Configuration Manual

MSc Research Project
Data Analytics

Sanica Kamble
Student ID: x21130701

School of Computing
National College of Ireland

Supervisor: Anderson Augusto Simiscuka

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sanica Kamble
Student ID:	x21130701
Programme:	Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Anderson Augusto Simiscuka
Submission Due Date:	15/12/2022
Project Title:	Configuration Manual
Word Count:	735
Page Count:	7

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sanica Kamble
Date:	28th January 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sanica Kamble
x21130701

1 Introduction

This paper's goal is to describe the coding procedure for the project. The hardware and software setups required to replicate the research in the future are described. This section describes the steps required to execute the script, as well as the programming and implementation procedures required for effective executable code.

2 Setup and Configuration

2.1 Hardware Configurations

The models used for this research project are processed on Windows 10 computer with a 2.40GHz 11th generation Intel Core i5 processor and 16GB RAM.

2.2 Software Configurations

1. Python 3.9.13 programming language is used for the development of the color models.
2. The code is run using Jupyter Notebooks created with Anaconda Navigator (anaconda3). The installation guide can be found on their official website[1].

3 Data Selection

Datasets have been selected and downloaded from three different data sources mentioned below:

1. FFHQ facial images dataset source - `https://github.com/NVlabs/ffhq-dataset`
2. Face Research Lab London dataser source - `https://figshare.com/articles/dataset/Face_Research_Lab_London_Set/5047666?file=8541955`
3. Foundation shades .csv dataset source - `https://www.kaggle.com/datasets/shivamb/makeup-shades-dataset`

[1]https://docs.anaconda.com/navigator/install/

4 Data Transformation

Background from the images is removed using PhotoRoom web application using this link `https://app.photoroom.com/batch`
Ground truth skin tone dataset is manually created by obtaining skin shade on **Dressika** Android application with is run on Windows 10 using BlueStacks emulator which can be installed from here - `https://www.bluestacks.com/` . After successful installation run the .exe file and follow the instructions on the screen to setup the application on the system. Once it is setup, **Dressika** app can be downloaded as shown below:

1. Go to the BlueStacks homepage and click on Google Play Store as shown in Figure 1

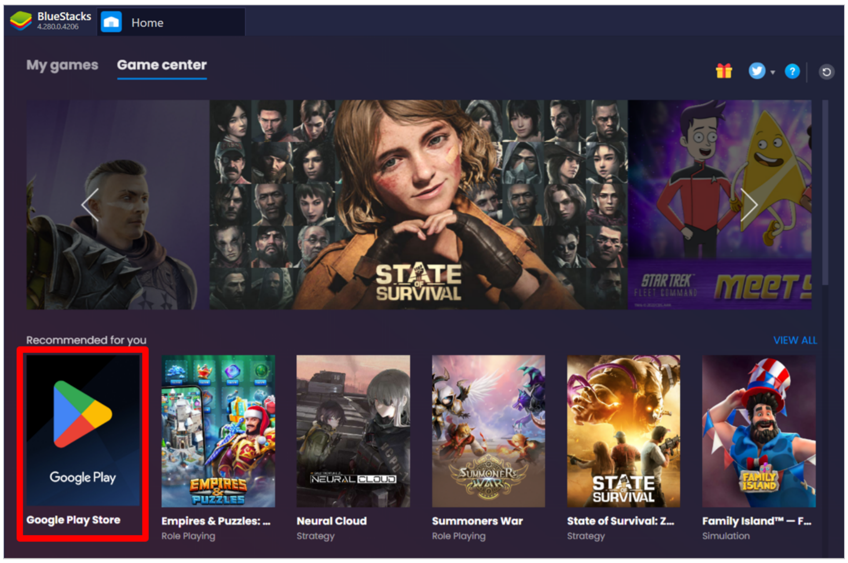


Figure 1: BlueStacks Homepage

2. Search for "dressika" and then click on "Open"/"Download" as shown in Figure 2

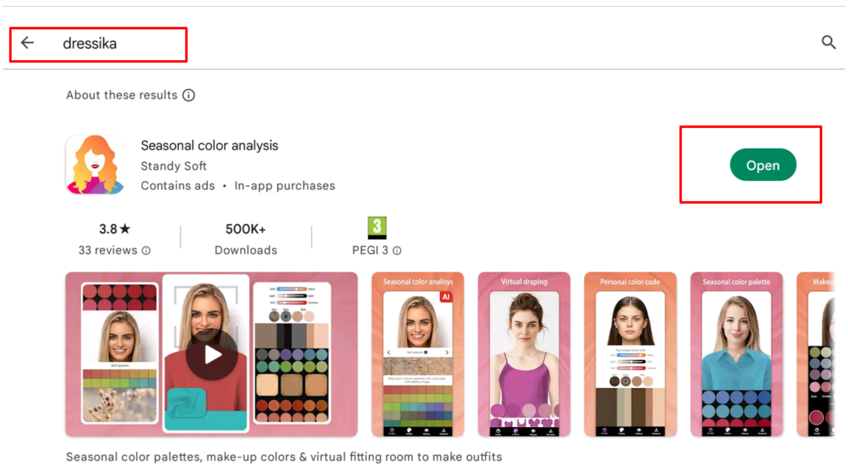
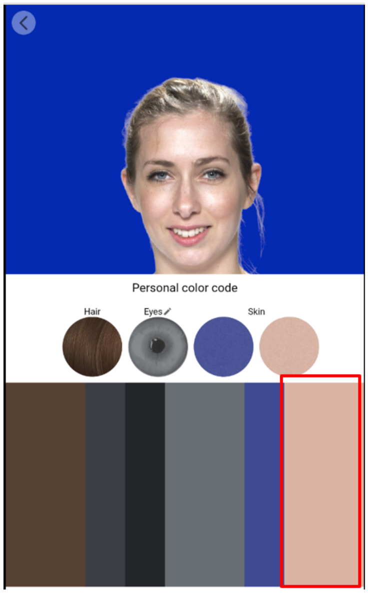


Figure 2: BlueStacks Google Play Store

3. Upload the facial image in the app and click on "Continue" as shown in Figure 3a and Figure 3b and obtain the ground truth skin tone shade. Get a screenshot of Figure 3b and using any color picker of your choice detect the skin tone HEX / RGB value and record it in csv file against image file name. Repeat the Steps for all the images to create the ground truth skin tone dataset.

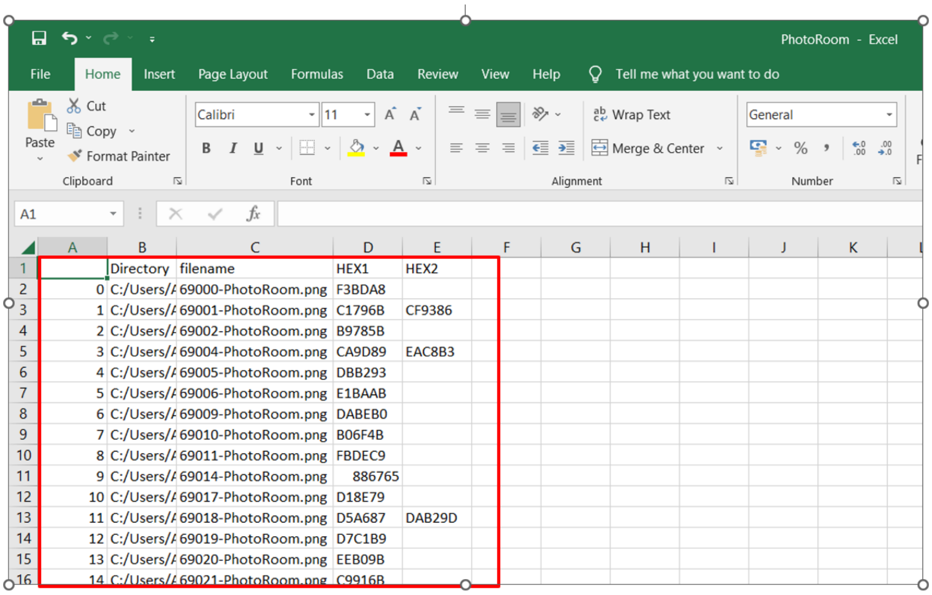
(a) Upload Image



(b) Get skin tone shade color

Figure 3: BlueStacks Dressika App

4. Figure 4 shows the groundtruth dataset .csv file for FFHQ dataset and same is to be followed for Face Research Lab London dataset.



