UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

BSD2513 ARTIFICIAL INTELLIGENCE

GROUP PROJECT

GROUP NAME: GlamCam

TITLE: Real-Time Image Analysis for Makeup Recommendations

PREPARED FOR: DR KU MUHAMMAD NAIM KU KHALIF

STUDENT ID	NAME	SECTION
SD22002	ALMIRA DAMIA BINTI SYAHNIZAM	02G
SD22030	TUAN NURSHAFIEKA WAHIDA BINTI TUAN NADIN	02G
SD22011	NOR MIMI AZURA BINTI HUZAIMI	01G
SD22003	NUR ATIEKA RAFIEKAH BINTI RAZAK	01G
SD22066	SHAHIRA BINTI MOHAIDEEN MEERA	02G

TABLE OF CONTENTS

1.0 EXECUTIVE SUMMARY

1.1 Description of the selected project

In today's beauty landscape, finding the right makeup that complements one's skin tone and style preferences can be a time-consuming and often confusing task. The Makeup Recommendation System is designed to simplify this process by providing personalized makeup suggestions using modern technology. This project aims to merge the ease of digital solutions with the traditional charm of beauty consultations, helping individuals effortlessly find makeup that enhances their natural beauty.

At the core of the Makeup Recommendation System is its ability to analyze images and real-time video to determine a person's skin undertone, a critical factor in choosing suitable makeup products. By capturing either video or still images, the system can assess the user's skin and identify whether their undertone is warm, cool, or neutral. This analysis is then used to recommend makeup products such as foundations and lipsticks that best match the user's unique skin characteristics. The application is built using Python and incorporates various tools for creating a user-friendly interface, processing images, and managing data to ensure a seamless experience.

Beyond product recommendations, the system allows users to input detailed personal information, including age, gender, skin type, and makeup preferences. This information is stored securely and used to refine and personalize recommendations further, ensuring that each user receives advice that is accurate and tailored to their needs. Users can also give feedback on their experiences, which helps in continuously improving the system. Additionally, the application offers data visualization features, enabling users to explore trends and preferences through easy-to-understand charts and graphs.

The Makeup Recommendation System revolutionizes the way people choose makeup by automating the process and providing instant, personalized recommendations. Unlike traditional methods, which often involve trial and error, this system offers a reliable and convenient way to discover products that suit one's individual needs and tastes. As the system continues to develop, with plans to integrate more advanced machine learning algorithms and expand its database of

products, it aims to make personalized beauty advice more accessible and effective for everyone, ensuring that the perfect makeup choices are just a click away.

1.2 Problem to be solved

The Makeup Recommendation System tackles the challenge of finding the right makeup that matches individual preferences and skin tones. One of its main goals is to simplify the process of choosing makeup items from a wide range of options available in stores. The system aims to make this process easier by using advanced technology to give personalized recommendations tailored to each person's unique characteristics and likes.

A big problem the Makeup Recommendation System solves is the confusion people face when trying to choose makeup that suits their skin tone and personal style. Many individuals, especially students at UMP, find it challenging to determine which products are best for them, leading to frustration and wasted time and money. This system offers customized suggestions based on factors like skin undertone, age, and makeup preferences, helping users avoid guesswork and feel more confident in their choices. Additionally, it aims to make personalized beauty advice accessible to everyone, not just those who can afford traditional consultations. By providing a user-friendly platform, the system ensures that people from all backgrounds can easily access tailored beauty tips.

For students at University Malaysia Pahang (UMP), the Makeup Recommendation System can help with makeup choices while they are busy with studies. Sometimes, it is hard for students to pick the right makeup for themselves, especially for events or presentations. This system gives personalized makeup ideas based on what each student likes and what suits their skin. It makes choosing makeup easier, so students can feel good about how they look without spending a lot of time figuring it out.

Ultimately, the Makeup Recommendation System wants to help people make smart decisions about their beauty routines and feel good about themselves. By using technology to offer personalized guidance and suggestions, it hopes to change the way people shop for makeup, making it simpler and more enjoyable for everyone.

1.3 Basic Description of the data selected

The Makeup Recommendation System is implemented as a desktop application using Python's Tkinter library for the GUI, OpenCV for real-time video capture and image processing, and Pandas for data handling. The GUI is designed to be intuitive, with a main window that provides easy access to various functionalities including skin tone detection, user detail entry, data visualization and feedback collection.

The skin tone detection feature utilizes the user's webcam to capture real-time video. The system processes these video frames to isolate and analyze skin regions, determining the user's skin undertone (warm, cool, or neutral) based on color properties in the LAB color space. Once the skin tone is identified, the system provides tailored makeup recommendations and saves them in a CSV file for future reference.

Users can input their details such as age, gender, skin type, skin tone, allergies, preferred makeup style, and eye color. This information is stored in a CSV file and can be used to refine the system's recommendations over time. Additionally, the system includes a feedback module where users can provide input on their experience, which is essential for continuous improvement of the application.

The data visualization component of the Makeup Recommendation System offers users an engaging way to explore their makeup preferences and the system's suggestions. Through interactive charts and graphs, users can see trends and patterns in the recommendations provided over time. This feature not only enhances the user experience by providing a clear understanding of how recommendations evolve but also helps users make more informed decisions about their

makeup choices. The visual insights derived from the data can highlight seasonal trends, popular products among similar users, and the effectiveness of certain recommendations.

2.0 SUMMARY OF THE PROJECT CONTEXT AND OBJECTIVES

2.1 Summary of the Project Context

The project focuses on developing an innovative makeup recommendation system that can overcome the consumer problem when they want to find the right makeup product that suits their preferences and skin tone. This system analyzes user preferences, and skin tone to provide product suggestions by applying knowledge of computer vision and AI algorithms. Unlike the previous makeup recommendation system that may focus on one thing only, either lipstick or foundation or eyes, our project aims to develop a comprehensive system that recommends both types of makeup products.

The main goal of our project is to enhance the makeup shopping experience by analyzing images to recommend suitable makeup products that match each user's preferences. This study helps in saving time to purchase the product makeup. Through this innovative approach, we seek to revolutionize the way consumers explore and purchase makeup products, making it easier and more enjoyable for them to find the perfect lipstick and foundation matches.

2.2 OBJECTIVES

1. To develop and implement an algorithm capable of swiftly and accurately analyzing images to recommend suitable makeup products. .
2. To design a user-friendly interface that allows for easy interaction with the real-time makeup recommendation system.
3. To validate the effectiveness and reliability of the recommendation system through extensive testing.

