CELEBRITY LOOK ALIKE

Python Jackfruit Level Problem Solving

BTECH FIRST SEM

SUBMITTED BY,

SHRUTHI P-PES2UG25EC120

SMRITHI J HERLE-PES2UG25AM256

SHRAVYA M-PES2UG25AM247

SANJANA V-PES2UG25EC108

PROBLEM STATEMENT DESCRIPTION

In today’s digital era, face recognition has become an essential technology in applications such as security, social media, and entertainment. One interesting use case of face recognition is identifying a **celebrity look-alike** for a given human face.

The problem addressed in this project is to **design and develop a desktop application** that allows a user to upload their photograph and automatically find the celebrity whose facial features most closely resemble the user’s face. The system should be easy to use, visually intuitive, and capable of performing face comparisons accurately.

Objectives

- To develop a GUI-based application for image selection.
- To use deep learning–based face recognition to compare facial features.
- To identify and display the closest matching celebrity image.
- To present the similarity score in an understandable format.

METHODOLOGY

The system follows a face verification and comparison approach using a pre-trained deep learning model.

1. The user selects an image from their system.
2. The application loads a dataset of celebrity images.
3. Each celebrity image is compared with the user image using a face recognition model.

4. A distance score is calculated for each comparison.
5. The celebrity image with the **minimum distance** is selected as the best match.
6. The matched celebrity image and similarity score are displayed on the GUI.

ALGORITHM

1. Start the application.
2. User selects an image through the file dialog.
3. Store the image path.
4. Load all celebrity images from the dataset folder.
5. For each celebrity image:
6. Compare it with the user image using DeepFace.
7. Extract the distance score.
8. Identify the image with the minimum distance.
9. Display:
10. Matched celebrity image.
11. Celebrity name.
12. Similarity score.
13. End.

FLOWCHART

Start

|

v

Select User Image

|

v

Load Celebrity Dataset

|

v

Compare User Image with Celebrity Images

|

v

Calculate Distance Scores

|

v

Find Best Match

SOFTWARE OR LIBRARIES USED

Programming Language

- **Python 3**
-

Libraries and Tools

- **DeepFace** – Deep learning–based face recognition and verification.
- **Tkinter** – GUI development.
- **PIL (Pillow)** – Image processing and resizing.
- **OS** – File handling.
- **Threading** – Prevents GUI freezing during processing.
-