	A	B	C	D	E
1		Directory	filename	HEX1	HEX2
2	0	C:/Users/	69000-PhotoRoom.png	F3BDA8	
3	1	C:/Users/	69001-PhotoRoom.png	C1796B	CF9386
4	2	C:/Users/	69002-PhotoRoom.png	B9785B	
5	3	C:/Users/	69004-PhotoRoom.png	CA9D89	EAC8B3
6	4	C:/Users/	69005-PhotoRoom.png	DBB293	
7	5	C:/Users/	69006-PhotoRoom.png	E1BAAB	
8	6	C:/Users/	69009-PhotoRoom.png	DABEB0	
9	7	C:/Users/	69010-PhotoRoom.png	B06F4B	
10	8	C:/Users/	69011-PhotoRoom.png	FBDEC9	
11	9	C:/Users/	69014-PhotoRoom.png	886765	
12	10	C:/Users/	69017-PhotoRoom.png	D18E79	
13	11	C:/Users/	69018-PhotoRoom.png	D5A687	DAB29D
14	12	C:/Users/	69019-PhotoRoom.png	D7C1B9	
15	13	C:/Users/	69020-PhotoRoom.png	EEB09B	
16	14	C:/Users/	69021-PhotoRoom.png	C9916B	

Figure 4: FFHQ skin tone ground truth dataset

5 Data Modelling

This section will cover the implementation of 3 color spaces models. All the models require OpenCV python Library and other required libraries are shown in Figure 5. Assuming these libraries are already installed in the python environment, if not they can be installed using pip install "libraryName" command.

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tqdm.notebook as tqdm
import skimage.color

import pandas as pd
import shutil
import os
import time
import matplotlib.colors
import re
from colormath.color_objects import sRGBColor, LabColor
from colormath.color_conversions import convert_color
from colormath.color_diff import delta_e_cie2000
import csv
import math
```

Figure 5: Required Python Libraries

5.1 HSC + YCrCb with histogram equalization and Otsu's image segmentation

1. Figure 6 is the code for HSV + YCrCb color spaces model.

```
img_bgr = cv2.imread(os.path.join(source_dir, file_name))
# get the start time
st = time.time()
#enter color model code here start
img_bgr = run_histogram_equalization(img_bgr)
img_grayscale = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
img_bgr = segment_otsu(img_grayscale, img_bgr)
img_hsv = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)
img_ycrcb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2YCrCb)


mask = (img_hsv[:, :, 0] <= 170) & \
    (img_ycrcb[:, :, 1] >= 140) & \
    (img_ycrcb[:, :, 1] <= 170) & \
    (img_ycrcb[:, :, 2] >= 90) & \
    (img_ycrcb[:, :, 2] <= 120)


img_bgr[~mask] = 0 #black pixels or non-skin

blue = np.ma.array(img_bgr[:, :, 0], mask=~mask).mean()
green = np.ma.array(img_bgr[:, :, 1], mask=~mask).mean()
red = np.ma.array(img_bgr[:, :, 2], mask=~mask).mean()
#end
result=red, green, blue
# get the end time
et = time.time()

# get the execution time
elapsed_time = et - st
```

Figure 6: HSV + YCrCb color spaces model

2. Histogram equalization and Otsu's image segmentation is performed as shown in Figure 7

```
def run_histogram_equalization(img_bgr):

    img_ycrcb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2YCrCb)
    img_ycrcb[:, :, 0] = cv2.equalizeHist(img_ycrcb[:, :, 0])
    img_bgr = cv2.cvtColor(img_ycrcb, cv2.COLOR_YCrCb2BGR)
    return img_bgr


def segment_otsu(image_grayscale, img_BGR):
    """Segment using otsu binarization and thresholding."""
    threshold_value, threshold_image = cv2.threshold(image_grayscale, 0, 255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
    threshold_image_binary = 1 - (threshold_image / 255)
    threshold_image_binary = np.repeat(threshold_image_binary[:, :, np.newaxis], 3, axis=2)
    img_face_only = np.multiply(threshold_image_binary, img_BGR).astype('uint8')
    return img_face_only
```

Figure 7: Histogram equalization and Otsu's image segmentation

5.2 HSV color space with Gaussian Blur

1. Figure 8 is the code for HSV color space and Gaussian blur

```
img_bgr = cv2.imread(os.path.join(source_dir, file_name))
# get the start time
st = time.time()
#enter color model code here start
 # Taking a copy of the image
img = img_bgr.copy()
# Converting from BGR Colours Space to HSV
img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# Defining HSV Threadholds
lower_threshold = np.array([0, 48, 80], dtype=np.uint8)
upper_threshold = np.array([20, 255, 255], dtype=np.uint8)

