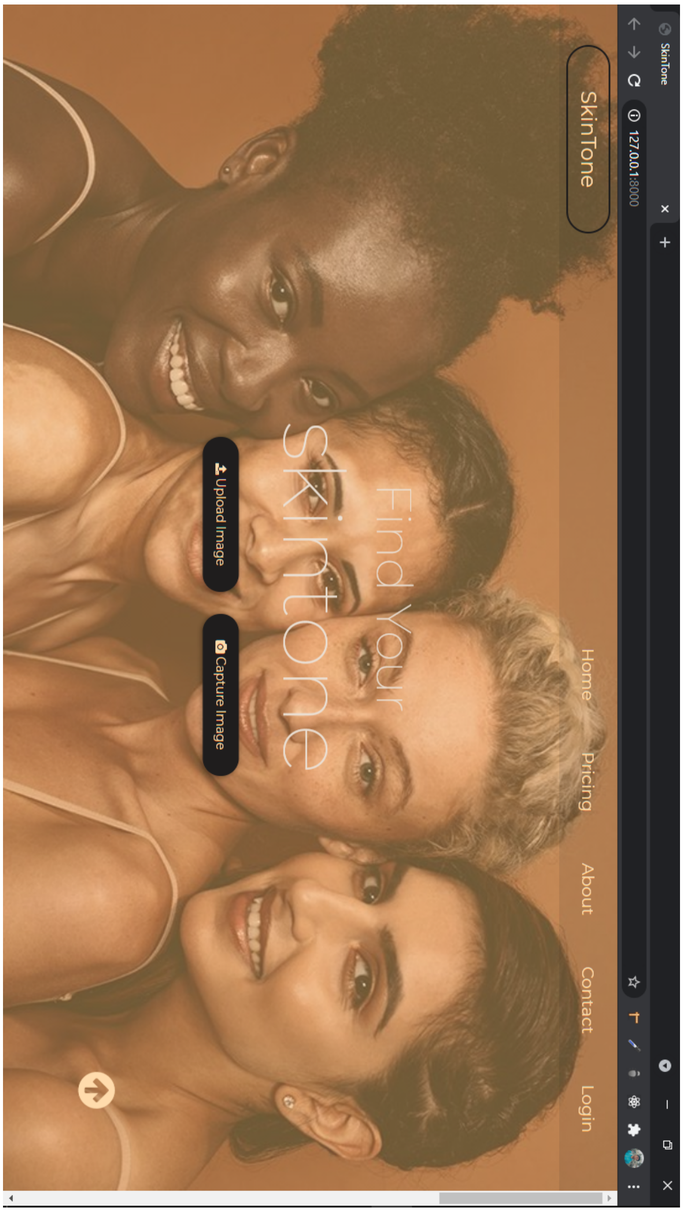


PROJECT REPORT

HUMAN SKIN TONE DETECTION

B9|VERZEO PROJECT|20/05/2021

ACKNOWLEDGEMENT

The internship opportunity we had with Verzeo was a great chance for learning and professional development. Therefore, We consider ourselves as a very lucky individual as we were provided with an opportunity to be a part of it. We are also grateful for having a chance to meet so many wonderful people and professionals who led us through this internship period.

Bearing in mind previous we are using this opportunity to express our deepest gratitude and special thanks to the verzeo team of the verzeo who in spite of being extraordinarily busy with his duties, took time out to hear, guide and keep us on the correct path and allowing us to carry out our project and extending during the training.

We perceive this opportunity as a big milestone in our career development. We will strive to use gained skills and knowledge in the best possible way, and we will continue to work on their improvement, in order to attain desired career objectives. Hope to continue cooperation with all of you in the future.

