Peer Evaluation System UI/UX

We worked on the signature recognition model and developed a system to classify and rename the files based on the Signature.

The load_stored_signatures function preloads and preprocesses all stored signatures in one step, eliminating redundant file reads and preprocessing during comparisons. By using multiprocessing.Pool for parallel processing, the comparison between uploaded and stored signatures is distributed across multiple CPU cores, significantly reducing runtime. Both uploaded and stored signatures are loaded and processed only once, optimizing efficiency. A similarity score threshold (e.g., > 0.85) is used to determine whether the signatures are similar enough to be considered a match.

The code for the model and the output screenshots are attached below: -

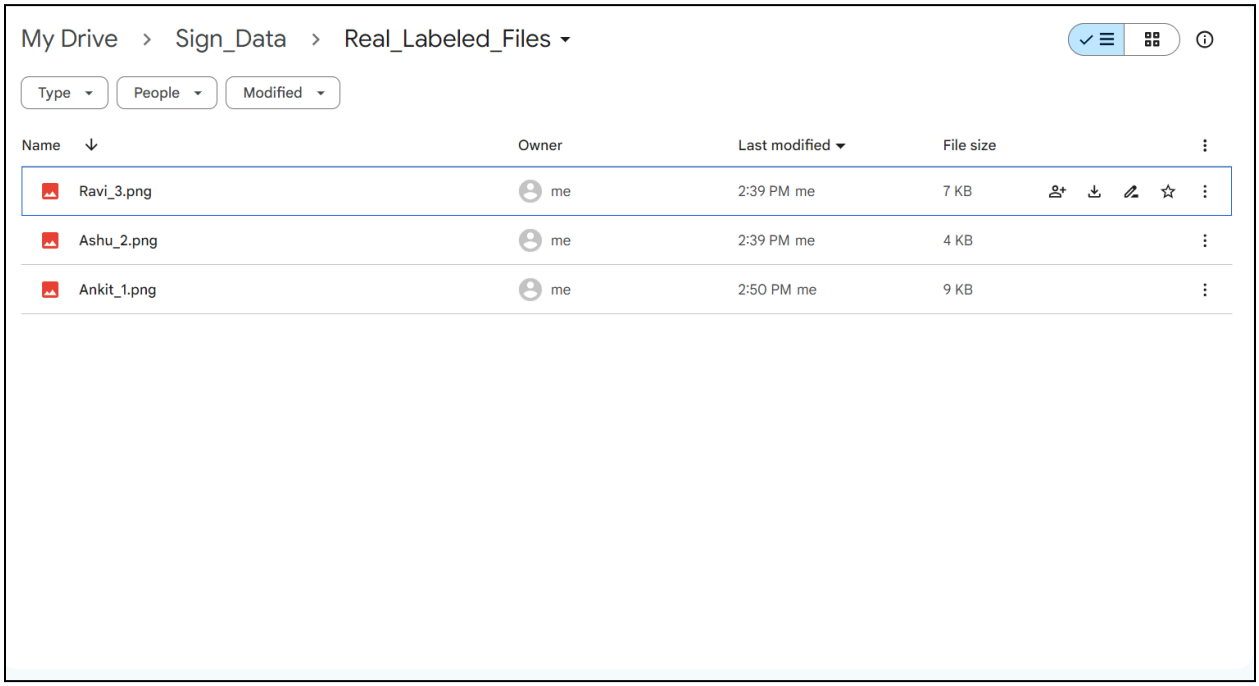
- **Python code: -**

```
import os
import cv2
import numpy as np
from skimage.metrics import structural_similarity as ssim
from scipy.ndimage import center_of_mass
from multiprocessing import Pool, cpu_count

IMG_WIDTH = 256
IMG_HEIGHT = 256

# Function to preprocess image (resize and normalize)
def preprocess_image(image):
    image = cv2.resize(image, (IMG_WIDTH, IMG_HEIGHT))  # Resize
    image = image / 255.0  # Normalize pixel values
    return image

# Align signatures by centering the image based on the center of mass
def align_images(image):
    cy, cx = center_of_mass(image)
    height, width = image.shape
    shift_x = int(width / 2 - cx)
    shift_y = int(height / 2 - cy)
    translation_matrix = np.float32([[1, 0, shift_x], [0, 1, shift_y]])
```

```
    aligned_image = cv2.warpAffine(image, translation_matrix, (width, height))
    return aligned_image

# Compare signatures using SSIM with the correct data range
def compare_signatures(stored_sig, uploaded_sig):
    aligned_stored_sig = align_images(stored_sig)
    aligned_uploaded_sig = align_images(uploaded_sig)
    score, _ = ssim(aligned_stored_sig, aligned_uploaded_sig, full=True, data_range=1.0)
    return score

# Load and preprocess stored signatures only once
def load_stored_signatures(stored_signatures_folder):
    stored_signatures = []
    stored_filenames = []
    for stored_filename in os.listdir(stored_signatures_folder):
        stored_path = os.path.join(stored_signatures_folder, stored_filename)
        stored_signature = cv2.imread(stored_path, cv2.IMREAD_GRAYSCALE)
        stored_signature = preprocess_image(stored_signature)
        stored_signatures.append(stored_signature)
        stored_filenames.append(stored_filename)
    return stored_signatures, stored_filenames

