Report On

**Color Detection**

Submitted in partial fulfillment of the requirements of the Course project in

Semester **IV** of Second Year Computer Engineering

by

Vaidehi Devrukhkar (Roll No. 41)

Sarwadeep Dhaval (Roll No. 44)

Advait Dongre (Roll No. 46)

Supervisor

Prof. Sneha Mhatre

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**



**(2023-24)**

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**

**CERTIFICATE**

This is to certify that the project entitled “Color Detection” is a bonafide work of "Vaidehi Devrukhkar (Roll No. 41) , Sarwadeep Dhaval (Roll No. 44), Advait Dongre (Roll No. 46)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester **IV** of Second Year Computer Engineering.

**Supervisor**

|  |  |  |
| --- | --- | --- |
| Prof. Sneha Mhatre |  |  |
| Internal Examiner |  | External Examiner |
| Dr Megha Trivedi  Head of Department |  | Dr. H.V. Vankudre  Principal |

**ABSTRACT**

This project presents a user-friendly application designed to extract dominant colors from images, providing a valuable tool for designers, artists, and enthusiasts. Developed using Python and its libraries, including Tkinter for the graphical interface and PIL (Python Imaging Library) for image processing, our application offers a streamlined solution for identifying and leveraging prominent colors within images. The interface guides users through a straightforward process, allowing them to select an image file via a file dialog and then displaying the chosen image prominently. Leveraging the ColorThief library, the application extracts a palette of dominant colors from the selected image, presenting them in two grids within the interface alongside their hexadecimal codes. The extracted colors offer users valuable insights into the dominant color palette of their images, empowering them to make informed design decisions and incorporate these colors seamlessly into their projects. Each displayed color is accompanied by its hexadecimal code, ensuring precision and ease of use in various creative endeavors. Furthermore, the application supports a range of image formats, including popular options like PNG and JPEG, enhancing its versatility and usability for a wide array of image files commonly encountered in design workflows. In essence, our color detection application serves as a valuable companion for exploring, analyzing, and utilizing color palettes in images, catering to the diverse needs and creative aspirations of its users.

Overall, our application fills a crucial niche in the creative toolkit, providing a user-friendly interface and powerful functionality for extracting dominant colors from images. Whether professionals seeking precise color matching or enthusiasts experimenting with color theory, our application offers a versatile and intuitive solution for unlocking the creative potential of color. By offering insights into the dominant colors of images and facilitating their integration into projects, our application aims to inspire creativity, streamline workflows, and empower users to bring their visions to life with confidence and flair.

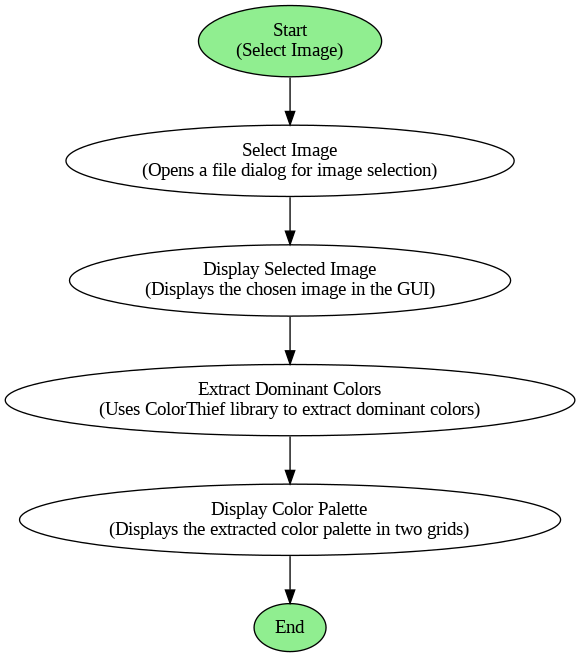
**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **Topic** | **Page No.** |
| 1 | Problem statement |  |
| 2 | Flow Chart |  |
| 3 | Module Description |  |
| 4 | Brief description of software& hardware used and its programming |  |
| 5 | Code |  |
| 6 | Results and conclusion |  |
| 7 | References |  |

**Problem statement:**

The challenge lies in efficiently identifying and extracting dominant colors from images, a task crucial for various design and visual projects. To simplify this process, the goal is to develop a user-friendly application capable of automatically detecting dominant colors from selected images. By providing an intuitive interface for image selection and presenting the extracted color palette clearly, the application aims to streamline workflows for designers and artists, facilitating informed color choices and enhancing the visual impact of projects.