4. To develop a real-time visualization tool that provides insights into the distribution of foundation shades, skin tones, skin types, and makeup preferences among Malaysian people.

3.0 METHODOLOGY

3.1 Data Collection and Preparation

The data collection process begins with real-time image acquisition, facilitated by the application interface. Users are prompted to use the camera function within the application to take pictures, ensuring immediate and direct data capture. To maintain the quality of the images, the system provides specific guidelines on image resolution, lighting conditions, and pose. These guidelines help users capture optimal images that are suitable for accurate analysis. By adhering to these recommendations, users contribute high-quality input that enhances the effectiveness of the makeup recommendations.

Before collecting any data, the application ensures that users are fully informed about how their images will be used. Explicit consent is obtained from each user, emphasizing transparency and ethical data handling practices. To protect user privacy, the system implements robust measures such as anonymizing data and securely storing the images. These privacy measures are crucial for maintaining user trust and complying with data protection regulations. Ensuring that users feel secure and confident in sharing their images is a key priority for the system.

Once the images are captured, they undergo a thorough preprocessing phase to prepare them for analysis. This begins with face detection, where algorithms such as OpenCV and Dlib are employed to locate faces within the images. Focusing the analysis on the relevant part of the image ensures precision in subsequent steps. The images are then normalized to adjust for consistent lighting, scale, and orientation. Techniques like histogram equalization and affine transformations are applied to achieve this consistency. Additionally, the detected face regions

are cropped and resized to a standard size suitable for detailed analysis, ensuring uniformity across all input data.

The final stage of data preparation involves feature extraction, which is critical for generating accurate makeup recommendations. The system analyzes the skin undertone by extracting color information from the identified facial regions. This undertone analysis is essential for recommending appropriate makeup shades, such as foundation and lipstick, that match the user's skin tone. The processed and standardized images, combined with the extracted features, provide a robust dataset that enables the system to offer precise and personalized makeup recommendations. This comprehensive data preparation ensures that the system can deliver high-quality, tailored suggestions to each user.

3.2 Coding of Project without GUI

In this project we use Jupyter as our main software in developing this system to help us run a real time analysis. In order to achieve the system output, we installed a few libraries into jupyter to ensure that our coding can run smoothly and allow us to do this makeup recommendation system efficiently. The libraries included are as follows:

- Import cv2: Imports the OpenCV library, which is used for computer vision and image processing tasks.
- Import numpy as np: Imports the NumPy library, allowing for efficient numerical operations on arrays and matrices.
- Import csv: Imports the CSV module, enabling the reading and writing of CSV (Comma-Separated Values) files.
- Import os: Imports the OS module, providing functions for interacting with the operating system, such as file and directory manipulation.
-from tkinter import *: Imports all components from Tkinter, a standard Python library for creating graphical user interfaces.
- from tkinter import messagebox: Imports the messagebox module from Tkinter, used for displaying pop-up message boxes in a GUI.

- from PIL import Image, ImageTk: Imports the Image and ImageTk modules from the Python Imaging Library (PIL), allowing for image processing and display in Tkinter GUIs.

```
import cv2
import numpy as np
import csv
import os
from tkinter import *
from tkinter import messagebox
from PIL import Image, ImageTk
```

In this system we implemented a few functions in order to make it a user-friendly interface as it includes a function for recognition of face in real-time, entering the user details, drawing out a visualization based on user details and a feedback comment. The recognition function is our main focus in this project as it gives recommendations of lipstick and foundations to people by detecting their skin tone colour.

Recognition Function

The followings are the coding and explanation of this recognition function:

```
def start_camera():
    global cap
    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
        messagebox.showerror("Error", "Could not open webcam.")
        return
    update_frame()
```

The function in the picture below is for starting the camera, it turns on the camera of the laptop or any devices it is used on, when clicked.

```
def update_frame():
    ret, frame = cap.read()
    if ret:
        frame = cv2.flip(frame, 1)
        img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(img)
        img = ImageTk.PhotoImage(img)
        lbl_video.imgtk = img
        lbl_video.configure(image=img)
        lbl_video.after(10, update_frame)
```

The start camera calls this update_frame function in order to showcase a frame of our picture so that we can see our face before capturing the shot.

```
def capture_snapshot():
    global snapshot
    ret, frame = cap.read()
    if ret:
        snapshot = cv2.flip(frame, 1)
        detect_and_display_recommendations(snapshot)
```

The capture_snapshot is for capturing the picture displayed in the frame for detection of skin tone.

```
def detect_skin_tone(image):
    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_skin = np.array([0, 30, 60], dtype=np.uint8)
    upper_skin = np.array([35, 255, 255], dtype=np.uint8)
    mask = cv2.inRange(hsv_image, lower_skin, upper_skin)
    skin = cv2.bitwise_and(image, image, mask=mask)
    return skin
```

The detect_skin_tone function receives the image and identifies what type of skin tone does the person in the picture have based on the intensity of skin colour of the person's face.

```
def get_skin_undertone(image):
    skin = detect_skin_tone(image)
    lab_image = cv2.cvtColor(skin, cv2.COLOR_BGR2LAB)
    mean_a = np.mean(lab_image[:, :, 1])
    mean_b = np.mean(lab_image[:, :, 2])
    if mean_a > 140 and mean_b > 140:
        return "Warm"
    elif mean_a < 120 and mean_b < 120:
        return "Cool"
    else:
        return "Neutral"
```

The get_skin_undetone receives the intensity from the image, and will print out what skin tone the person has such as warm,cool and neutral.

```
def recommend_lipstick(undertone):
    recommendations = {
        "Warm": ["Peach", "Coral", "Orange"],
        "Cool": ["Berry", "Mauve", "Cherry"],
        "Neutral": ["Rose", "Pink", "Nude"]
    }
    return recommendations.get(undertone, ["Shade not found"])
```

The recommed_lipstick is for recommendation of lipstick shades, as different skin undertones are suitable with certain colour of lipsticks. If the person has warm undertones the suitable shades are peach,coral or orange, if cool undertones the suitable shades are berry, mauve orcherry, lastly if it has neutral undertones the suitable shades are rose,pink or nude. These shades are based on basic ones which can be found in many cosmetic stores.

```
def recommend_foundation(undertone):
    recommendations = {
        "Warm": ["Golden Beige", "Honey", "Caramel"],
        "Cool": ["Porcelain", "Ivory", "Sand"],
        "Neutral": ["Buff", "Nude", "Beige"]
    }
    return recommendations.get(undertone, ["Shade not found"])
```