# Function to compare one uploaded signature with all stored signatures
def compare_with_stored(uploaded_signature, stored_signatures, stored_filenames,
uploaded_filename, threshold=0.85):
    matched_filename = None
    for stored_signature, stored_filename in zip(stored_signatures, stored_filenames):
        similarity_score = compare_signatures(stored_signature, uploaded_signature)
        if similarity_score > threshold:
            matched_filename = stored_filename
            break
    return uploaded_filename, matched_filename

# Main function to process all uploaded signatures and compare them
def process_signatures(uploaded_signatures_folder, stored_signatures, stored_filenames,
matched_signatures_folder):
    matched_files = []
    uploaded_files = os.listdir(uploaded_signatures_folder)

    # Load all uploaded signatures in memory
```

```
    uploaded_signatures = [(cv2.imread(os.path.join(uploaded_signatures_folder, fname),
cv2.IMREAD_GRAYSCALE), fname)
                    for fname in uploaded_files]

   # Preprocess all uploaded signatures
        uploaded_signatures = [(preprocess_image(img), fname) for img, fname in
uploaded_signatures]

   # Use multiprocessing to parallelize the comparison
   pool = Pool(cpu_count())
    results = pool.starmap(compare_with_stored, [(uploaded_signature, stored_signatures,
stored_filenames, fname)
                                  for uploaded_signature, fname in uploaded_signatures])
   pool.close()
   pool.join()

   # Rename matched files
   for uploaded_filename, matched_filename in results:
      if matched_filename:
         old_path = os.path.join(uploaded_signatures_folder, uploaded_filename)
         new_path = os.path.join(matched_signatures_folder, matched_filename)
         os.rename(old_path, new_path)
         matched_files.append((uploaded_filename, matched_filename))
         print(f"Match found! Renamed {uploaded_filename} to {matched_filename}")

   return matched_files

# Folder paths
stored_signatures_folder = '/content/drive/MyDrive/Sign_Data/Real_Sign_Dump/'
uploaded_signatures_folder = '/content/drive/MyDrive/Sign_Data/Real_Test/'
matched_signatures_folder = '/content/drive/MyDrive/Sign_Data/Real_Labeled_Files/'

# Load stored signatures once
stored_signatures, stored_filenames = load_stored_signatures(stored_signatures_folder)

# Process and match uploaded signatures
matched_files = process_signatures(uploaded_signatures_folder, stored_signatures,
stored_filenames, matched_signatures_folder)

print(f"Matching process completed. Total matched files: {len(matched_files)}")
```