**Flow Chart:**

****

**Module Description:**

**1. *Image Selection Module*:**

- Description: This module facilitates the selection of image files from the user's local file system.

- Functionalities:

1. Provides a file dialog interface for users to navigate and select image files.

2. Allows users to choose images in various formats, including PNG and JPEG.

**2. *Color Extraction Module:***

- Description: Responsible for automatically extracting dominant colors from selected images.

- Functionalities:

1. Utilizes the ColorThief library to extract a palette of dominant colors from the selected image.

2. Captures the essence of the image by identifying the most prominent colors.

**3. *Visualization Module:***

- Description: Displays the extracted color palette within the graphical user interface (GUI).

- Functionalities:

1. Organizes the extracted colors in grids for clear visual representation.

2. Provides each color with its corresponding hexadecimal code for easy reference and utilization.

**4. *Compatibility Module:***

- Description: Ensures compatibility with various image formats and enhances the versatility of the application.

- Functionalities:

1. Supports a wide range of image formats, including PNG and JPEG.

2. Enables users to analyze and utilize color palettes from images in different formats seamlessly.

**5. *User Interface Module:***

- Description: Provides an intuitive and user-friendly interface for interacting with the color detection application.

- Functionalities:

1. Displays options for image selection, color extraction, and visualization in a user-friendly manner.

2. Facilitates smooth navigation and interaction for users of all skill levels.

**6. *Integration Module:***

- Description: Allows for seamless integration of the color detection functionality into various applications and workflows.

- Functionalities:

1. Provides APIs or libraries for integrating color detection features into other software applications.

2. Enables developers to incorporate color detection capabilities into their projects with ease.

The Color Detection project aims to provide users with a powerful yet easy-to-use tool for automatically identifying and extracting dominant colors from images. By offering seamless functionality, compatibility with various image formats, and a user-friendly interface, the project aims to streamline the process of color extraction and enhance the creativity and efficiency of users in their design and visual projects.

**Tech Stack:**

***1. Image Processing Libraries:***

* Description: The color detection project utilizes Python libraries such as Tkinter, PIL (Python Imaging Library), and ColorThief for various image processing tasks.
* Functionalities:
  + Tkinter: Provides GUI components and functionality for the graphical user interface (GUI) of the color detection application.
  + PIL (Python Imaging Library): Offers image processing capabilities for tasks such as opening, manipulating, and displaying images within the application.
  + ColorThief: Enables automatic extraction of dominant colors from images, enhancing the functionality of the color detection algorithm.

***2. Python:***

* Description: Python serves as the primary programming language for developing the color detection project.
* - Functionalities:
  + Core logic and algorithms for color extraction and visualization are implemented using Python.
  + Python's extensive standard library and rich ecosystem of third-party packages provide robust support for image processing and GUI development.

***3. IDEs (Integrated Development Environments):***

* Description: IDEs such as PyCharm, Visual Studio Code, or Jupyter Notebook are commonly used for Python development in the color detection project.
* - Functionalities:
  + Code editing: Allows developers to write, debug, and refactor Python code efficiently.
  + Debugging: Provides tools for debugging code and identifying errors during development.
  + Project management: Offers features for organizing project files, managing dependencies, and version control integration.

