1
    except:
        return 0

# calling the feature extraction function for the uploaded image.
def feature_extraction(img_path, model):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    expanded_img_array = np.expand_dims(img_array, axis=0)
    preprocessed_img = preprocess_input(expanded_img_array)
    result = model.predict(preprocessed_img).flatten()
    normalized_result = result / norm(result)
    return normalized_result

# recommendation
def recommend(features, feature_list):
    neighbors = NearestNeighbors(n_neighbors=6, algorithm='brute', metric='euclidean')
    neighbors.fit(feature_list)
    distances, indices = neighbors.kneighbors([features])
    return indices
```

```
# steps
# file upload -> save
uploaded_file = st.file_uploader("Choose an image")
print(uploaded_file.name)
if uploaded_file is not None:
    if save_uploaded_file(uploaded_file):
        # display the file
        display_image = Image.open(uploaded_file)
        st.image(display_image)
        # feature extract
        features = feature_extraction(os.path.join("uploads", uploaded_file.name), model)
        # st.text(features)
        # recommendention
        indices = recommend(features, feature_list)
        # show
        col1, col2, col3, col4, col5 = st.beta_columns(5)

        with col1:
            st.image(filenames[indices[0][0]])
        with col2:
            st.image(filenames[indices[0][1]])
        with col3:
            st.image(filenames[indices[0][2]])
        with col4:
            st.image(filenames[indices[0][3]])
        with col5:
            st.image(filenames[indices[0][4]])
    else:
        st.header("Some error occurred in file upload")

```

APP.PY

```python
import tensorflow
# image module to access functionalities related to images.
from tensorflow.keras.preprocessing import image
from tensorflow.keras.layers import GlobalMaxPooling2D
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
import numpy as np
from numpy.linalg import norm
import os
# tqdm - for knowing the progress of the for loop
from tqdm import tqdm
import pickle

model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
model.trainable = False

model = tensorflow.keras.Sequential([
    model,
    GlobalMaxPooling2D()
])

# print(model.summary())
```

```python
def extract_features(img_path, model):
    # loading the image
    img = image.load_img(img_path, target_size=(224, 224))

    # converting the image to an array - 224 * 224 * 3 - since RGB image
    img_array = image.img_to_array(img)

    # to reshape the images - input= image array , output=batch of images.
    expanded_img_array = np.expand_dims(img_array, axis=0)

    # preprocesses the input - by sending it to the function - preprocess_input - converts the input to the
    # appropriate input required by resnet
    # output given by it - Image is converted from RGB to BGR and other preprocessing is done for the resnet model.
    preprocessed_img = preprocess_input(expanded_img_array)

    # pass the image to resnet and convert it to one dimensional
    result = model.predict(preprocessed_img).flatten()

    # normalize the value from 0 to 1
    normalized_result = result / norm(result)

    return normalized_result
```

```python
filenames = []

# for the images - we are appending the path name for the filenames.
for file in os.listdir('images'):
    filenames.append(os.path.join('images', file))

feature_list = []

for file in tqdm(filenames):
    feature_list.append(extract_features(file, model))