The recommend_foundation is for recommendation of foundation shades, it receives the undertone colour. Warm undertones, the suitable shades are golden beige, honey or caramel, while for cool undertones the suitable shades are porcelain ,ivory or sand . The neutral undertone is suitable with buff ,nude or beige shades of foundation.

```
def save_recommendations_to_csv(undertone, lipstick_recommendations, foundation_recommendations, filename="recommendations.csv"):
    file_exists = os.path.isfile(filename)
    with open(filename, mode='a', newline='') as file:
        writer = csv.writer(file)
        if not file_exists:
            writer.writerow(["Undertone", "Lipstick Recommendations", "Foundation Recommendations"])
        writer.writerow([undertone, ", ".join(lipstick_recommendations), ", ".join(foundation_recommendations)])
```

The save_recommendation_to_csv function is for saving the recommendation that has been detected during the real time analysis into a csv file named recommendation.

```
def detect_and_display_recommendations(image):
    undertone = get_skin_undertone(image)
    lipstick_recommendations = recommend_lipstick(undertone)
    foundation_recommendations = recommend_foundation(undertone)
    recommendations_text.set(f"Skin Undertone: {undertone}\n"
                             f"Lipstick: {', '.join(lipstick_recommendations)}\n"
                             f"Foundation: {', '.join(foundation_recommendations)}")
    save_recommendations_to_csv(undertone, lipstick_recommendations, foundation_recommendations)
    messagebox.showinfo("Recommendations Saved", "Recommendations saved to recommendations.csv")
```

The detect_and_display_recommendation function helps in printing out the recommendation of lipstick and foundation shade, also the skin undertone colour. It pops up a messagebox that notifies the user the recommendations are saved in the csv file.

```
def quit_app():
    if cap:
        cap.release()
    root.destroy()
```

The quit_app function is for quitting the recognition part of this system, after clicking on quit it will end the function of recognition.

User Details Function

The followings are the coding and explanation of this user details function, where the user details will be used for visualization:

```
tk.Label(form_frame, text="Age", bg="#fae7ce", font=custom_font).grid(row=0, column=0, sticky="e", padx=5, pady=5)
age_entry = tk.Entry(form_frame)
age_entry.grid(row=0, column=1, padx=5, pady=5)
```

This displays age with a text box besides it in order for the user to enter their age.

```
tk.Label(form_frame, text="Gender", bg="#fae7ce", font=custom_font).grid(row=1, column=0, sticky="e", padx=5, pady=5)
gender_var = tk.StringVar(details_window)
tk.OptionMenu(form_frame, gender_var, "Female", "Male").grid(row=1, column=1, padx=5, pady=5)
```

This displays gender with a drop down menu besides it in order for the user to choose their gender.

```
tk.Label(form_frame, text="Skin Type", bg="#fae7ce", font=custom_font).grid(row=2, column=0, sticky="e", padx=5, pady=5)
skin_type_var = tk.StringVar(details_window)
tk.OptionMenu(form_frame, skin_type_var, "Oily", "Dry", "Combination", "Sensitive").grid(row=2, column=1, padx=5, pady=5)
```

This displays skin type with a drop down menu beside it for the user to choose their skin type whether it is oily,dry,combination or sensitive.

```
tk.Label(form_frame, text="Skin Tone", bg="#fae7ce", font=custom_font).grid(row=3, column=0, sticky="e", padx=5, pady=5)
skin_tone_var = tk.StringVar(details_window)
tk.OptionMenu(form_frame, skin_tone_var, "Fair", "Light", "Medium", "Tan", "Deep").grid(row=3, column=1, padx=5, pady=5)
```

This displays skin tone with a drop down menu beside it , so the user can choose their skin tone whether it is fair, light, medium,tan or deep.

```
tk.Label(form_frame, text="Allergies/Sensitivities", bg="#fae7ce", font=custom_font).grid(row=4, column=0, sticky="e", padx=5, pady=5)
allergies_entry = tk.Entry(form_frame)
allergies_entry.grid(row=4, column=1, padx=5, pady=5)
```

This displays allergies/sensitivities with a text box beside it , allowing the user to enter if they have any sensitivities on their face such as fragrance products.

```
tk.Label(form_frame, text="Preferred Makeup Style", bg="#fae7ce", font=custom_font).grid(row=5, column=0, sticky="e", padx=5, pady=5)
preferred_style_var = tk.StringVar(details_window)
tk.OptionMenu(form_frame, preferred_style_var, "Natural", "Glam", "Bold").grid(row=5, column=1, padx=5, pady=5)
```

This displays preferred makeup style with a drop down menu beside it , so the user can choose their style whether it is neutral, glam or bold.

```
tk.Label(form_frame, text="Eye Color", bg="#fae7ce", font=custom_font).grid(row=6, column=0, sticky="e", padx=5, pady=5)
eye_color_entry = tk.Entry(form_frame)
eye_color_entry.grid(row=6, column=1, padx=5, pady=5)
```

This displays eye colour with a text box beside it , allowing the user to enter the colour of their eyes.

```
tk.Button(form_frame, text="Save", command=save_details, font=custom_font).grid(row=7, column=0, columnspan=2, pady=10)
```

This displays a save button, when it is clicked the user details will be saved.

```
def save_details():
    details = {
        'age': age_entry.get(),
        'gender': gender_var.get(),
        'skin_type': skin_type_var.get(),
        'skin_tone': skin_tone_var.get(),
        'allergies': allergies_entry.get(),
        'preferred_style': preferred_style_var.get(),
        'eye_color': eye_color_entry.get()
    }
    save_user_details(details)
    messagebox.showinfo("Details Saved", "Users details saved to userdetails.csv")
```

The save_details gets the user details that have been entered , and will call the save_user_details as it will help in saving the information. A messagebox will appear indicating that the details have been saved.

```
def save_user_details(details):
    with open('userdetails.csv', mode='a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(details.values())
```

The save_user_details will save the information of the user after clicking on save button, it will save the details in a csv file named userdetails.