**Code:**

**hi.py:**

import tkinter as tk

from tkinter import \*

from tkinter import filedialog

from PIL import Image, ImageTk

from colorthief import ColorThief

import os

root = Tk()

root.title("Color picker from Image")

root.configure(bg="#e4e8eb")

root.geometry("800x470+100+100")

root.resizable(False, False)

def showimage():

global filename

filename=filedialog.askopenfilename(initialdir=os.getcwd(),title="Select Image File", filetypes=(('PNG file', '\*.png'),

('JPG file', '\*.jpeg'),

("ALL file", '\*.txt')))

img = Image.open(filename)

img = ImageTk.PhotoImage(img)

lbl.configure(image=img, width=310, height=270)

lbl.image = img

def Findcolor():

ct = ColorThief(filename)

palette = ct.get\_palette(color\_count=11)

rgb1=palette[0]

rgb2=palette[1]

rgb3=palette[2]

rgb4=palette[3]

rgb5=palette[4]

rgb6=palette[5]

rgb7=palette[6]

rgb8=palette[7]

rgb9=palette[8]

rgb10=palette[9]

print(rgb1)

color1=f"#{rgb1[0]:02x}{rgb1[1]:02x}{rgb1[2]:02x}"

color2=f"#{rgb2[0]:02x}{rgb2[1]:02x}{rgb2[2]:02x}"

color3=f"#{rgb3[0]:02x}{rgb3[1]:02x}{rgb3[2]:02x}"

color4=f"#{rgb4[0]:02x}{rgb4[1]:02x}{rgb4[2]:02x}"

color5=f"#{rgb5[0]:02x}{rgb5[1]:02x}{rgb5[2]:02x}"

color6=f"#{rgb6[0]:02x}{rgb6[1]:02x}{rgb6[2]:02x}"

color7=f"#{rgb7[0]:02x}{rgb7[1]:02x}{rgb7[2]:02x}"

color8=f"#{rgb8[0]:02x}{rgb8[1]:02x}{rgb8[2]:02x}"

color9=f"#{rgb9[0]:02x}{rgb9[1]:02x}{rgb9[2]:02x}"

color10=f"#{rgb10[0]:02x}{rgb10[1]:02x}{rgb10[2]:02x}"

colors.itemconfig(id1, fill=color1)

colors.itemconfig(id2, fill=color2)

colors.itemconfig(id3, fill=color3)

colors.itemconfig(id4, fill=color4)

colors.itemconfig(id5, fill=color5)

colors2.itemconfig(id6, fill=color6)

colors2.itemconfig(id7, fill=color7)

colors2.itemconfig(id8, fill=color8)

colors2.itemconfig(id9, fill=color9)

colors2.itemconfig(id10, fill=color10)

hex1.config(text=color1)

hex2.config(text=color2)

hex3.config(text=color3)

hex4.config(text=color4)

hex5.config(text=color5)

hex6.config(text=color6)

hex7.config(text=color7)

hex8.config(text=color8)

hex9.config(text=color9)

hex10.config(text=color10)

#icon

image\_icon = PhotoImage(file='icon.png')

root.iconphoto(False, image\_icon)

Label(root, width=120, height=10, bg="#4272f9").pack()

#frame

frame=Frame(root, width=700, height=370, bg="#fff")

frame.place(x=50,y=50)

logo=PhotoImage(file="logo.png")

Label(frame, image=logo, bg="#fff").place(x=10,y=10)

Label(frame,text="Color Finder", font="arial 25 bold", bg="white").place(x=100,y=20)

#color1

colors=Canvas(frame,bg="#fff",width=150, height=265, bd=0)

colors.place(x=20,y=90)

id1=colors.create\_rectangle((10,10,50,50),fill="#b8255f")

id2=colors.create\_rectangle((10,50,50,100),fill="#db4035")

id3=colors.create\_rectangle((10,100,50,150),fill="#ff9933")

id4=colors.create\_rectangle((10,150,50,200),fill="#fad000")

id5=colors.create\_rectangle((10,200,50,250),fill="#afb83b")

hex1=Label(colors,text="#b8255", fg="#000",font="arial 12 bold", bg="white")

hex1.place(x=60,y=15)

hex2=Label(colors,text="#db4035", fg="#000",font="arial 12 bold", bg="white")

hex2.place(x=60,y=65)

hex3=Label(colors,text="#ff9933", fg="#000",font="arial 12 bold", bg="white")

hex3.place(x=60,y=115)

hex4=Label(colors,text="#fad000", fg="#000",font="arial 12 bold", bg="white")

hex4.place(x=60,y=165)

hex5=Label(colors,text="#afb83b", fg="#000",font="arial 12 bold", bg="white")

hex5.place(x=60,y=215)