# Single Channel mask,denoting presence of colours in the about threshold
skinMask = cv2.inRange(img, lower_threshold, upper_threshold)

# Cleaning up mask using Gaussian Filter
skinMask = cv2.GaussianBlur(skinMask, (3, 3), 0)

# Extracting skin from the threshold mask
skin = cv2.bitwise_and(img, img, mask=skinMask)
blue = np.ma.array(cv2.cvtColor(skin, cv2.COLOR_HSV2BGR)[:, :, 0], mask=~skinMask).mean()
green = np.ma.array(cv2.cvtColor(skin, cv2.COLOR_HSV2BGR)[:, :, 1], mask=~skinMask).mean()
red = np.ma.array(cv2.cvtColor(skin, cv2.COLOR_HSV2BGR)[:, :, 2], mask=~skinMask).mean()
#end
result=red, green, blue
# get the end time
et = time.time()
```

Figure 8: HSV Color Space and Gaussian blur

5.3 HSV color space model

1. Figure 9 is the code for HSV color space model

```
img_bgr = cv2.imread(os.path.join(source_dir, file_name))
# get the start time
st = time.time()
#enter color model code here start
min_HSV = np.array([0, 58, 30], dtype = "uint8")
max_HSV = np.array([33, 255, 255], dtype = "uint8")
imageHSV = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2HSV)
skinRegionHSV = cv2.inRange(imageHSV, min_HSV, max_HSV)

skinHSV = cv2.bitwise_and(img_bgr, img_bgr, mask = skinRegionHSV)
blue = np.ma.array(skinHSV[:, :, 0], mask=~skinRegionHSV).mean()
green = np.ma.array(skinHSV[:, :, 1], mask=~skinRegionHSV).mean()
red = np.ma.array(skinHSV[:, :, 2], mask=~skinRegionHSV).mean()
#end
result=red, green, blue
# get the end time
et = time.time()

# get the execution time
elapsed_time = et - st
```

Figure 9: HSV Color Space Model

The above three color spaces models are implemented on each image of FFHQ (testing dataset) and Face Research Lab London datasets(validation dataset) to obtain model calculated RGB skin tone values. Which are then compare with the ground truth skin tone value obtained in section 4 for accuracy using Delta-E metric which is calculated using the code shown in Figure 10. Color model achieving lowest average Delta-E value

```
def calculateDeltaE(red,green,blue,hexVal):

    r=float(red)
    g=float(green)
    b=float(blue)

    color1_rgb = sRGBColor(r,g,b);

    h=checkString(hexVal)

    rgb2=hex_to_rgb(h)
    rgb2Strip=str(rgb2).strip("()")
    rgb2Split=rgb2Strip.split(",")

    color2_rgb = sRGBColor(rgb2Split[0],rgb2Split[1],rgb2Split[2]);

    # Convert from RGB to Lab Color Space
    color1_lab = convert_color(color1_rgb, LabColor);


    # Convert from RGB to Lab Color Space
    color2_lab = convert_color(color2_rgb, LabColor);


    # Find the color difference
    delta_e = delta_e_cie2000(color1_lab, color2_lab);

    #print("The difference between the 2 color = ", delta_e)
    return delta_e
```

Figure 10: Delta-E calculation

is selected to recommend foundation shade with foundation shades dataset .csv file as shown in Figure 11

```python
def getFoundationShade(red,gree,blue):
    filename = './shades.csv'


    with open(filename, 'r') as csvfile:
        csvreader = csv.reader(csvfile)
        a=[]
        dict= {}
        # This skips the first row of the CSV file.
        next(csvreader)
        datareader = csv.reader(csvfile)
        for row in datareader:
            dict['brand']=row[0]
            dict['product']=row[2]
            #print(row[4])
            dict['hex']=row[4]
            diff=calculateDeltaE(red,green,blue,checkString(row[4]))
            dict['diff']=diff
            a.append(dict.copy())

        newlist = sorted(a, key=lambda d: d['diff'])
        a=newlist[0]