OUR TEAM – B9

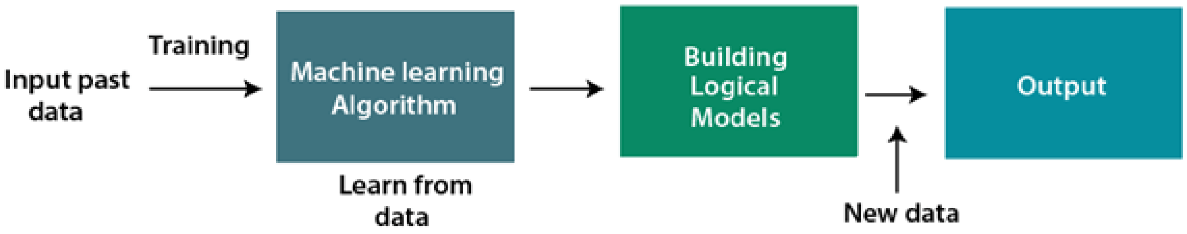
1. Sabareesh S
2. Yasir Hussain
3. Rutvij Choudhary
4. Khushboo Verma
5. Prakash
6. Manish Roy
7. M Ritik Kumar
8. Krishnash Gaur
9. Brinda Potluri
10. Ritika Sadawarte
11. Bhavana Mamillapalli
12. Rashmi Kulkarni
13. Adish Raipure
14. M. Naresh
15. Karan
16. Rishab Bhatt
17. Neha Behera
18. N Sai Nithish Kumar
19. Sahana Kulkarni
20. Ramagiri Tharun
21. Sakshi Kamble
22. Mansi Joshi
23. Akshata Savalagi
24. Rineetha G
25. Priyanka Priyadarshini
26. Shreedeep Nair
27. Purandhar Sri Sai
28. Meghana S Naidu

PROJECT AT GLANCE

INTRODUCTION

Machine learning (ML) is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so.

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.



Web development is the building and maintenance of web sites. It’s the work that happens behind the scenes to make a website look great,work and perform well for a seamless user experience.

Web developers or devs do this by using a variety of coding languages. The languages they use depends on the types of tasks they are performing and the platforms on which they are working.

Web development is the work involved in developing a website for the internet(World Wide Web) or an intranet(a private network). Web development can range from developing a simple single static page of plain text to complex web applications, electronic businesses, and social network services. A more comprehensive list of tasks to which web development commonly refers, may include Web engineering, Web design, Web content development, client liaison, client-side/server-side scripting, Web server and network security configuration, and e-commerce development.

PROJECT DESCRIPTION

- **PROJECT OBJECTIVE** - A web portal to detect the skin tone (i.e, Fair,Mild or Dark) from a human face image.
- **PROJECT DESCRIPTION** - The project aims to create a webpage which will take image as an input; Use image processing techniques to process the image as per requirement; To detect whether the skin tone in the image is Fair, Mild or Dark.

Skin detection is the process of finding skin-colored pixels and regions in an image or a video. This process is typically used as a preprocessing step to find regions that potentially have human faces and limbs in images . Skin image recognition is used in a wide range of image processing applications like face recognition, skin disease detection, gesture tracking and human-computer interaction . The primary key for skin recognition from an image is the skin color. But color cannot be the only deciding factor due to the variation in skin tone according to different races. Other factors such as the light conditions also affect the results. Therefore, the skin tone is often combined with other cues like texture and edge features. This is achieved by breaking down the image into individual pixels and classifying them into skin colored and non-skin colored . One simple method is to check if each skin pixel falls into a defined color range or values in some coordinates of a color space. There are many skin color spaces like RGB, HSV, YCbCr, YIQ, YUV, etc. that are used for skin color segmentation.

RGB color space is widely used and is normally the default color space for storing and representing digital images. We can get any other color space from a linear or non-linear transformation of RGB . The RGB color space is the color space used by computers, graphics cards and monitors or LCDs. It consists of three components, red, green and blue, the primary colors. Any color can be obtained by mixing the three base colors. Depending on how much is taken from each base color, any color can be created. Reversing this technique, a specific color can be broken down into its red, blue and green components. These values can be used to find out similar colored pixels from the image. Normalized RGB is a representation that is easily obtained from the RGB values by a simple normalization procedure. A remarkable property of this representation is that for matte surfaces, while ignoring ambient light, normalized RGB is invariant (under certain assumptions) to changes of surface orientation relatively to the light source.

OPENCV provides advance image processing in which the image pixels can be sorted accordingly and skin can be extracted. After the skin is extracted the dominant color is extracted using a function defined in program. After the skin color is extracted it is compared with the threshold values provided in program to find out the type of skin color. This provided skin color is based on mere trial and error methods. After the color is compared the image, mask and skin color is displayed using matplotlib. Opencv provides a good library for performing masking functions. However face detection model cannot be applied using opencv

$$r = \frac{R}{R+G+B}$$

$$g = \frac{G}{R+G+B}$$

$$b = \frac{B}{R+G+B}$$

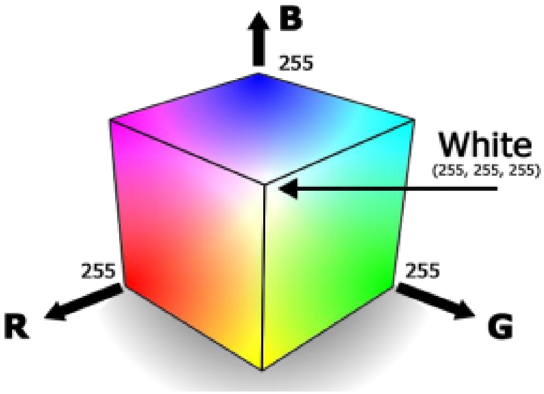


Fig. 1.RGB Color Model

LANGUAGES USED:

HTML stands for Hyper Text Markup Language. It is the standard markup language for creating web pages. In Hyper Text Markup Language “Hypertext” refers to the hyperlinks that an HTML page may contain. “Markup Language” refers to the way tags are used to define the page layout and elements within the page.