Visualization Function

The followings are the coding and explanation of visualization unction, where it uses the data from user details function:

```
def create_ui(self):
    # Create buttons for different visualizations
    age_skin_button = tk.Button(self.root, text="Age and Skin Type Distribution", command=self.plot_age_skin_type_distribution)
    age_skin_button.pack(pady=10)

    style_button = tk.Button(self.root, text="Preferred Makeup Style Distribution", command=self.plot_preferred_style_pie_chart)
    style_button.pack(pady=10)
```

The create_ui function above is for displaying buttons where it allows the user to choose which visualization they would like to display.

```
data = pd.read_csv('userdetails.csv', names=['age', 'gender', 'skin_type', 'skin_tone', 'allergies', 'preferred_style', 'eye_color'])
```

This function reads the data from userdetails.csv file, and the names of variables for visualization purposes.

```
if data is not None and not data.empty:
    # Create a new window for the visualization
    age_skin_window = tk.Toplevel(self.root)
    age_skin_window.title("Age and Skin Type Distribution")
    age_skin_window.geometry("800x600")

    plt.figure(figsize=(12, 8))
    sns.countplot(x='age', hue='skin_type', data=data, palette='Set2')
    plt.title('Age and Skin Type Distribution')
    plt.xlabel('Age')
    plt.ylabel('Count')
    plt.legend(title='Skin Type')

    # Create a canvas for the plot
    canvas = FigureCanvasTkAgg(plt.gcf(), master=age_skin_window)
    canvas.draw()
    canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
else:
    print("No data to display.")
```

This function is for creating a bar chart visualization with a relationship of age and skin type. It displays a title of Age and Skin Type Distribution where it counts the number of users with each skin type and displays it in a bar chart, the x axis is the age of the user.

```
# Count the occurrences of each preferred style
style_counts = data['preferred_style'].value_counts()

fig, ax = plt.subplots(figsize=(8, 8))
ax.pie(style_counts, labels=style_counts.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Paired(range(len(style_counts))))
ax.set_title('Preferred Makeup Style Distribution')
```

This function helps in creating a pie chart to see the distribution of preferred_style. It counts the percentage of users that have each preferred style : glam, bold or natural , and labels the percentage on each sector of the pie. The preferred style is also labeled based on the percentage.

Feedback Function

The followings are the coding and explanation of feedback function:

```
def give_feedback(self):
    feedback_window = tk.Toplevel(self.root)
    feedback_window.title("Feedback")
    feedback_window.configure(bg="#fae7ce")

    feedback_label = tk.Label(feedback_window, text="We'd love to hear your feedback!", font=("Helvetica", 16),bg="#fae7ce")
    feedback_label.pack(pady=10)

    feedback_text = tk.Text(feedback_window, height=10, width=50)
    feedback_text.pack(pady=10)

    submit_button = tk.Button(feedback_window, text="Submit",bg="#ffbc85", command=lambda: self.save_feedback(feedback_text.get("1.0", "end-1c")))
    submit_button.pack(pady=10)
```

The give_feedback function is for accepting feedback from the user in text.

```
def save_feedback(self, feedback, filename="feedback.csv"):
    with open(filename, mode='a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow([feedback])
    messagebox.showinfo("Feedback Saved", "Thank you for your feedback!")
```

The save_feedback function is to save the feedback from the user into a csv file named feedback. It also appears a messagebox that indicates the feedback is saved.

3.3 Adding the GUI to Coding

After completing the code without GUI, we include the function of GUI into it to make it more interactive and user-friendly. Before continuing to do so, we installed a few additional libraries to use in the GUI . The libraries are as follows:

- Import pandas as pd: Imports the Pandas library, which is used for data manipulation and analysis, particularly with data structures like DataFrames.
- Import matplotlib.pyplot as plt: Imports the pyplot module from Matplotlib, providing a MATLAB-like interface for creating static, interactive, and animated visualizations in Python.
- Import seaborn as sns: Imports the Seaborn library, which is built on Matplotlib and provides a high-level interface for creating attractive and informative statistical graphics.

- From matplotlib.backends.backend_tkagg import FigureCanvasTkAgg: Imports FigureCanvasTkAgg from the Matplotlib backends, which allows embedding Matplotlib plots into Tkinter graphical user interfaces.

```
from PIL import Image, ImageTk
import tkinter as tk
from tkinter import messagebox, ttk
import cv2
import numpy as np
import os
import csv
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
```

The following is the code for adding a menu or usually known as a home page:

```
class MakeupRecommendationSystem:
    def __init__(self, root):
        self.root = root
        self.root.title("Makeup Recommendation System")
        self.root.geometry("800x600")

        self.create_ui()

        # Load the background image
        self.original_image = Image.open("homepage.jpeg")

        # Create a label to display the background image
        self.background_label = tk.Label(self.root)
        self.background_label.place(relwidth=1, relheight=1)
```