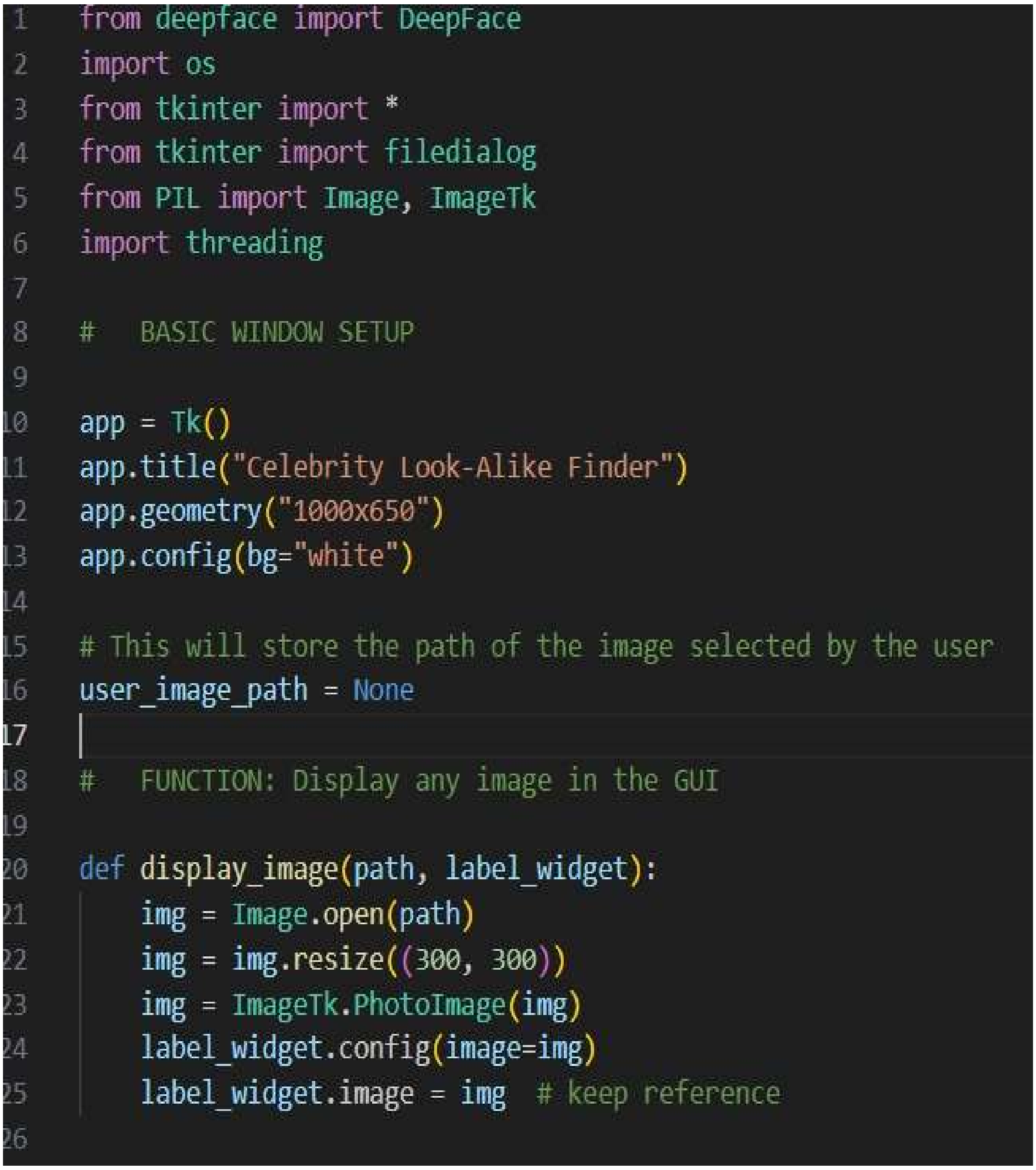
Framework / Model

- **VGG-Face Model** (used via DeepFace)
-

