**[Color Detection Web Application]**

**Group Name: Python\_Killers**

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **First name** | **Last Name** | **Student number** |
| Bruno | Andrade | C0796309 |
| Giuliana | Gois | C0789227 |
| Nishanth | Parthasarathy | C0828612 |

**Submission date: *[December 17, 2021]***

Contents

[Abstract 3](#_Toc4571289)

[Introduction 3](#_Toc4571290)

[Methods 3](#_Toc4571291)

[Results 3](#_Toc4571292)

[Conclusions and Future Work 4](#_Toc4571293)

[References 4](#_Toc4571294)

# **Abstract**

The objective of this project is to display an image in a web application that the user would be able to click anywhere and automatically displays the color name in the coordinates respective to the image where it was selected.

# **Introduction**

The project aims to identify a set of colors from an image. Additionally, a web application page was designed to upload pictures and execute the color detection functionality.

The challenge of this project is to develop an app of color detection image to the final user. The main idea of it, is to deliver a program which the user would be able to select a picture and upload it to the API (application programming interface), chose whatever pixel available in the image, and then the program would display the color name and RGB number.

# **Methods**

The project was built in three steps:

1. Backend development - The functionalities to upload the files and capture the mouse event clicks were created as REST API using FastApi() in the Backend.
2. Frontend development - Where Users will be able to interact with API’s created to upload an image and find out the colors selected in the image.
3. An Open cv algorithm to find the R, G, B value of the coordinates in the image and find the color name from a database.

* **Backend development**

The Backend folder has two important python files, in which they are “main.py” and “colorDetector.py”.

Under the file colorDetector, we have used two functions: “*recognize\_color()”* and *“mouse\_click()”.*

The first function, called “recognize\_color()” uses the R, B, G values as input from mouse\_click() and return the color name corresponding to the RGB code.

def recognize\_color(R, G, B):  
 minimum = 10000  
 for i in range(len(csv)):  
 d = abs(R - int(csv.loc[i, "R"])) + abs(G - int(csv.loc[i, "G"])) + abs(B - int(csv.loc[i, "B"]))  
 if (d <= minimum):  
 minimum = d  
 cname = csv.loc[i, "color\_name"]  
 return cname

The second function, called “*mouse\_click()*” finds the R, G, B value respective to the given X” and “Y” coordinates in the image and return the RGB values.

def mouse\_click(x, y, img):  
 B, G, R = img[y, x]  
 return int(R), int(G), int(B)

Now under the second file named “main.py” we also have created two functions. They are: “*get\_image()”* and *“detect\_color()”*. Also, we are using these libraries, cv2, uvicorn, fastapi, pydantic, PIL and OS:

import cv2  
import uvicorn  
from fastapi import File  
from fastapi import FastAPI  
from fastapi import UploadFile  
from pydantic import BaseModel  
from PIL import Image  
import os

The “get\_image()” function has the logic to receive the file submitted on webpage application. This function returns a success or failure status.

# save the image file  
@app.post("/upload")  
async def get\_image(file: UploadFile = File(...)):  
 image = Image.open(file.file).convert('RGB')  
 try:  
 image.save(f"./storage/{file.filename}")  
 return {"status": "success", "message": "Uploaded Successfully", "filename": file.filename}  
 except OSError as e:  
 return {"status": "failed", "message": "Uploaded Failed retry again\n" + str(e), "filename": file.filename}

Looking at the “detect\_color()” function, we are receiving the X, Y and IMAGE\_NAME as value to return the color name.

# get x,y,image\_name return the color name  
@app.post("/detect")  
async def detect\_color(body: colorDetect):  
 img = cv2.imread(f"./storage/{body.image\_name}")  
 color\_names = []  
 for index, value in enumerate(body.x):  
 r, g, b = colorDetector.mouse\_click(int(value), int(body.y[index]), img)  
 color\_names.append(colorDetector.recognize\_color(r, g, b))  
 return {"color\_names": color\_names}

* **Frontend development**

The Frontend folder has one python file named “main.py”. Under the main, we have used pandas, streamlit, PIL and OS.

import requests  
import pandas as pd  
import streamlit as st  
from PIL import Image  
from streamlit\_drawable\_canvas import st\_canvas  
import os

PIL (Python Imaging Library) was one important library for this phase of implementation, because they add support for opening, and saving many different images file formats.

Streamlit also was the most important library in that phase, it allowed us to implement a web app.

Therefore, we have created a logic to check if the image was uploaded or not and returned the value of X and Y in the list. The most challenging part in front-end was to display the image and ability to select a particular location in the image and map the coordinates to exact location in the image. Which after multiple attempts to use opencv-python window function, we found the best library, streamlit-drawable-canvasa canvas that notes the clicked location in “Left”/X and “Top”/Y axis .

if image is not None:  
 files = {"file": (image.name, image.getvalue())}  
 if not uploaded:  
 res = requests.post(f"{url}upload", files=files)  
 st.title(res.json()['message'])  
 if res.json()['status'] == "success":  
 uploaded = True  
 else:  
 uploaded = False  
  
 if uploaded:  
 bg\_image = Image.open(image)  
 width, height = bg\_image.size  
  
 canvas\_result = st\_canvas(  
 stroke\_width=3,  
 background\_image=bg\_image,  
 height=height,  
 width=width,  
 drawing\_mode="circle",  
 key="color\_annotation\_app",  
 )  
 df = pd.json\_normalize(canvas\_result.json\_data["objects"])  
  
 if df.size > 0:  
 detect\_resp = requests.post(f"{url}detect",  
 json={"x": df['left'].tolist(),  
 "y": df['top'].tolist(),  
 "image\_name": res.json()['filename']  
 })  
  
 df['color\_name'] = pd.Series(detect\_resp.json()['color\_names'])  
 st.dataframe(df[['left', 'top', 'color\_name']])

**Project Features:**

* A docker image of the project was created locally in which the frontend and backend components were separated containers.
* The FastApi framework was used to build the web application.
* The Streamlit library was applied for image handling.

**Web Application:**

Graphical user interface, application

Description automatically generated

A picture containing graphical user interface

Description automatically generated

**Results**

The Final product developed is a prototype that is container deployment ready with the ability to find exact color in a position if the image for multiple times highly efficient without need to send the image every time we need to detect color as it has been saved in a centralized location. We found that this type of modularity helped in improving scalability and efficient hardware resource usage as the main advantage.

# **Conclusions and Future Work**

This prototype satisfies all the requirements set in the project with a CI/CD workflow ready to be made for the next improvements like adding a database to backend to store the images in a cloud file storage, developing front-end file management system to select which file to be picked. And other improvements to front-end user interface

# **References**

Behic Guven (September 12, 2020) – Building a Color Recognizer in Python - Article  
<https://towardsdatascience.com/building-a-color-recognizer-in-python-4783dfc72456>

Image-Color-Detection (December 16, 2021) – GitHub Project Repository

<https://github.com/Image-Color-Detection/image-color-detection>

Streamlit - documentation

<https://docs.streamlit.io>

Streamlit-drawable-canvas – GitHub website

<https://github.com/andfanilo/streamlit-drawable-canvas>