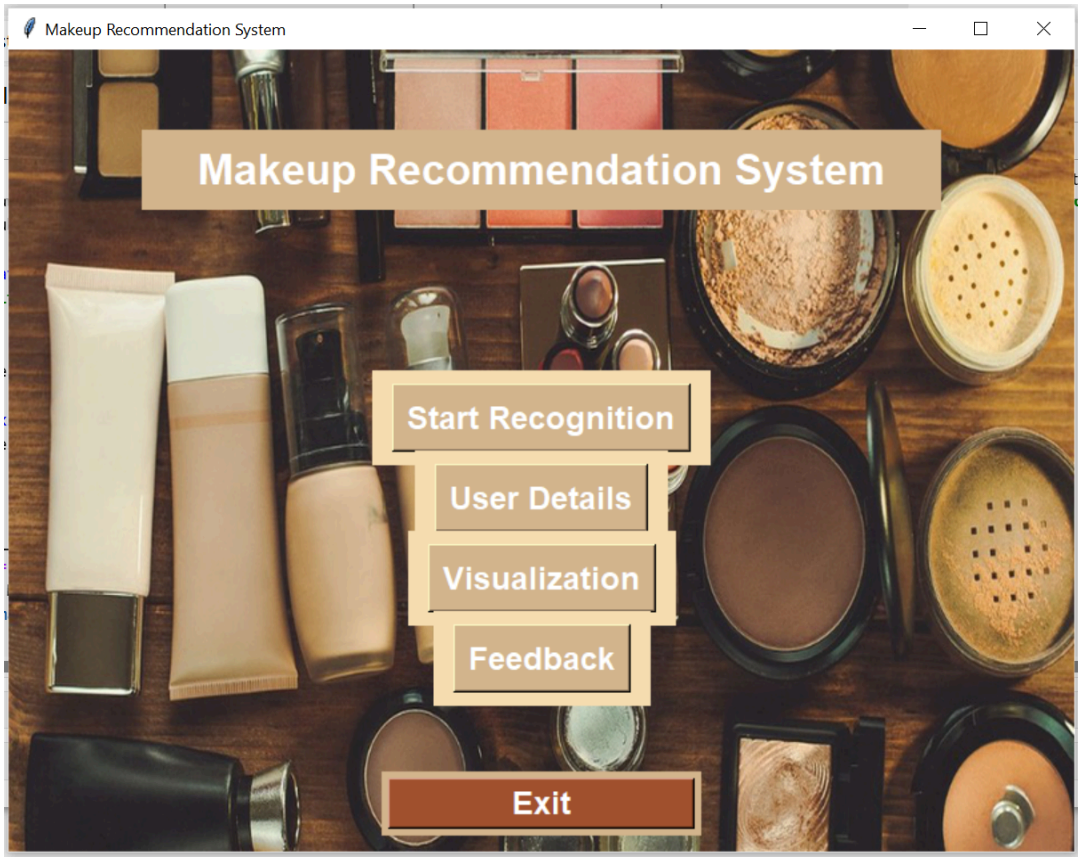
```
    # Bind the <Configure> event to the resize_image function
    self.root.bind("<Configure>", self.resize_image)

    # Create the main heading frame
    self.heading_frame = tk.Frame(self.root, bg="#D2B48C", bd=5)
    self.heading_frame.place(relx=0.5, rely=0.1, relwidth=0.75, relheight=0.1, anchor='n')

    self.heading_label = tk.Label(self.heading_frame, text="Makeup Recommendation System",
                                  font=("Helvetica", 24, "bold"), bg="#D2B48C", fg="white")
    self.heading_label.place(relwidth=1, relheight=1)

    # Create frames for different functionalities
    self.create_function_frame("Start Recognition", 0.4, self.start_recognition)
    self.create_function_frame("User Details", 0.5, self.user_details)
    self.create_function_frame("Visualization", 0.6, self.create_visualization_window)
    self.create_function_frame("Feedback", 0.7, self.give_feedback)

    # Create the exit button frame
    self.exit_frame = tk.Frame(self.root, bg="#D2B48C", bd=5)
    self.exit_frame.place(relx=0.5, rely=0.9, relwidth=0.3, relheight=0.08, anchor='n')

    self.exit_button = tk.Button(self.exit_frame, text="Exit", font=("Helvetica", 18, "bold"), bg="#A0522D",
                                 fg="white", command=self.root.quit)
    self.exit_button.place(relwidth=1, relheight=1)
    self.cap = None
```

This function is to add the buttons to the homepage so that it is easy to navigate through the system. The buttons that were added are start recognition,user details,visualization and feedback. Each of the buttons are connected with a command, where when clicked on it , the function of command will appear.

```
if __name__ == "__main__":
    root = tk.Tk()
    app = MakeupRecommendationSystem(root)
    root.mainloop()
```

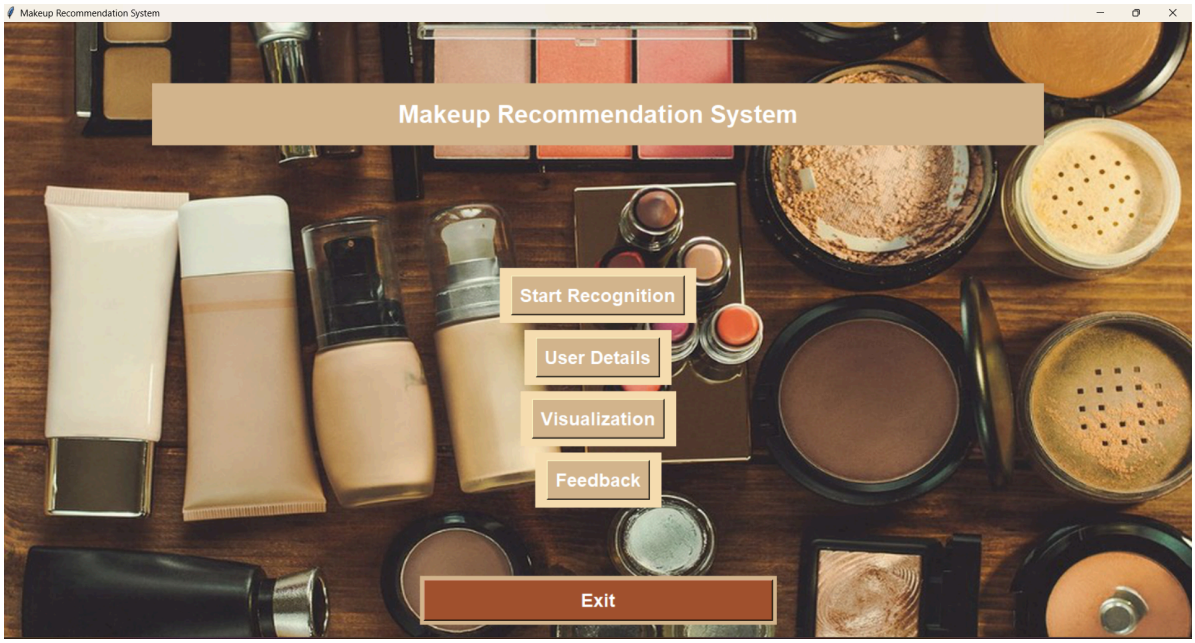
This code sets up and runs a Tkinter-based GUI application if the script is executed as the main program

Output of GUI:

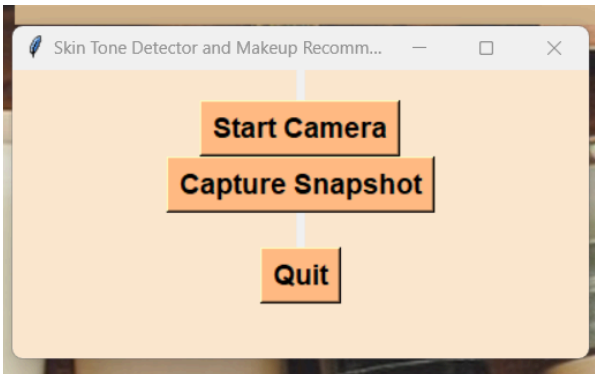


4.0 RESULTS AND DISCUSSION

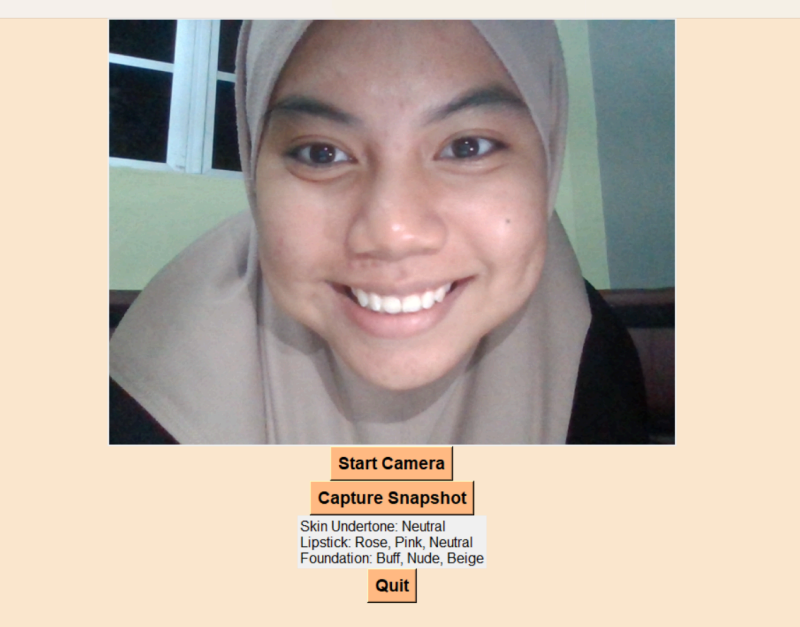
When the user runs the code, user would see the GUI main menu with the title Makeup Recommendation system. The user can select from five buttons in this main menu: Start Recognition, User Details, Visualization and Feedback and Exit.



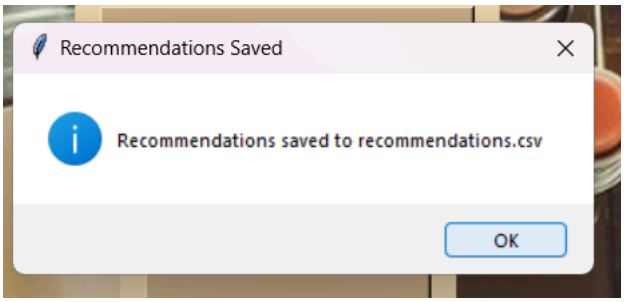
Since there are five buttons, the user may test the system by selecting any of them. "Start Recognition" is chosen to be tested for the first example. Window Skin tone and Detector Makeup Recommendation will appear.



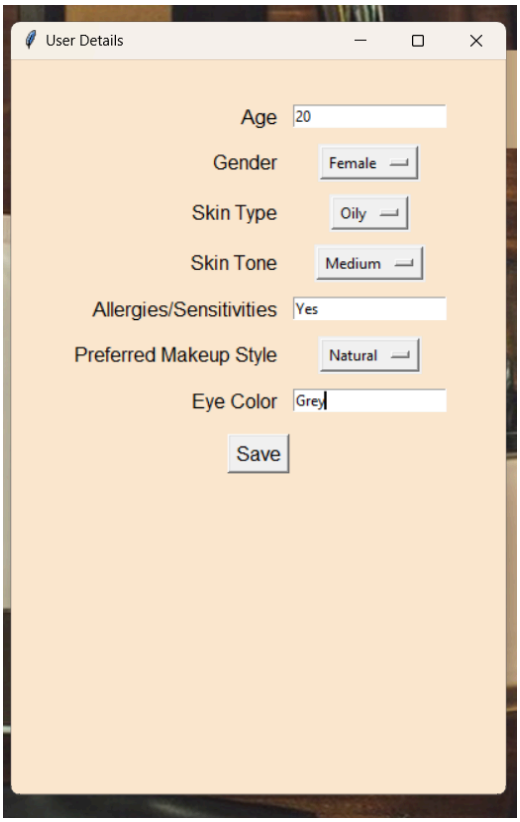
When the user clicks the "Start Camera" button, the camera will activate, allowing the user to capture a live feed of their face. Recommendation will appear when clicking “Capture Snapture”. Users can know their skin undertone, tone lipstick and foundation that suits them. Clicking the "Quit" button will close this smaller window or the entire application.



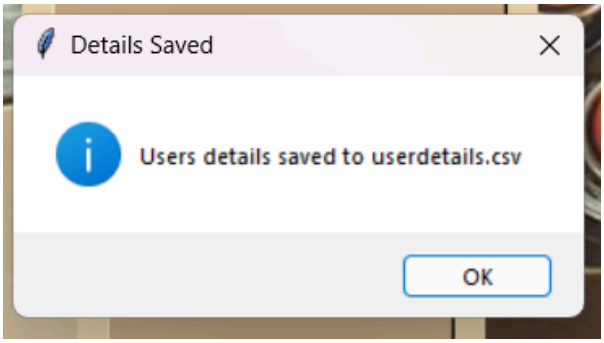
Makeup recommendations will be saved in the `recommendations.csv` file. Every time the user snaps a photo, the CSV file will be updated with new recommendations, continuing to store the data in the same CSV file.



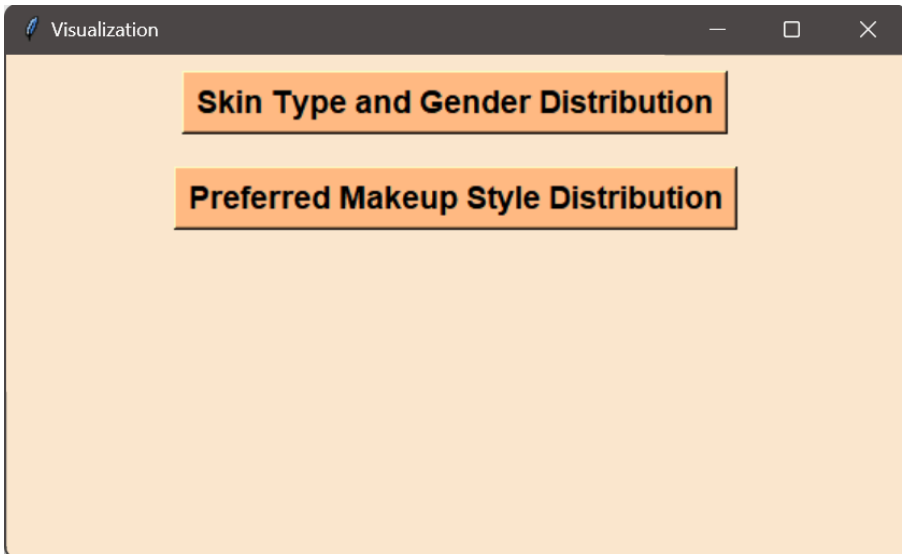
When the user clicks the "User Details" button, a window titled "User Details" will appear, allowing the user to input personal information. The user can enter their age, gender, skin type, skin tone, allergies or sensitivities, preferred makeup style, and eye color. After filling out these details, the user can save the information by clicking the "Save" button



User details will be saved in the `userdetails.csv` file. Every time the user input the details, the CSV file will be updated with new details, continuing to store the data in the same CSV file.