    return newlist[0]
```

Figure 11: Foundation Shade Recommendation

The Delta-E results and foundation shade recommendation is then stored in the .csv file using csv file read and write operations in Python. The output .csv file is shown in Figure 12

	A	B	C	D	E	F	H	I	J	K	L	M	N	O	P	Q
1		Directory	filename	HEX1	HEX2	ModelRGB	ExecutionTime	HEX1	HEX2	Delta-E	GroundTruthHEX	ModelHEX	FoundationShade	Brand	Product	FoundationShadeDelta
2	0	C:/Users/A	001_08-Ph	#ebc0a6		(239.648758930207, 207.118414	0.51879406	#ebc0a6		11.14855	#ebc0a6	EFCFB7	edcfb9	Maybelline	Fit Me	2.000094382
3	1	C:/Users/A	002_08-Ph	#f0d5ca		(238.26466891656, 209.2127196	0.376368761	#f0d5ca		7.727418	#f0d5ca	EED1C3	e8c7b8	L'OrÃ©al	True Matc	4.842222161
4	2	C:/Users/A	003_08-Ph	#e5b7a6		(243.9892072577914, 206.69572	0.365788221	#e5b7a6		10.10753	#e5b7a6	F3CEBE	e8c7b8	L'OrÃ©al	True Matc	3.498926766
5	3	C:/Users/A	004_08-Ph	#e3b6ab		(239.17064177200913, 201.3623	0.341289043	#e3b6ab		8.29856	#e3b6ab	EFC9BD	dfb9af	Laws of Na	Foxy Finish	5.747739938
6	4	C:/Users/A	005_08-Ph	#eabba1		(239.2440350638701, 200.54103	0.321245193	#eabba1		7.070552	#eabba1	EFC8B1	eac4ae	Covergirl +	Simply Age	2.518714514
7	5	C:/Users/A	006_08-Ph	#bb8a71		(243.3085699150464, 210.71351	0.326348543	#bb8a71		29.17018	#bb8a71	F3D2BA	f7d5bb	Kate	Secret Skin	1.549878816
8	6	C:/Users/A	007_08-Ph	#e5bca2		(243.14779509164038, 209.0428	0.321308613	#e5bca2		11.20649	#e5bca2	F3D1B7	f7d5bb	Kate	Secret Skin	1.030641137
9	7	C:/Users/A	008_08-Ph	#b98a6c		(243.20536334852113, 203.7559	0.308642626	#b98a6c		20.20002	#b98a6c	F3CBB1	edc8af	Dior	Diorskin Fo	1.850907352
10	8	C:/Users/A	009_08-Ph	#e2b8a7		(244.7951099102284, 211.14514	0.334942341	#e2b8a7		13.70933	#e2b8a7	F4D3C1	f9d5c3	Make Up F	Ultra HD	2.740167886
11	9	C:/Users/A	010_08-Ph	#e6bfb0		(240.36086067325587, 208.0547	0.322276354	#e6bfb0		7.612053	#e6bfb0	F0D0C1	e8c7b8	L'OrÃ©al	True Matc	2.93633926
12	10	C:/Users/A	011_08-Ph	#f9d1c2	#ddaa9a	(248.0625414454132, 208.8584	0.310856342	#f9d1c2	#ddaa9a	6.869617	#f9d1c2	F8D0BF	f0cbb9	L'OrÃ©al	True Matc	3.764500178
13	11	C:/Users/A	012_08-Ph	#dca995		(244.87748455118813, 200.4110	0.34920454	#dca995		10.25916	#dca995	F4C8B9	f7c3b3	Make Up F	Ultra HD	4.9971516
14	12	C:/Users/A	013_08-Ph	#f6cdcc		(237.39980909255854, 200.5785	0.322172165	#f6cdcc		37.08221	#f6cdcc	EDC8BA	fcd6c7	L'OrÃ©al	Infalliable	3.402982429
15	13	C:/Users/A	014_08-Ph	#d19d89		(247.49494192792528, 209.8797	0.329814911	#d19d89		19.82119	#d19d89	F7D1BE	f9d5c3	Make Up F	Ultra HD	2.070156635
16	14	C:/Users/A	016_08-Ph	#cba28d		(242.8791880947057, 209.54056	0.351440907	#cba28d		14.31064	#cba28d	F2D1C1	f9d5c3	Make Up F	Ultra HD	2.79133508

Figure 12: .csv Output File

1. Columns **ModelRGB** and **ModelHEX** are the skin tone values obtained using color model.
2. Column **GroundTruthHEX** is ground truth skin tone value.