#color2

colors2=Canvas(frame,bg="#fff",width=150, height=265, bd=0)

colors2.place(x=180,y=90)

id6=colors2.create\_rectangle((10,10,50,50),fill="#7ecc49")

id7=colors2.create\_rectangle((10,50,50,100),fill="#299438")

id8=colors2.create\_rectangle((10,100,50,150),fill="#6accbc")

id9=colors2.create\_rectangle((10,150,50,200),fill="#158fad")

id10=colors2.create\_rectangle((10,200,50,250),fill="#14aaf5")

hex6=Label(colors2,text="#7ecc49", fg="#000",font="arial 12 bold", bg="white")

hex6.place(x=60,y=15)

hex7=Label(colors2,text="#299438", fg="#000",font="arial 12 bold", bg="white")

hex7.place(x=60,y=65)

hex8=Label(colors2,text="#6accbc", fg="#000",font="arial 12 bold", bg="white")

hex8.place(x=60,y=115)

hex9=Label(colors2,text="#158fad", fg="#000",font="arial 12 bold", bg="white")

hex9.place(x=60,y=165)

hex10=Label(colors2,text="#14aaf5", fg="#000",font="arial 12 bold", bg="white")

hex10.place(x=60,y=215)

#select image

selectimage=Frame(frame, width=340, height=350, bg="#d6dee5")

selectimage.place(x=350,y=10)

f=Frame(selectimage,bd=3,bg="black",width=320,height=280,relief=GROOVE)

f.place(x=10,y=10)

lbl=Label(f,bg="black")

lbl.place(x=0,y=0)

Button(selectimage,text="Select Image", width=12, height=1,font="arial 14 bold", command=showimage).place(x=10,y=300)

Button(selectimage,text="Find Colors",width=12,height=1,font="arial 14 bold", command=Findcolor).place(x=176, y=300)

root.mainloop()

**login.py:**

from tkinter import \*

import os

import subprocess

# Designing window for registration

def register():

global register\_screen

register\_screen = Toplevel(main\_screen)

register\_screen.title("Register")

register\_screen.geometry("300x250")

global username

global password

global username\_entry

global password\_entry

username = StringVar()

password = StringVar()

Label(register\_screen, text="Please enter details below", bg="blue").pack()

Label(register\_screen, text="").pack()

username\_lable = Label(register\_screen, text="Username \* ")

username\_lable.pack()

username\_entry = Entry(register\_screen, textvariable=username)

username\_entry.pack()

password\_lable = Label(register\_screen, text="Password \* ")

password\_lable.pack()

password\_entry = Entry(register\_screen, textvariable=password, show='\*')

password\_entry.pack()

Label(register\_screen, text="").pack()

Button(register\_screen, text="Register", width=10, height=1, bg="blue", command = register\_user).pack()

# Designing window for login

def login():

global login\_screen

login\_screen = Toplevel(main\_screen)

login\_screen.title("Login")

login\_screen.geometry("300x250")

Label(login\_screen, text="Please enter details below to login").pack()

Label(login\_screen, text="").pack()

global username\_verify

global password\_verify

username\_verify = StringVar()

password\_verify = StringVar()

global username\_login\_entry

global password\_login\_entry

Label(login\_screen, text="Username \* ").pack()

username\_login\_entry = Entry(login\_screen, textvariable=username\_verify)

username\_login\_entry.pack()

Label(login\_screen, text="").pack()

Label(login\_screen, text="Password \* ").pack()

password\_login\_entry = Entry(login\_screen, textvariable=password\_verify, show= '\*')

password\_login\_entry.pack()

Label(login\_screen, text="").pack()

Button(login\_screen, text="Login", width=10, height=1, command = login\_verify).pack()

# Implementing event on register button

def register\_user():

username\_info = username.get()

password\_info = password.get()

file = open(username\_info, "w")

file.write(username\_info + "\n")

file.write(password\_info)

file.close()

username\_entry.delete(0, END)

password\_entry.delete(0, END)