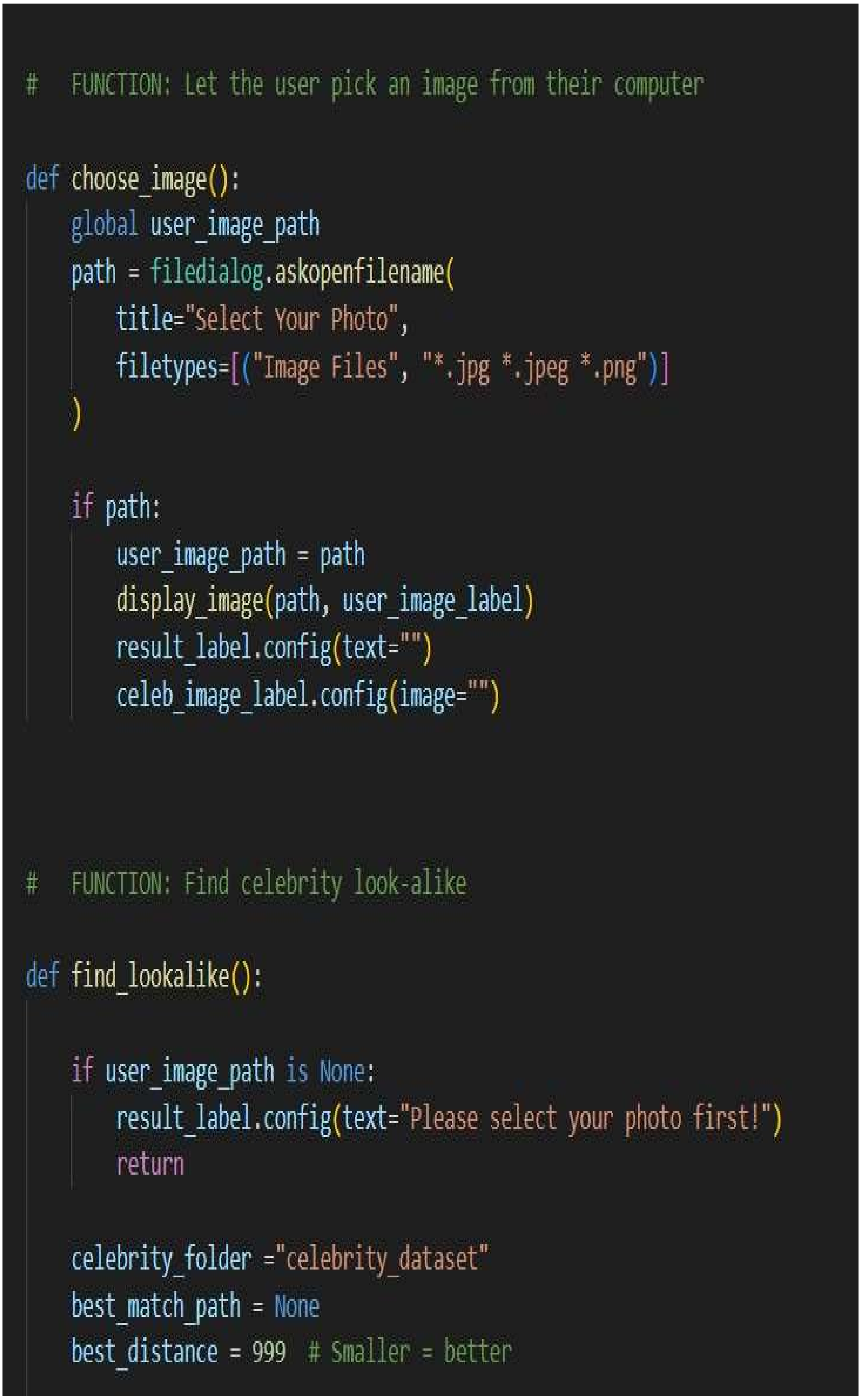
Platform

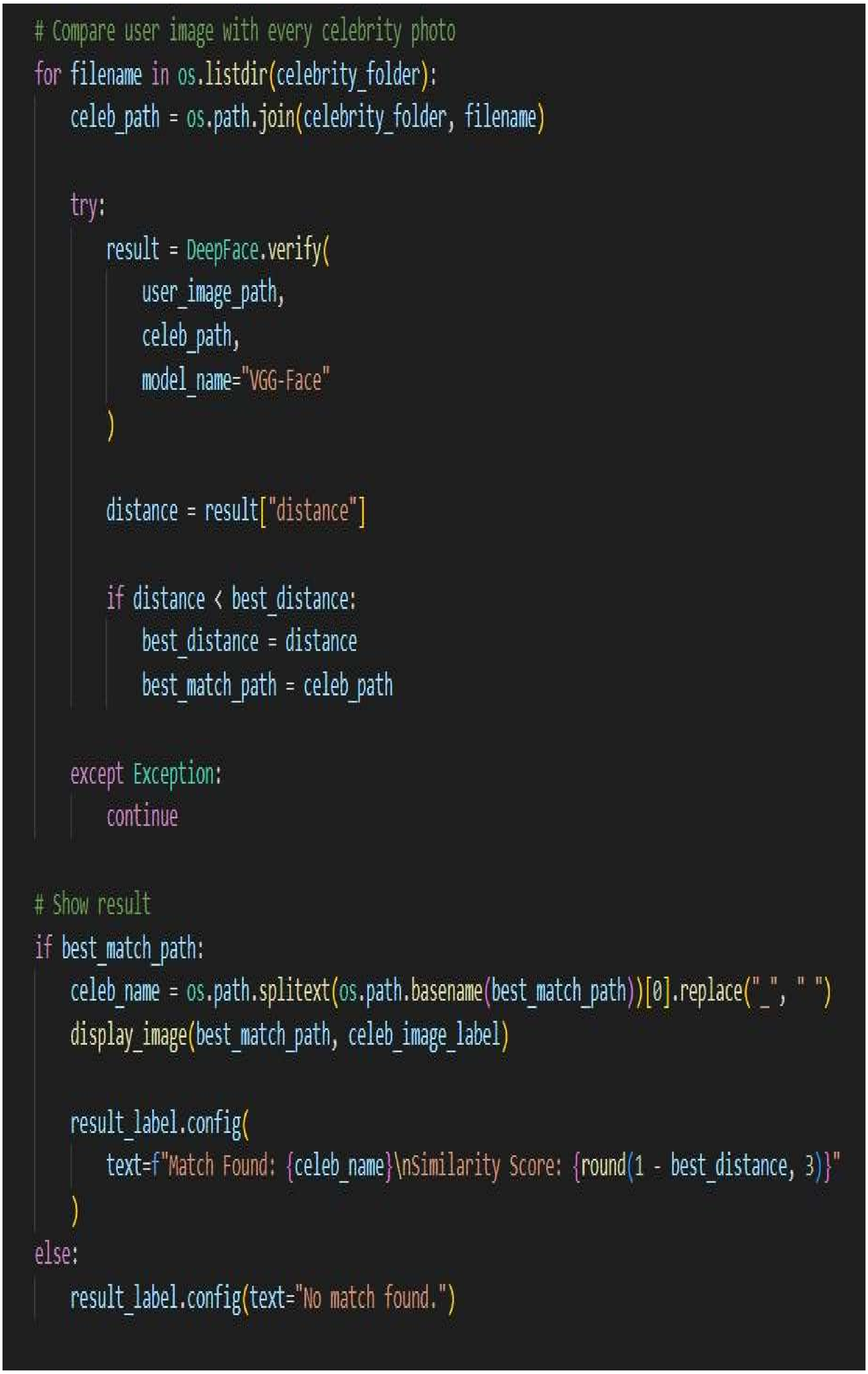
- Desktop Application (VS CODE)