It describes the structure of a web page. It consists of a series of elements. It can be assisted by technologies such as Cascading Style Sheets(CSS) and scripting languages such as javaScript. HTML elements that tell the browser how to display the content. HTML elements label pieces of content such as “this is a heading”, “this is a paragraph”, “this is a link” etc.

HTML elements are the building blocks of HTML pages. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

CSS stands for Cascading Style Sheets with an emphasis placed on “Style”. It is the style sheet language used for describing the presentation of a document written in a markup language such as HTML. It describes how html elements are to be displayed on screen, paper, or in the other media. It saves a lot of work. It is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes. It is a cornerstone technology of the World Wide Web, alongside HTML and javaScript. It can control the layout of multiple web pages all at once. External stylesheets are stored in CSS files. It is designed to enable the separation of presentation and content, including layout, colors, and fonts. In CSS there are 3 types. They are:

1. Inline CSS
2. Internal CSS
3. External CSS

Python is a high-level, general-purpose and a very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting edge technology in Software Industry.

LIBRARIES USED:

1. OPENCV -

Computer vision is a process by which we can understand the images and videos how they are stored and how we can manipulate and retrieve data from them. Computer Vision is the base or mostly used for Artificial Intelligence. Computer-Vision is playing a major role in self-driving cars, robotics as well as in photo correction apps.

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as Numpy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

2. NUMPY -

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

NumPy was created in 2005 by Travis Oliphant. It is an open-source library and you can use it freely. NumPy stands for Numerical Python.

3. KMEANS -

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

"It is an iterative algorithm that divides the unlabelled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties."

The algorithm takes the unlabelled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

4. IMUTILS –

A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV and both Python 2.7 and Python 3.

5. MATPLOTLIB –

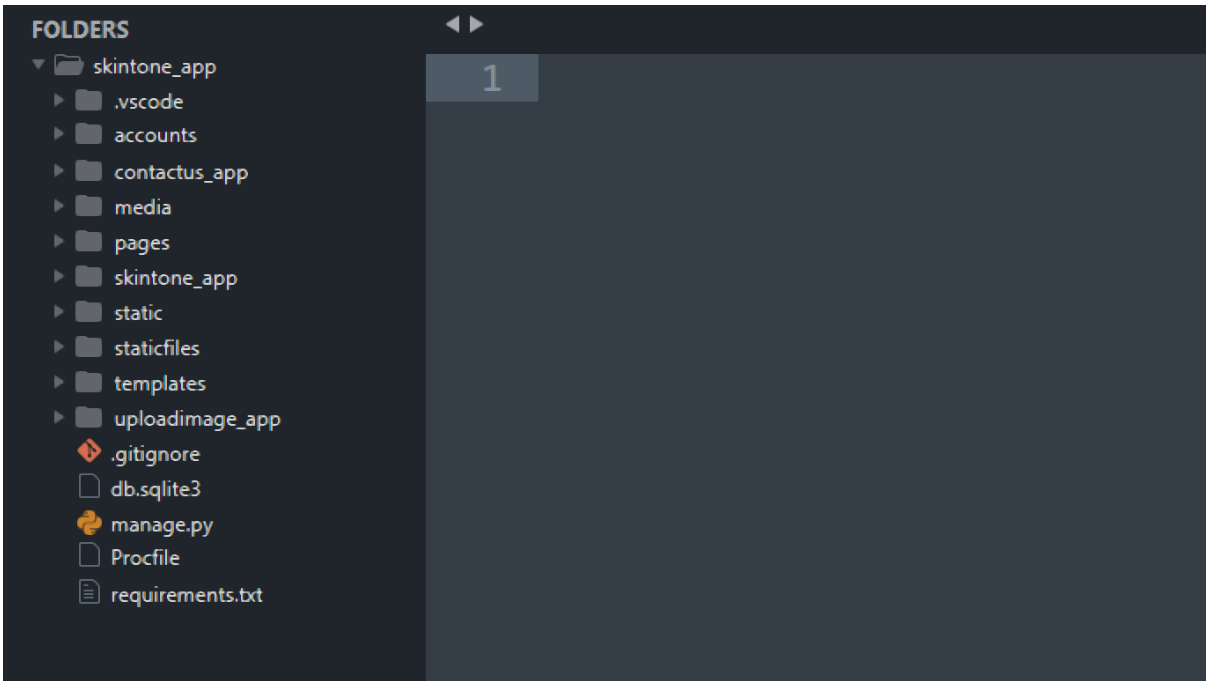
Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