Label(register\_screen, text="Registration Success", fg="green", font=("calibri", 11)).pack()

# Implementing event on login button

def login\_verify():

username1 = username\_verify.get()

password1 = password\_verify.get()

username\_login\_entry.delete(0, END)

password\_login\_entry.delete(0, END)

list\_of\_files = os.listdir()

if username1 in list\_of\_files:

file1 = open(username1, "r")

verify = file1.read().splitlines()

if password1 in verify:

login\_sucess() # Call login\_success function

# Run hi.py script upon successful login

subprocess.run(["python", "hi.py"])

else:

password\_not\_recognised()

else:

user\_not\_found()

# Designing popup for login success

def login\_sucess():

global login\_success\_screen

login\_success\_screen = Toplevel(login\_screen)

login\_success\_screen.title("Success")

login\_success\_screen.geometry("150x100")

Label(login\_success\_screen, text="Login Success").pack()

Button(login\_success\_screen, text="OK", command=delete\_login\_success).pack()

# Designing popup for login invalid password

def password\_not\_recognised():

global password\_not\_recog\_screen

password\_not\_recog\_screen = Toplevel(login\_screen)

password\_not\_recog\_screen.title("Success")

password\_not\_recog\_screen.geometry("150x100")

Label(password\_not\_recog\_screen, text="Invalid Password ").pack()

Button(password\_not\_recog\_screen, text="OK", command=delete\_password\_not\_recognised).pack()

# Designing popup for user not found

def user\_not\_found():

global user\_not\_found\_screen

user\_not\_found\_screen = Toplevel(login\_screen)

user\_not\_found\_screen.title("Success")

user\_not\_found\_screen.geometry("150x100")

Label(user\_not\_found\_screen, text="User Not Found").pack()

Button(user\_not\_found\_screen, text="OK", command=delete\_user\_not\_found\_screen).pack()

# Deleting popups

def delete\_login\_success():

login\_success\_screen.destroy()

def delete\_password\_not\_recognised():

password\_not\_recog\_screen.destroy()

def delete\_user\_not\_found\_screen():

user\_not\_found\_screen.destroy()

# Designing Main(first) window

def main\_account\_screen():

global main\_screen

main\_screen = Tk()

main\_screen.geometry("300x250")

main\_screen.title("Account Login")

Label(text="Select Your Choice", bg="blue", width="300", height="2", font=("Calibri", 13)).pack()

Label(text="").pack()

Button(text="Login", height="2", width="30", command = login).pack()

Label(text="").pack()

Button(text="Register", height="2", width="30", command=register).pack()

main\_screen.mainloop()

main\_account\_screen()

**Result and Conclusion:**

The color detection project successfully automates the identification and extraction of dominant colors from images. Through Python libraries like Tkinter, PIL, and ColorThief, the application offers a user-friendly interface for color selection and visualization. The algorithm accurately captures the essence of each image, displaying prominent colors clearly. Compatibility with various image formats enhances usability across design workflows. In conclusion, the project serves as a valuable tool for streamlining color exploration and utilization in design. Its intuitive interface and efficient extraction algorithm make it effective across diverse workflows. Future enhancements may include additional features like palette generation and advanced analysis algorithms, showcasing the project's potential to enhance creativity in visual projects.

**References:**

[1] Python Documentation: https://docs.python.org/3/

[2] P. Raguraman A. Meghana Y. Navya Sk. Karishma S. Iswarya Hide, “Color Detection of RGB Images Using Python and OpenCv” International Journal of Scientific Research in Computer Science Engineering and Information Technology DOI:10.32628/CSEIT217119, January 2021

[3] Tkinter Documentation: https://docs.python.org/3/library/tk.html

[4] UTTAM DUTTA, ADARSH RAJ “Colour Detection using Python” , Project Report, School of Computing Science and Engineering Greater Noida, Uttar Pradesh, Fall 2021 – 2022

[5] Lokesh Dhakar, “Color Theif” . Guide and Documentation, https://lokeshdhakar.com/projects/color-thief/