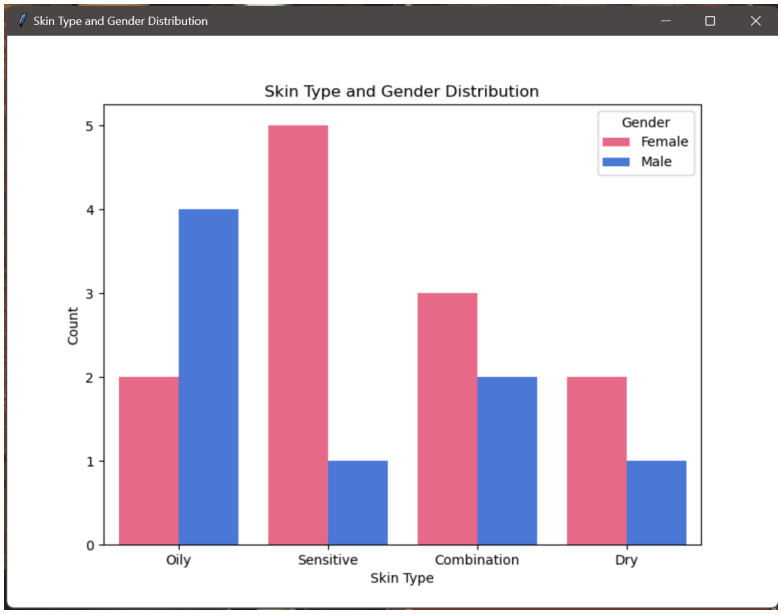


For our ecosystem, we can do some analysis by clicking on the visualization. There are two analyses that can be observed which are first, the relationship between skin type and gender distribution. Second, are preferred makeup style distribution. The data is from user details that have been saved to a CSV file.

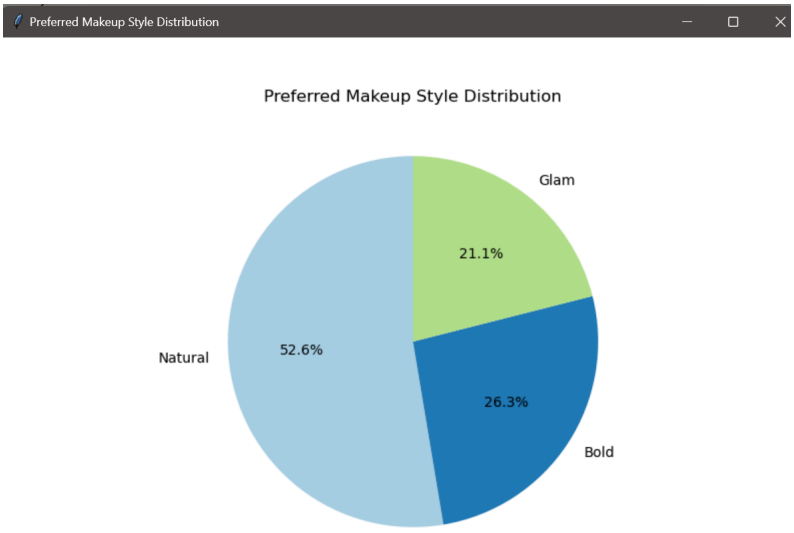


Based on the graph, we can observe that the majority of users are female, with females outnumbering males across three skin type categories. The highest frequency of female users is within the sensitive skin type category. It is due to many females wear makeup throughout the day or frequently change their makeup products, which can lead to skin sensitivity since exposure to various fragrances and chemicals used in these products.

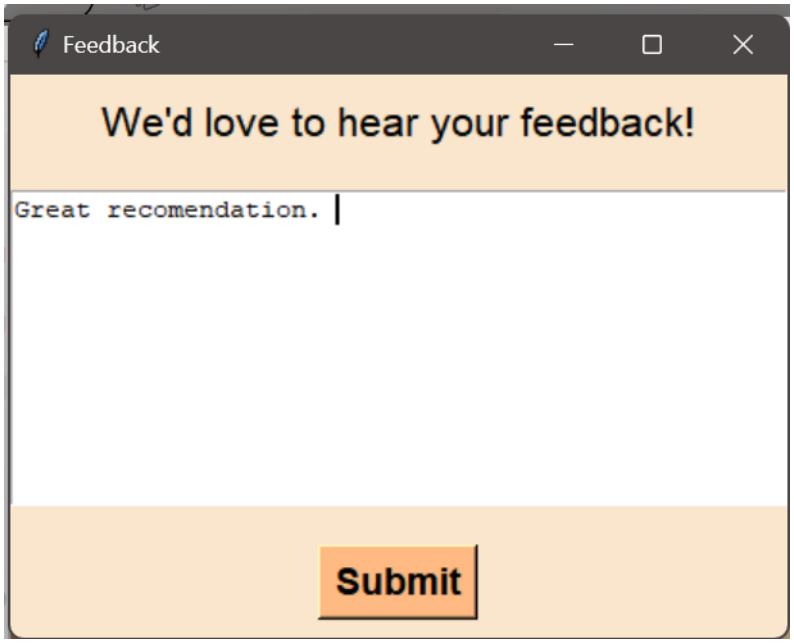
For males, the most common skin type observed is oily, with four individuals into this category. This is likely due to increased physical activities, which can lead to higher levels of sweating and subsequently, oilier skin.



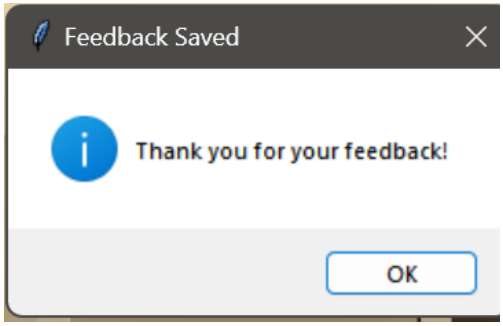
The graph is a pie chart that shows the preferred makeup style distribution. The majority, 52.6% of users, prefer a natural makeup style. This preference is likely because the natural style is suitable for everyday use, such as at university. The second most popular makeup style is bold, favored by 26.3% of users. This indicates that a significant number of users enjoy a more pronounced and striking appearance. The least preferred style is glam, with 21.1% of users opting for this look. Although it's the least popular among the three categories, a considerable number of users still favor a glamorous and elaborate makeup style.