PROJECT CODE

Folder Structure

In folder structure we have Four APPs that are listed below:

1. Accounts
2. Pages
3. Upload images app
4. Contact us app



```
FOLDERS
skintone_app
    .vscode
    accounts
    contactus_app
    media
    pages
    skintone_app
    static
    staticfiles
    templates
    uploadimage_app
    .gitignore
    db.sqlite3
    manage.py
    Procfile
    requirements.txt
```

1. **Accounts**

Any web app which will provide some sort of services and that work on user data is called user specific, so for security reasons and restricting access, app must contain a user authentication system. So, for that reason we have provided a secure registration system with email OTP verification that provides a better way for registering user with their real credentials. The logic for providing these features are listed below:

- **#urls.py - It handles all the Urls related to account sections.**

```
from django.urls import path
from . import views
urlpatterns = [
path('register', views.register, name='register'),
path('verify', views.verify, name='verify'),
path('login', views.login, name='login'),
path('logout', views.logout, name='logout'),
path('dashboard', views.dashboard, name='dashboard'),
]
```

- **#views.py - It handles all the views related to account sections.**

```
from django.contrib import messages
from django.shortcuts import redirect, render
from django.contrib.auth.models import User, auth
from django.core.mail import EmailMultiAlternatives
from django.template.loader import render_to_string
from django.utils import html
from django.utils.html import strip_tags
from django.conf import settings
from django.contrib.auth.decorators import login_required
from .models import Profile
import random
```

- **# register route**

```
def register(request):
if request.user.is_authenticated:
return redirect('home')
else:
```

```
if request.method == 'POST':
username = request.POST.get('username')
email = request.POST.get('email')
password = request.POST.get('password')
confirm_password = request.POST['confirm_password']
if password == confirm_password:
if User.objects.filter(username=username).exists():
messages.info(request, 'User Already exist')
return redirect('register')
else:
if User.objects.filter(email=email).exists():
messages.info(request, 'Email Already exist')
return redirect('register')
else:
request.session['username'] = username
request.session['email'] = email
request.session['password'] = password
otp = []
num = str(random.randint(1000, 9999))
for x in num:
otp.append(x)
context = {'name': username, 'auth_otp': otp}
html_content = render_to_string(
'accounts/email.html', context)
text_content = strip_tags(html_content)

send_mail = EmailMultiAlternatives(
"Account email verification",
text_content,
settings.EMAIL_HOST_USER,
[email]
)
send_mail.attach_alternative(html_content, 'text/html')
send_mail.send()
# user_obj = User(username=username, email=email)
# user_obj.set_password(password)
# user_obj.save()
profile = Profile(
user=email, auth_otp=num)
profile.save()
return redirect('verify')
return render(request, 'accounts/register.html')
```

- **# login route**

```
def login(request):
if request.user.is_authenticated:
return redirect('home')
else:
if request.method == 'POST':
username = request.POST['username']
password = request.POST['password']
user = auth.authenticate(username=username, password=password)
if user is not None:
auth.login(request, user)
return redirect('home')
else:
messages.info(request, "Invalid Username or Password")
return redirect('login')
return render(request, 'accounts/login.html')
```

- **# verify route**

```
def verify(request):
if request.user.is_authenticated:
return redirect('home')
else:
if request.method == 'POST':
count = Profile.objects.filter(
user=request.session['email']).count()
profile = Profile.objects.filter(
user=request.session['email'])
if request.POST.get('auth_otp') == profile.values('auth_otp')[count-1]['auth_otp']:
user = User.objects.create_user(
username=request.session['username'], password=request.session['password'], email=request.session['email'])
user.save()
auth.login(request, user)
profile.delete()
context = {'name': request.session['username']}
html_content = render_to_string(
'accounts/confirm_email.html', context)
text_content = strip_tags(html_content)

send_mail = EmailMultiAlternatives(
"Registration Successfull",
text_content,
settings.EMAIL_HOST_USER,
[request.session['email']]
)
send_mail.attach_alternative(html_content, 'text/html')
send_mail.send()
return redirect('home')
```

```
profile.delete()
messages.info(request, "Invalid otp or email register again!")
return redirect('register')
return render(request, 'accounts/verify.html')

@login_required(login_url='login')
def logout(request):
auth.logout(request)
return redirect('login')

@login_required(login_url='login')
def dashboard(request):
return render(request, 'accounts/dashboard.html')
```