COMPLETE CODE:

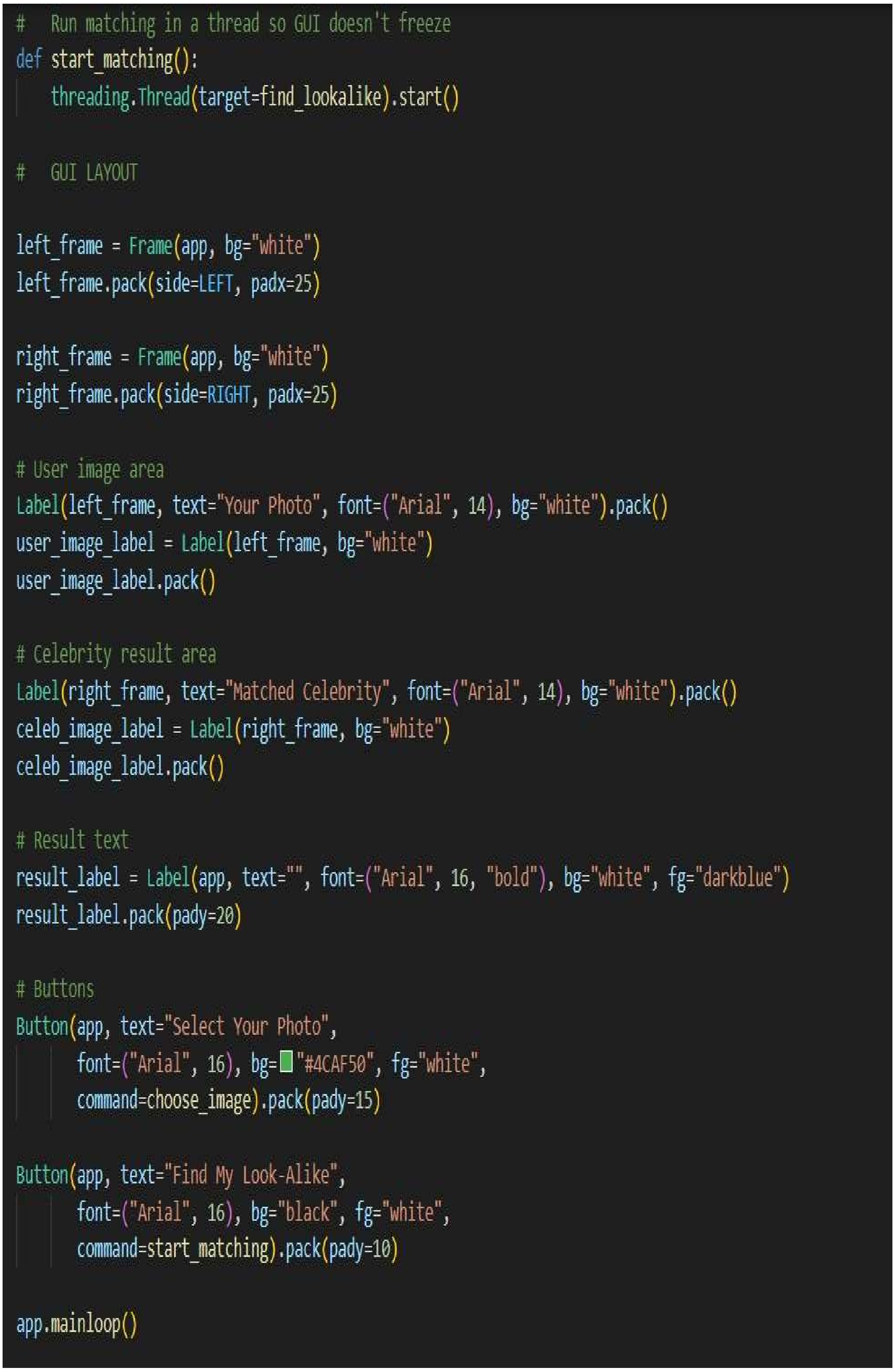


```python
from deepface import DeepFace
import os
from tkinter import *
from tkinter import filedialog
from PIL import Image, ImageTk
import threading

#   BASIC WINDOW SETUP

app = Tk()
app.title("Celebrity Look-Alike Finder")
app.geometry("1000x650")
app.config(bg="white")

# This will store the path of the image selected by the user
user_image_path = None

#   FUNCTION: Display any image in the GUI

def display_image(path, label_widget):
    img = Image.open(path)
    img = img.resize((300, 300))
    img = ImageTk.PhotoImage(img)
    label_widget.config(image=img)
    label_widget.image = img  # keep reference