pickle.dump(feature_list, open('embeddings.pkl', 'wb'))
pickle.dump(filenames, open('filenames.pkl', 'wb'))
```

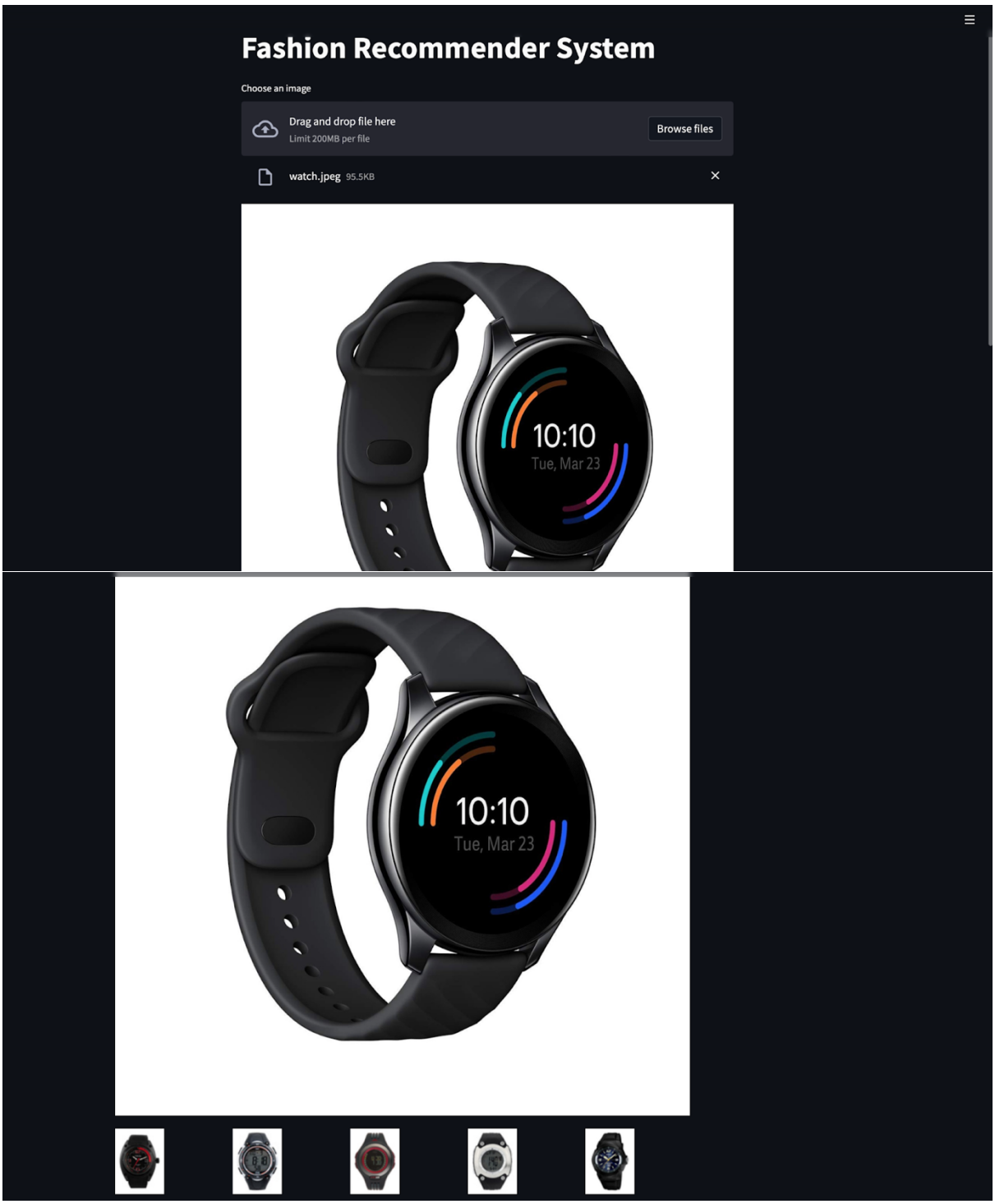
TEST.PY

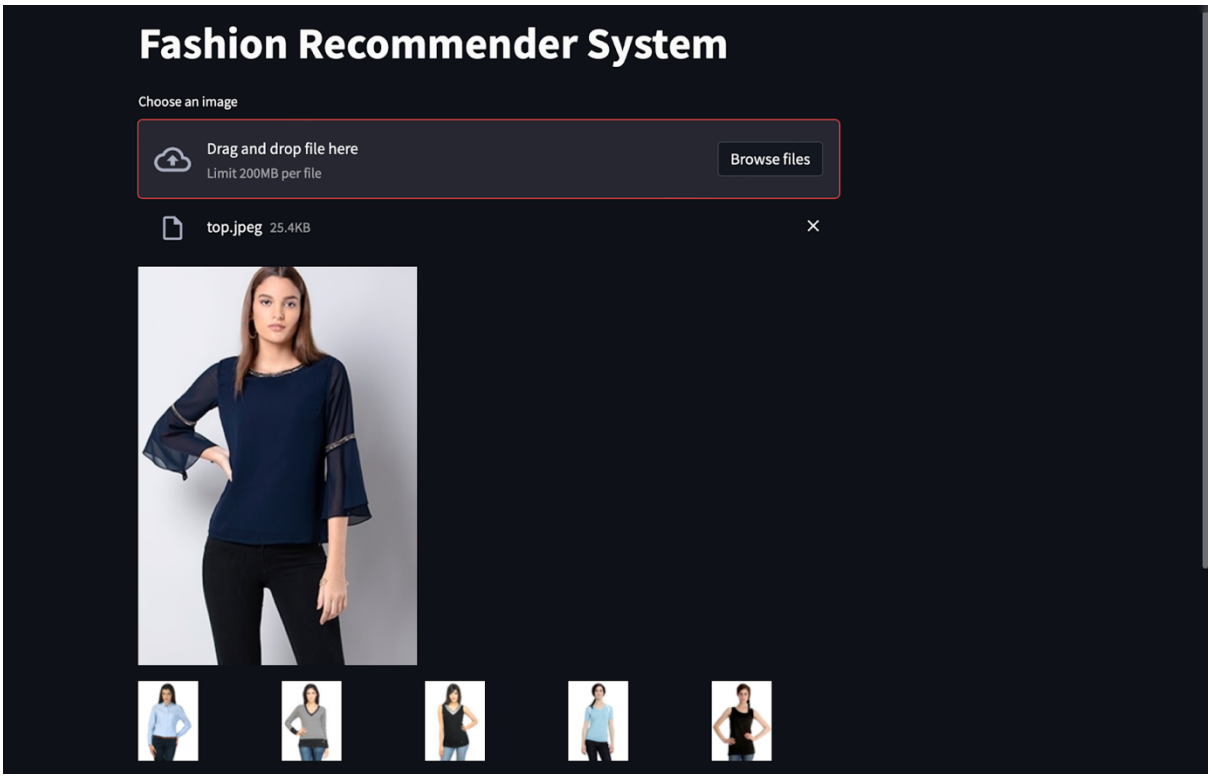
```python
import pickle
import tensorflow
import numpy as np
from numpy.linalg import norm
from tensorflow.keras.preprocessing import image
from tensorflow.keras.layers import GlobalMaxPooling2D
from tensorflow.keras.applications.resnet50 import ResNet50,preprocess_input
from sklearn.neighbors import NearestNeighbors
import cv2