1. **Pages - All the routs related to navigate inside web page is handled by this app.**

urls.py

```
from django.urls import path
from . import views
urlpatterns = [
path('', views.home, name='home'),
path('camera', views.camera, name='camera'),
path('result', views.result, name='result'),
path('screenshot', views.screenshot, name='screenshot'),
]
```

views.py

```
from skintone_app.settings import MEDIA_ROOT, MEDIA_URL
from django.conf import settings
from django.contrib.auth.decorators import login_required
from base64 import b64decode
from django.shortcuts import redirect, render
from .models import *
import uuid
from .skintone import *


def home(request):
    return render(request, 'pages/home.html')


def camera(request):
    if request.user.is_authenticated:
        return render(request, 'pages/camera.html')
    else:
        return redirect('login')


def upload_webcam_blob(blob, id):
    with open(f'media/uploads/{id}.png', 'wb') as fh:
        # Get only revelant data, deleting "data:image/png;base64,"
        data = blob.split(',', 1)[1]
        fh.write(b64decode(data))
        return fh


def screenshot(request):
    if request.method == "POST":
        id = str(uuid.uuid4())
        img = request.POST.get('image')
        upload_webcam_blob(img, id)
        link = f"uploads/{id}.png"
        image = Image_upload(img_id=id, image=link)
        image.save()
        request.session['id'] = id
        request.session['image'] = img
        return redirect('result')
    else:
        return redirect('camera')


def result(request):
    id = request.session['id']
    img = Image_upload.objects.filter(img_id=id).first()
    # url = f"{settings.BASE_DIR}/media/{img.image.url}"
    url = request.session['image']
    result = skintone(url)[0]
    return render(request, 'pages/result.html', {'image': img, 'result': result})
```

2. **Upload Image App - It contains all the routes related to image uploads.**

urls.py

```
from django.urls import path
from . import views


urlpatterns = [
 path('', views.upload, name='upload'),
]
```

views.py

```
from pages.skintone import skintone
from pages.views import result
from django.shortcuts import render
from .models import Image_data
from .form import ImageForm

import base64
```

```
def image_to_data_url(filename):
    ext = filename.split('.')[-1]
    prefix = f'data:image/{ext};base64,'
    with open(filename, 'rb') as f:
        img = f.read()
    return prefix + base64.b64encode(img).decode('utf-8')


def upload(request):
    if request.method == "POST":
        form = ImageForm(data=request.POST, files=request.FILES)
        if form.is_valid():
            form.save()
            obj = form.instance
            link = f"./{obj.image.url}"
            url = image_to_data_url(link)
            result = skintone(url)[0]
            return render(request, "upload/upload.html", {"obj": obj, 'result': result})
    else:
        form = ImageForm()
        img = Image_data.objects.all()
    return render(request, "upload/upload.html", {"img": img, "form": form})
```

3. **Contact Us App - It contains all the routes related to contact us functionality.**

urls.py

```
from django.urls import path
from . import views


urlpatterns = [
    path('', views.contact, name='contact'),
]
```

views.py

```
from django.shortcuts import redirect, render
from django.http import HttpResponse
from .models import Contactus


def contact(request):
    if request.method == 'POST':
        name = request.POST.get('Name')
        email = request.POST.get('Email')
        subject = request.POST.get('Subject')
        message = request.POST.get('Message')
        contact = Contactus(name=name, email=email,
                            subject=subject, message=message)
        contact.save()
        return render(request, 'pages/thankyou.html')
    return redirect('home')
```