```


```python
#   FUNCTION: Let the user pick an image from their computer

def choose_image():
    global user_image_path
    path = filedialog.askopenfilename(
        title="Select Your Photo",
        filetypes=[("Image Files", "*.jpg *.jpeg *.png")]
    )

    if path:
        user_image_path = path
        display_image(path, user_image_label)
        result_label.config(text="")
        celeb_image_label.config(image="")



#   FUNCTION: Find celebrity look-alike

def find_lookalike():

    if user_image_path is None:
        result_label.config(text="Please select your photo first!")
        return

    celebrity_folder ="celebrity_dataset"
    best_match_path = None
    best_distance = 999  # Smaller = better
```



```
# Compare user image with every celebrity photo
for filename in os.listdir(celebrity_folder):
    celeb_path = os.path.join(celebrity_folder, filename)

    try:
        result = DeepFace.verify(
            user_image_path,
            celeb_path,
            model_name="VGG-Face"
        )

        distance = result["distance"]

        if distance < best_distance:
            best_distance = distance
            best_match_path = celeb_path

    except Exception:
        continue

# Show result
if best_match_path:
    celeb_name = os.path.splitext(os.path.basename(best_match_path))[0].replace("_", " ")
    display_image(best_match_path, celeb_image_label)

    result_label.config(
        text=f"Match Found: {celeb_name}\nSimilarity Score: {round(1 - best_distance, 3)}"
    )
else:
    result_label.config(text="No match found.")
```



```python
#   Run matching in a thread so GUI doesn't freeze
def start_matching():
    threading.Thread(target=find_lookalike).start()

#   GUI LAYOUT

left_frame = Frame(app, bg="white")
left_frame.pack(side=LEFT, padx=25)

right_frame = Frame(app, bg="white")
right_frame.pack(side=RIGHT, padx=25)

# User image area
Label(left_frame, text="Your Photo", font=("Arial", 14), bg="white").pack()
user_image_label = Label(left_frame, bg="white")
user_image_label.pack()

# Celebrity result area
Label(right_frame, text="Matched Celebrity", font=("Arial", 14), bg="white").pack()
celeb_image_label = Label(right_frame, bg="white")
celeb_image_label.pack()

# Result text
result_label = Label(app, text="", font=("Arial", 16, "bold"), bg="white", fg="darkblue")
result_label.pack(pady=20)

# Buttons
Button(app, text="Select Your Photo",
       font=("Arial", 16), bg="#4CAF50", fg="white",
       command=choose_image).pack(pady=15)

Button(app, text="Find My Look-Alike",
       font=("Arial", 16), bg="black", fg="white",
       command=start_matching).pack(pady=10)

app.mainloop()
```