Customers can give the feedback via click on feedback at the home page. They can give any good or bad reviews related to our makeup recommendation system. After filling out these details, the user can save the information by clicking the "Save" button. This can help us to improve our system time by time



User feedback will be saved in the `feedback` file. Every time the user gives the feedback, the CSV file will be updated with new details, continuing to store the data in the same CSV file.



5.0 CONCLUSION

In this study, several limitations were encountered, such as the limited time available for development and testing. Despite these constraints, the Makeup Recommendation System shows great potential for further enhancements. Future improvements could include adding filters to visualize makeup looks, allowing users to see how different products would appear on their faces in real-time. Additionally, integrating products available at UMP stores would make the system more practical and convenient for students.

We successfully developed an algorithm capable of swiftly and accurately analyzing images to recommend suitable makeup products. The system includes a user-friendly interface with intuitive buttons, facilitating easy interaction with the real-time makeup recommendation features. We validated the effectiveness and reliability of the recommendation system through extensive testing with friends who have different skin tones, demonstrating its accuracy and inclusivity. Additionally, a real-time visualization tool was developed to provide insights into the distribution of foundation shades, skin tones, skin types, and makeup preferences among Malaysian people. Overall, the Makeup Recommendation System helps users find suitable makeup products, enhancing the shopping experience and ensuring personalized beauty advice is accessible to everyone.

REFERENCES

Chung, K. (2013). Effect of facial makeup style recommendation on visual sensibility. *Multimedia Tools and Applications*, *71*(2), 843–853. https://doi.org/10.1007/s11042-013-1355-6

Jeong Bin Whang, J. H. (2021). The effect of Augmented Reality on purchase intention of beauty products:. *Elsevier*. Retrieved from https://pdf.sciencedirectassets.com/271680/1-s2.0-S0148296321X00082/1-s2.0-S0148296321002939/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEIv%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJHMEUCIQDQbzbk9TzdO2dowJd52OzVSW%2BRTaNhDn3X5KeD2m32PwIgTnHFBxN3jl

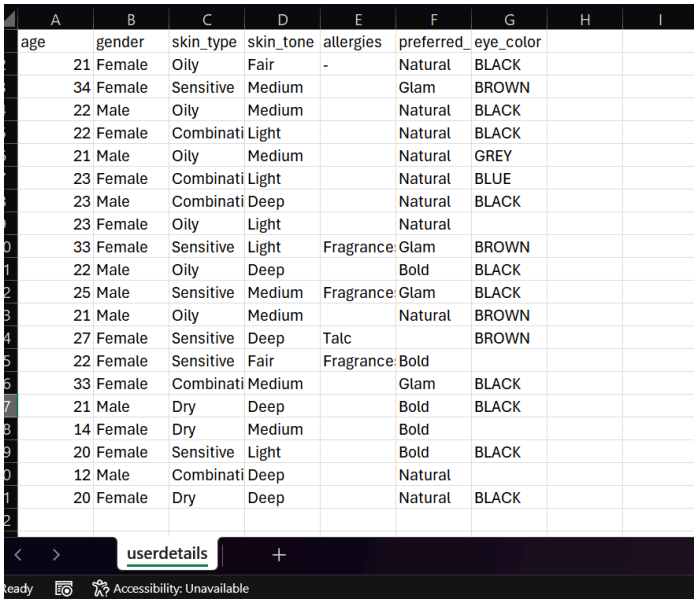
Mai1902. (01 Jan, 2022). *Github*. Retrieved from Machine Learning VirtualTtryon : https://github.com/Mai1902/MachineLearning-VirtualTryon

Sokal Kanstantsin, S. K. (n.d.). High-Quality AR Lipstick Simulation via Image Filtering Technique. -.srivatsan-ramesh. (05 March, 2016). *GitHub*. Retrieved from Virtual Makeup: https://github.com/srivatsan-ramesh/Virtual-Makeup

Foundation for each skin tone - Bing. (n.d.). Bing. https://www.bing.com/images/search?view=detailV2&ccid=46x42sr%2B&id=12FC50E00277B75C4D8C5760FBD6EA0958BDD7E1&thid=OIP.46x42sr-m5gC8r2Z_AsT0QHaHZ&mediaurl=https%3A%2F%2Fstatic.wixstatic.com%2Fmedia%2Fe81de0_fffb337a06a0434fae156078143ee4ff~mv2.jpg%2Fv1%2Ffit%2Fw_750%252Ch_749%252Cal_c%252Cq_80%2Ffile.jpg&exph=749&expw=750&q=foundation+for+each+skin+tone&simid=607989764392189040&form=IRPRST&ck=87377CEBC3B3EDDA1E3D96E48F943923&selectedindex=3&itb=0&ajaxhist=0&ajaxserp=0&vt=0&sim=11

APPENDIX

1. Example of user detail file.



	A	B	C	D	E	F	G	H	I
	age	gender	skin_type	skin_tone	allergies	preferred_	eye_color		
	21	Female	Oily	Fair	-	Natural	BLACK		
	34	Female	Sensitive	Medium		Glam	BROWN		
	22	Male	Oily	Medium		Natural	BLACK		
	22	Female	Combinati	Light		Natural	BLACK		
	21	Male	Oily	Medium		Natural	GREY		
	23	Female	Combinati	Light		Natural	BLUE		
	23	Male	Combinati	Deep		Natural	BLACK		
	23	Female	Oily	Light		Natural			
0	33	Female	Sensitive	Light	Fragrance	Glam	BROWN		
1	22	Male	Oily	Deep		Bold	BLACK		
2	25	Male	Sensitive	Medium	Fragrance	Glam	BLACK		
3	21	Male	Oily	Medium		Natural	BROWN		
4	27	Female	Sensitive	Deep	Talc		BROWN		
5	22	Female	Sensitive	Fair	Fragrance	Bold			
6	33	Female	Combinati	Medium		Glam	BLACK		
7	21	Male	Dry	Deep		Bold	BLACK		
8	14	Female	Dry	Medium		Bold			
9	20	Female	Sensitive	Light		Bold	BLACK		
0	12	Male	Combinati	Deep		Natural			
1	20	Female	Dry	Deep		Natural	BLACK		
2									

userdetails

eady Accessibility: Unavailable

2. Google Drive Link : https://drive.google.com/drive/folders/17Wge3o1AXWqSpAwJ0vIl7rC9jHK8GCDl?usp=sharing

- Full coding of this project
- Home page picture