4. **skintone.py**

```
import numpy as np
from sklearn.cluster import KMeans
from collections import Counter
import imutils
from matplotlib import pyplot as plt
import cv2
rgb_lower = [100,50,0]
rgb_higher = [255,219,172]

skin_shades = {
    'dark' : [rgb_lower,[140,110,66]],
    'mild' : [[140,110,66],[180,163,105]],
    'fair':[[180,172,105],rgb_higher],
}
convert_skintones = {}
for shade in skin_shades:
    convert_skintones.update({
        shade : [
            (skin_shades[shade][0][0] * 256 * 256) + (skin_shades[shade][0][1] * 256) +
skin_shades[shade][0][2],
            (skin_shades[shade][1][0] * 256 * 256) + (skin_shades[shade][1][1] * 256) +
skin_shades[shade][1][2]
        ]
    })

def extractSkin(image):
    img = image.copy()
    black_img = np.zeros((img.shape[0],img.shape[1],img.shape[2]),dtype=np.uint8)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    lower_threshold = np.array([0, 48, 80], dtype=np.uint8)
    print(lower_threshold)
    upper_threshold = np.array([20, 255, 255], dtype=np.uint8)
    print(upper_threshold)

    skinMask = cv2.inRange(img, lower_threshold, upper_threshold)
    skin = cv2.bitwise_and(img, img, mask=skinMask)
    return cv2.cvtColor(skin, cv2.COLOR_HSV2BGR)

def extractDominantColor(image, number_of_colors=1, hasThresholding=False):
    if hasThresholding == True:
        number_of_colors += 1
    img = image.copy()
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = img.reshape((img.shape[0]*img.shape[1]), 3)
    estimator = KMeans(n_clusters=number_of_colors, random_state=0)
    estimator.fit(img)
    colorInformation = getColorInformation(
        estimator.labels_, estimator.cluster_centers_, hasThresholding)
    return colorInformation


def removeBlack(estimator_labels, estimator_cluster):
    hasBlack = False
    occurance_counter = Counter(estimator_labels)
    def compare(x, y): return Counter(x) == Counter(y)
    for x in occurance_counter.most_common(len(estimator_cluster)):
        color = [int(i) for i in estimator_cluster[x[0]].tolist()]
        if compare(color, [0, 0, 0]) == True:
            del occurance_counter[x[0]]
            hasBlack = True
            estimator_cluster = np.delete(estimator_cluster, x[0], 0)
            break
    return (occurance_counter, estimator_cluster, hasBlack)

def getColorInformation(estimator_labels, estimator_cluster, hasThresholding=False):
    occurance_counter = None
    colorInformation = []
    hasBlack = False
    if hasThresholding == True:
        (occurance, cluster, black) = removeBlack(
            estimator_labels, estimator_cluster)
        occurance_counter = occurance
        estimator_cluster = cluster
        hasBlack = black
    else:
        occurance_counter = Counter(estimator_labels)
    totalOccurance = sum(occurance_counter.values())
    for x in occurance_counter.most_common(len(estimator_cluster)):
        index = (int(x[0]))
        index = (index-1) if ((hasThresholding & hasBlack)
                              & (int(index) != 0)) else index
        color = estimator_cluster[index].tolist()
        color_percentage = (x[1]/totalOccurance)
        colorInfo = {"cluster_index": index, "color": color,
                     "color_percentage": color_percentage}
        colorInformation.append(colorInfo)
    return colorInformation

def plotColorBar(colorInformation):
    color_bar = np.zeros((100, 500, 3), dtype="uint8")
    top_x = 0
    for x in colorInformation:
        bottom_x = top_x + (x["color_percentage"] * color_bar.shape[1])
        color = tuple(map(int, (x['color'])))
        cv2.rectangle(color_bar, (int(top_x), 0),
                      (int(bottom_x), color_bar.shape[0]), color, -1)
        top_x = bottom_x
    return color_bar

url = input("Enter image url :")

image = imutils.url_to_image(url)

image = imutils.resize(image, width=250)
plt.subplot(3, 1, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title("Original Image")

skin = extractSkin(image)
plt.subplot(3, 1, 2)
plt.imshow(cv2.cvtColor(skin, cv2.COLOR_BGR2RGB))
plt.title("Thresholded  Image")

unprocessed_dominant = extractDominantColor(skin, number_of_colors=1,
hasThresholding=True)

decimal_lower = (rgb_lower[0] * 256 * 256) + (rgb_lower[1] * 256) + rgb_lower[2]
decimal_higher = (rgb_higher[0] * 256 * 256) + (rgb_higher[1] * 256) + rgb_higher[2]
dominantColors = []
for clr in unprocessed_dominant:
    clr_decimal = int((clr['color'][0] * 256 * 256) + (clr['color'][1] * 256) + clr['color'][2])
    if clr_decimal in range(decimal_lower,decimal_higher+1):
        clr['decimal_color'] = clr_decimal
        dominantColors.append(clr)

skin_tones = []
if len(dominantColors) == 0:
    skin_tones.append('Unrecognized')
else:
    for color in dominantColors:
        for shade in convert_skintones:
            if color['decimal_color'] in
range(convert_skintones[shade][0],convert_skintones[shade][1]+1):
                skin_tones.append(shade)
```

```
print(skin_tones)

print("Color Bar")
colour_bar = plotColorBar(dominantColors)
plt.subplot(3, 1, 3)
plt.axis("off")
plt.imshow(colour_bar)
plt.title("Color Bar")

plt.tight_layout()
plt.show()
```

INSTRUCTIONS ON USE

- Modules

Modules required for Website: For using these app in you local environment you must have following modules installed in your system.