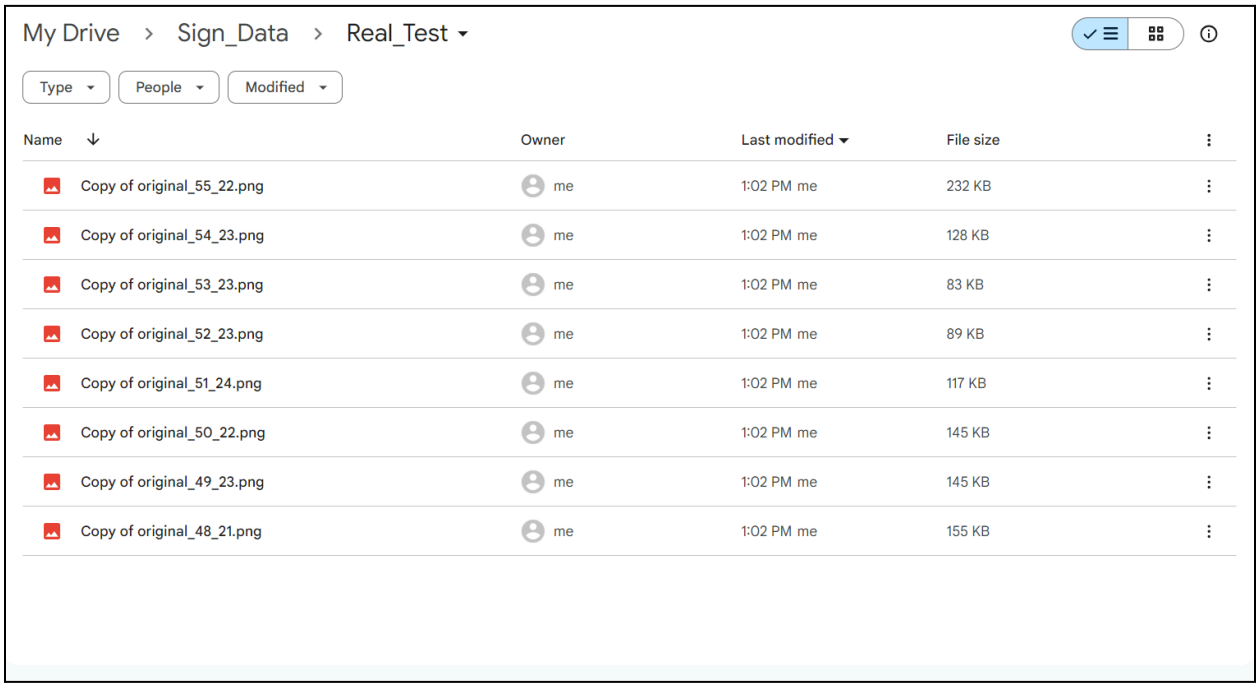
- **The outputs: -**



My Drive > Sign_Data > Real_Labeled_Files

Type | People | Modified

Name	Owner	Last modified	File size
Ravi_3.png	me	2:39 PM me	7 KB
Ashu_2.png	me	2:39 PM me	4 KB
Ankit_1.png	me	2:50 PM me	9 KB



My Drive > Sign_Data > Real_Test

Type | People | Modified

Name	Owner	Last modified	File size
Copy of original_55_22.png	me	1:02 PM me	232 KB
Copy of original_54_23.png	me	1:02 PM me	128 KB
Copy of original_53_23.png	me	1:02 PM me	83 KB
Copy of original_52_23.png	me	1:02 PM me	89 KB
Copy of original_51_24.png	me	1:02 PM me	117 KB
Copy of original_50_22.png	me	1:02 PM me	145 KB
Copy of original_49_23.png	me	1:02 PM me	145 KB
Copy of original_48_21.png	me	1:02 PM me	155 KB

My Drive > Sign_Data > Real_Sign_Dump

Type | People | Modified

Name	Owner	Last modified	File size
Ravi_4.png	me	Sep 19, 2024 me	5 KB
Ravi_3.png	me	Sep 19, 2024 me	5 KB
Ravi_2.png	me	Sep 19, 2024 me	6 KB
Ravi_1.png	me	Sep 19, 2024 me	7 KB
Ashu_4.png	me	Sep 19, 2024 me	4 KB
Ashu_3.png	me	Sep 19, 2024 me	4 KB
Ashu_2.png	me	Sep 19, 2024 me	4 KB
Ankit_4.png	me	2:51 PM me	9 KB
Ankit_3.png	me	Sep 19, 2024 me	8 KB
Ankit_2.png	me	Sep 19, 2024 me	9 KB