# loading the feature list and filenames
feature_list = np.array(pickle.load(open('embeddings.pkl','rb')))
filenames = pickle.load(open('filenames.pkl','rb'))


model = ResNet50(weights='imagenet',include_top=False,input_shape=(224,224,3))
model.trainable = False

model = tensorflow.keras.Sequential([
    model,
    GlobalMaxPooling2D()
])

# taking a new image and extracting features and passing it to the model.
img = image.load_img('sample/jersey.jpeg',target_size=(224,224))
img_array = image.img_to_array(img)
expanded_img_array = np.expand_dims(img_array, axis=0)
preprocessed_img = preprocess_input(expanded_img_array)
result = model.predict(preprocessed_img).flatten()
normalized_result = result / norm(result)

# Finding the top-k nearest images using KNN
# Metric used = Euclidean Distance
# We can use cosine similarity as well.

neighbors = NearestNeighbors(n_neighbors=6,algorithm='brute',metric='euclidean')
#neighbors = NearestNeighbors(n_neighbors=6,algorithm='brute',metric='cosine')
# give the input data to .fit function
neighbors.fit(feature_list)

# finding the nearest k neighbours for the normalized result.
distances,indices = neighbors.kneighbors([normalized_result])

# printing the indices of the vectors which are nearest to the given image.
print(indices)

for file in indices[0][1:6]:
    temp_img = cv2.imread(filenames[file])
    # to display the image.
    cv2.imshow('output',cv2.resize(temp_img,(512,512)))
    # to make the screen wait.
    cv2.waitKey(0)
```

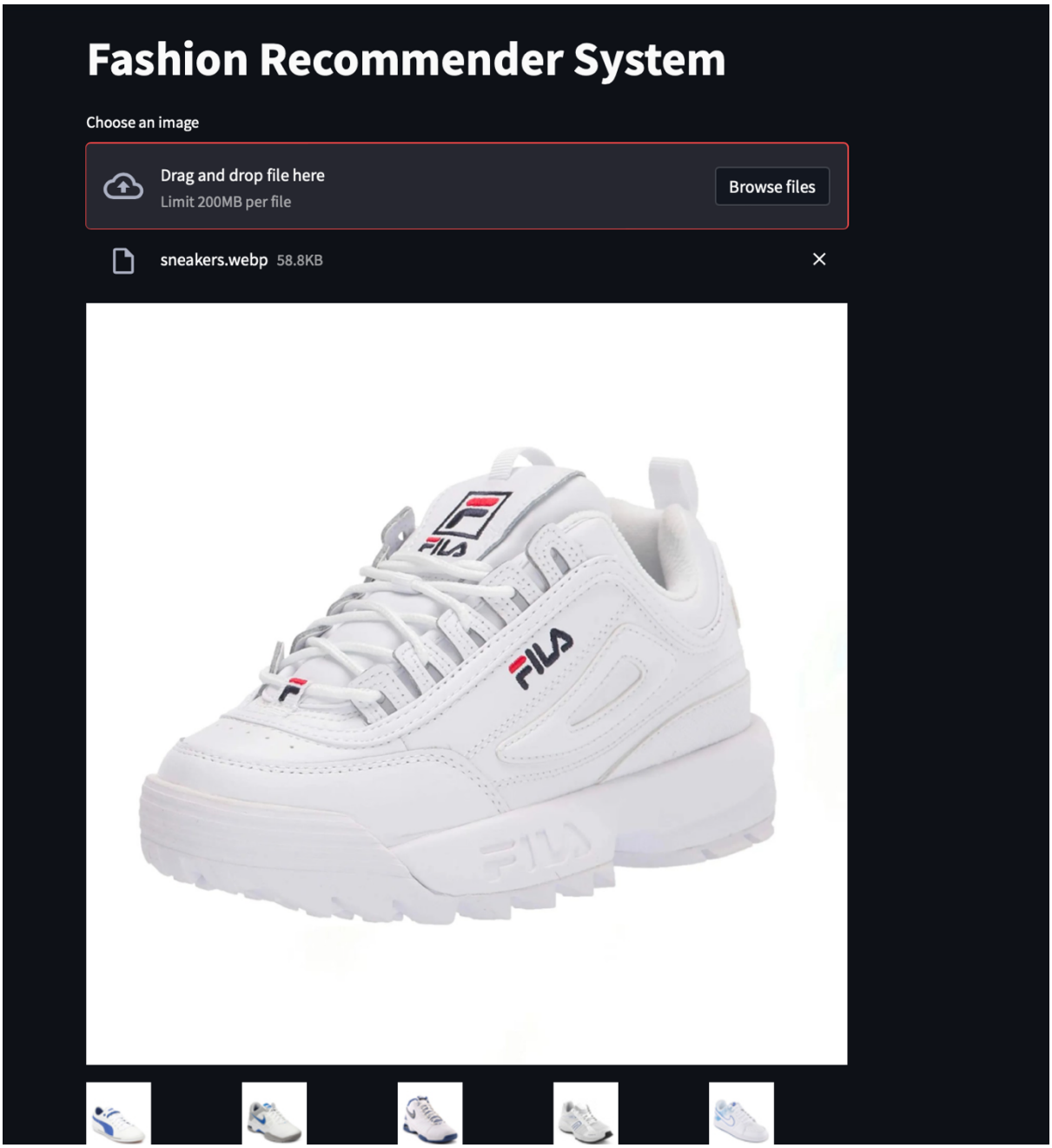
To run the code :

```
python3 app.py [ generates embeddings.pkl , filenames.pkl ]
streamlit run “path to main.py”
```

OUTPUT :


Fashion Recommender System
Choose an image
Drag and drop file here
Limit 200MB per file
Browse files
watch.jpeg 95.5KB
10:10
Tue, Mar 23
10:10
Tue, Mar 23


Fashion Recommender System
Choose an image
Drag and drop file here
Limit 200MB per file
Browse files
top.jpeg 25.4KB


Fashion Recommender System
Choose an image
Drag and drop file here
Limit 200MB per file
Browse files
sneakers.webp 58.8KB