1. asgiref==3.3.4
2. astroid==2.4.2
3. autopep8==1.5.5
4. certifi==2020.12.5
5. colorama==0.4.4
6. cycler==0.10.0
7. dj-database-url==0.5.0
8. Django==3.2.2
9. django-heroku==0.3.1
10. gunicorn==20.1.0
11. imutils==0.5.4
12. isort==5.6.4
13. joblib==1.0.1
14. kiwisolver==1.3.1
15. lazy-object-proxy==1.4.3
16. matplotlib==3.4.2
17. mccabe==0.6.1
18. numpy==1.20.3
19. opencv-python==4.5.2.52
20. opencv-python-headless==4.5.2.52
21. Pillow==8.2.0
22. psycopg2==2.8.6
23. psycopg2-binary==2.8.6
24. pycodestyle==2.7.0
25. pylint==2.6.0
26. pyparsing==2.4.7
27. python-dateutil==2.8.1
28. pytz==2021.1
29. scikit-learn==0.24.2
30. scipy==1.6.3
31. six==1.15.0
32. skia-python==87.1
33. sqlparse==0.4.1
34. threadpoolctl==2.1.0
35. toml==0.10.2
36. whitenoise==5.2.0
37. wincertstore==0.2
38. wrapt==1.12.1

Some Modules like numpy, scikit-learn, opencv, etc should be manually installed. For installing these libraries the steps are given below-

1) **How to install NumPy** :-
   Open command prompt and write the command

```
pip install numpy
```

Or we can use a python distribution that already has NumPy installed like, Anaconda, Spyder etc.

2) **Downloading and Installing OpenCV**:

OpenCV can be directly downloaded and installed with the use of pip (package manager). To install OpenCV, just go to the command-line and type the following command:

```
pip install opencv-python
```

3) **How to install K-Means** :-

To use k-means we have to install scikit-learn and to install scikit-learn open command prompt and write the command

```
pip install -U scikit-learn
```

4) **We can install matplotlib as follows:**

```
pip install matplotlib
```

By executing it in anaconda prompt matplotlib will be installed in your pc.

- **How To Run**
  a. After installing all required modules you have to run following command from the project directory

  python manage.py migrate(only for first time)

