**Hackathon Submission Template(Level-1-Solution)**

**Use Case Title**: Color Detection from Images  
**Student Name**: M Saravanan   
**Register Number**: 621523104143  
**Institution**: Mahendra College of Engineering  
**Department:** Computer Science and Engineering  
**Date of Submission:** 5-9-2025



### **1. Problem Statement**

*In creative fields like design, branding, and UI/UX development, professionals often need to identify the exact color used in digital images. Manually finding these colors is inefficient and error-prone. There is a lack of intuitive tools that allow users to click on an image and instantly receive the color information in both human-readable (name) and technical (RGB) formats. This project aims to solve that problem by building a simple and interactive application for real-time color detection from uploaded images.*

### 

### **2. Proposed Solution**

*We propose a web-based Color Detection Application where users can upload any image, click on any part of it, and instantly get the RGB value and the closest matching color name. The detected color will be visually shown in a reference rectangle, making it easier to compare and reuse. The tool will use OpenCV to process image inputs and detect pixel values, while a predefined color dataset will help match RGB values to standard color names. A clean Streamlit-based interface will ensure ease of use.*

### 

### **. 3 Technologies&Tools Considered**

**. *. Programming Language:*** *Python*

*· ·* ***Libraries:*** *OpenCV, Pandas, NumPy*

*· ·* ***Framework:*** *Streamlit (for web deployment)*

*· ·* ***Dataset:*** *colors.csv (with predefined color names and RGB values)*

*· ·* ***Version Control:*** *Git & GitHub*

*· ·* ***Optional (for enhancements):*** *scikit-learn (for improved color matching)*

### **Solution Architecture&Workflow**

***High-Level Workflow:***

*Image Upload: User uploads an image using Streamlit’s file uploader.*

*Image Display: The uploaded image is rendered using OpenCV.*

*Pixel Click Detection: When the user clicks on the image, OpenCV captures the x, y coordinates*

*RGB Extraction: The color at the selected pixel is extracted.*

*Color Matching: The RGB value is matched to the nearest color in colors.csv.*

*UI Display: Color name, RGB values, and a color preview box are shown in real time.*

***Major Components:***

*Image Uploader (Streamlit)*

*Image Renderer&Click Detector (OpenCV)*

*RGB&Color Name Extractor (Python + CSV matching)*

*Real-Time Display Panel (Streamlit)*

1. **Feasability&Challenges**

***Feasibility:***

*This project is highly feasible for a hackathon setting. All the required tools and libraries are lightweight and open-source. OpenCV and Streamlit are both easy to set up and integrate.*

***Challenges:***

*Accurate Color Matching: Minor differences in RGB values may map to wrong names.*

*Solution: Use color distance algorithms (e.g., Euclidean distance).*

*Click Event Mapping: Mapping OpenCV click events inside a Streamlit canvas can be tricky.*

*Solution: Use OpenCV’s built-in mouse event handlers outside Streamlit or integrate with Streamlit components.*

*UI Responsiveness: Maintaining real-time response while processing large images.*

*Solution: Resize large images or use optimization techniques.*

### **Expected Outcome&Impact**

*A fully functional app that helps users detect colors instantly with high accuracy.*

*Enhances productivity for designers, developers, and educators by simplifying the color extraction process.*

*Helps maintain visual consistency in branding or UI design by identifying correct color codes.*

*A user-friendly, deployable tool that can be accessed via the web (Streamlit hosted app).*

### **7. Future Enhancements**

### *Add HEX code and CMYK conversions.*

### *Allow image zooming for pixel-accurate selection.*

### *Let users build their own color palettes from selected points.*

### *Support color-blind friendly filters.*

### *Enable mobile-friendly interface and support drag-and-drop uploads.*

### *Add support for detecting multiple colors in a selected region (not just 1 pixel).*