INPUT AND OUTPUT

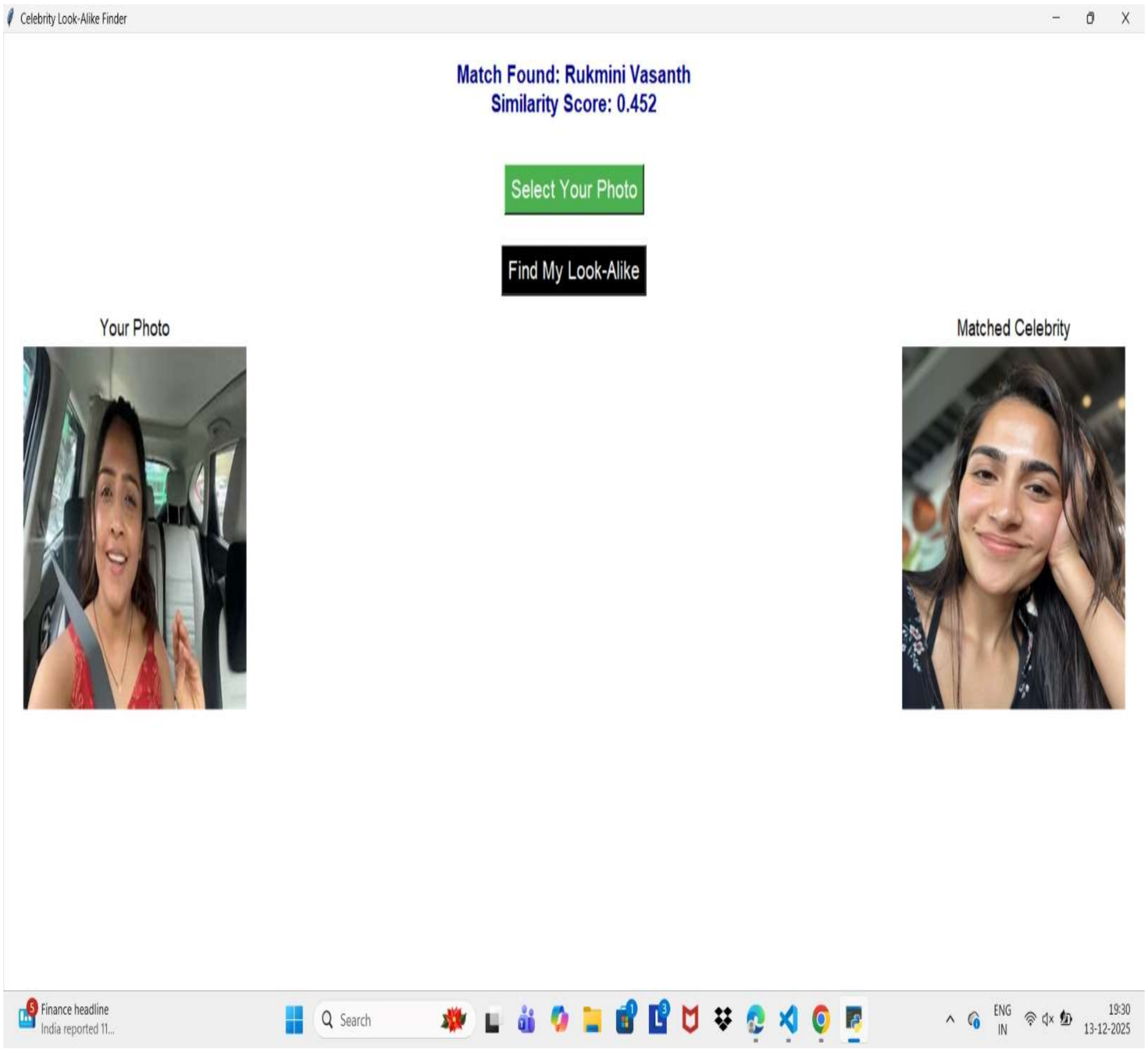
Input

- User uploads an image (JPG / PNG format).
- Celebrity dataset folder containing multiple celebrity images.