  b. and then run following command

  ```
  python manage.py runserve
  ```

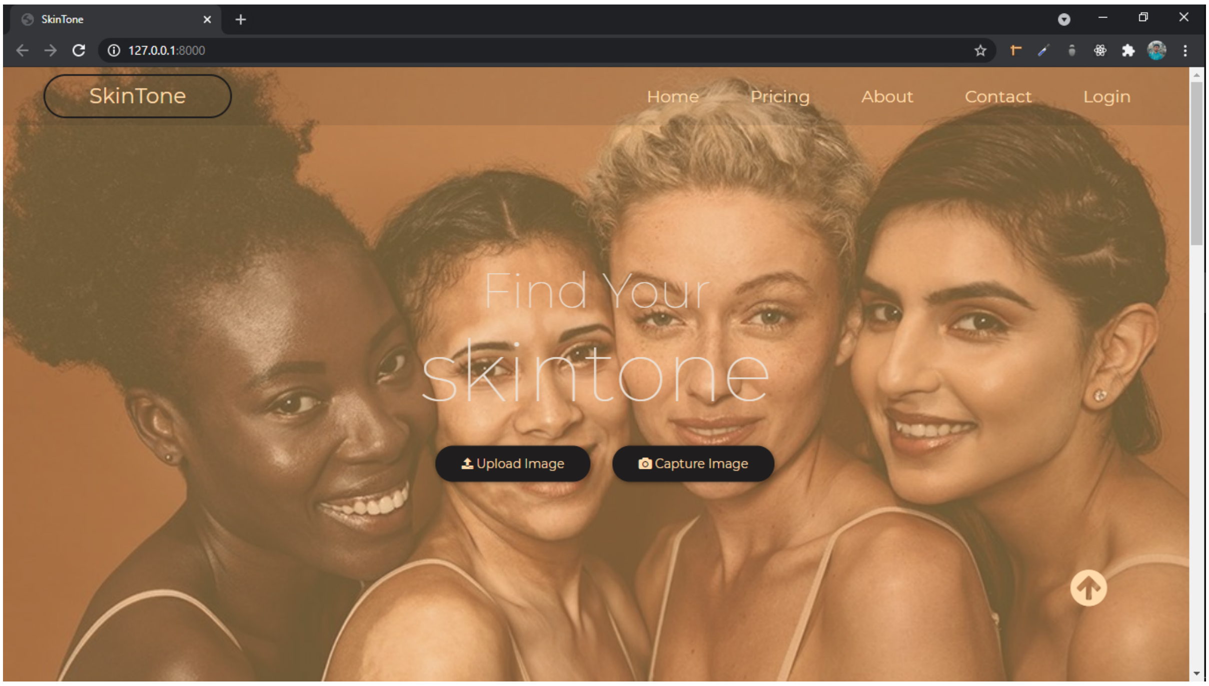
  c. Just type on your browser

  localhost:8000

  (or)

  http://127.0.0.1:8000/

Now you see a web page like this:



For the admin panel Just type on your browser

localhost:8000/admin

or

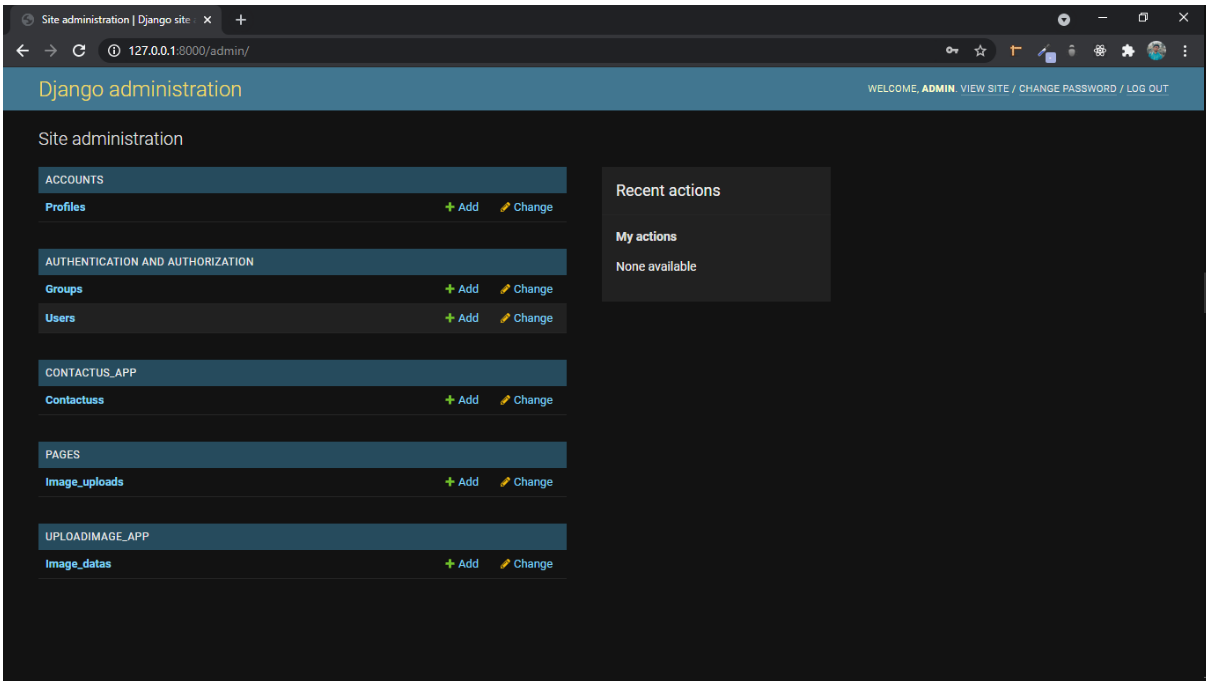
http://127.0.0.1:8000/admin

For using this website use the following username and password:

Username: admin

Password: admin@1234

Now you see following web page:



Django administration

WELCOME, **ADMIN**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

ACCOUNTS		
Profiles	+ Add	Change

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

CONTACTUS_APP		
Contactuss	+ Add	Change

PAGES		
Image_uploads	+ Add	Change

UPLOADIMAGE_APP		
Image_datas	+ Add	Change

Recent actions

My actions

None available

For live testing use *https://whispering-cliffs-35654.herokuapp.com*

SUMMARY/CONCLUSION

The Website is successfully accepting images both via uploading and also via capturing the photo. For Capturing Photo we need to authenticate to the website. The End result give you , your image the masked image and the skin tone color accurately.