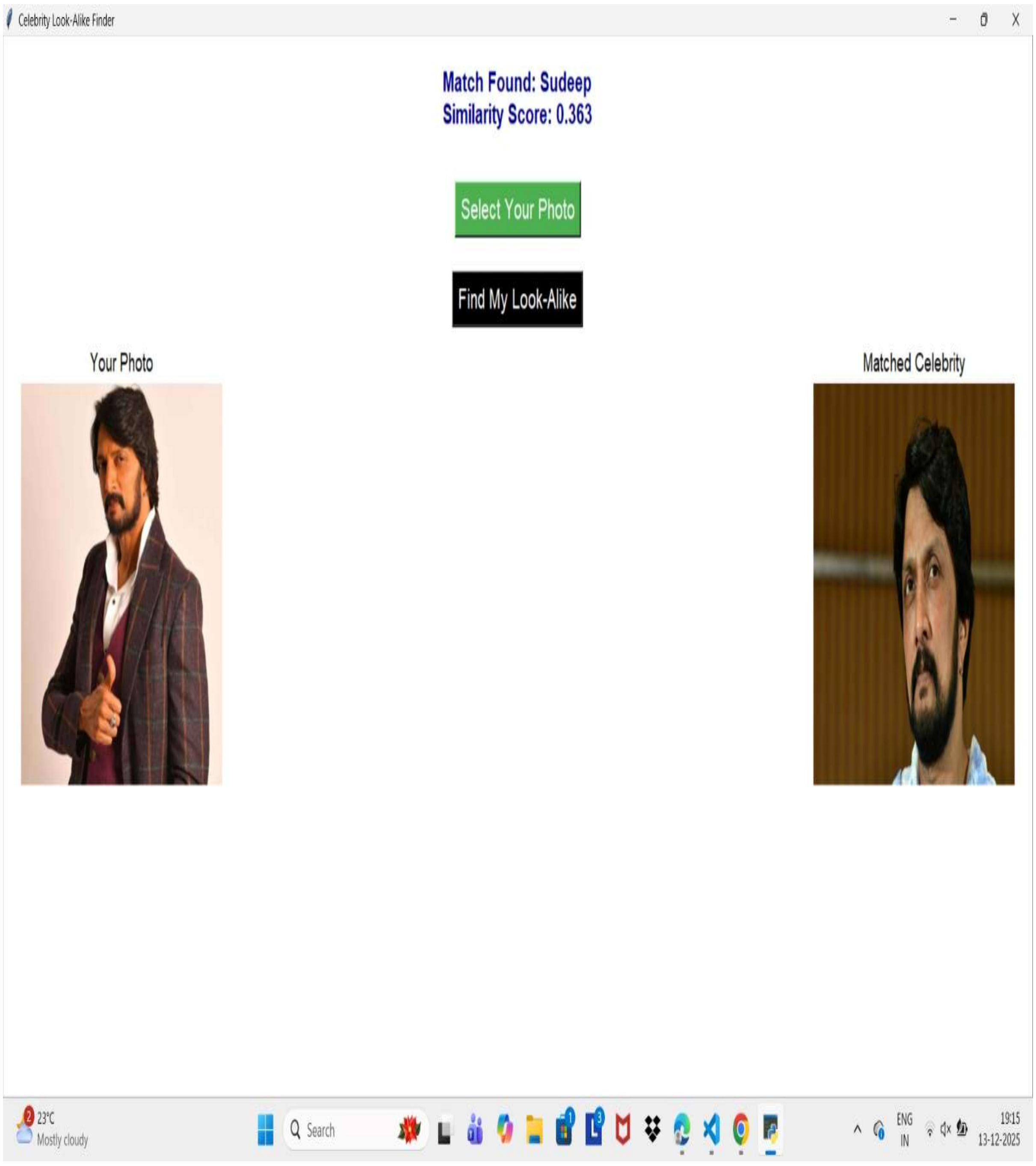
Output

- Display of the user’s uploaded image.
- Display of the matched celebrity image.
- Celebrity name.
- Similarity score.

SCREENSHOT OF THE IMAGE SELECTION AND RESULT DISPLAY



Match Found: Rukmini Vasanth
Similarity Score: 0.452



Celebrity Look-Alike Finder

Match Found: Sudeep
Similarity Score: 0.363

CONCLUSION

The Celebrity Look-Alike Finder successfully demonstrates the use of deep learning–based face recognition in a real-world application. By integrating DeepFace with a user-friendly GUI, the system provides accurate and engaging results. This project highlights the potential of artificial intelligence in entertainment and image analysis applications.